Building Robot Hands and Teaching Dexterity

Kenneth Shaw

CMU-RI-TR-TO FILL OUT-NN

November 23, 2023

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

Thesis Committee:
Prof. Deepak Pathak, *Carnegie Mellon University* (Chair)
Prof. Nancy Pollard, *Carnegie Mellon University*
Prof. David Held, *Carnegie Mellon University*
Murtaza Dalal, *Carnegie Mellon University*

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

Abstract

Try typing on your keyboard, hammering a nail, or using chopsticks. Our hands are the key to manipulating the world around us. They have incredible strength at the fingertips to be able to pinch and grasp in over 70 different ways. Its ability to feel is unparalleled. This great sensing and adaptation is controlled by the great capabilities of our brains. In fact, the development of our brains is often attributed to the need to manipulate the world around us using our hands [99].

In robotics, manipulation has been mostly limited to the claw gripper or suction cup for pick-and-place in factories. However, our shared dream in robotics has always been to have robots cohabitate with us. These human-like robots should be able to complete similar tasks that humans do. Why don't robotic humanoids with useful robot hands exist?

While there are a few robot hands available today, the popular opinion is that they are difficult to use, expensive, and hard to obtain. We argue that this is not an inherent problem of robot hands, but rather that these robot hands are not built using the right design principles. In this thesis, we introduce a new class of robot hands that are significantly more dexterous, lower cost, and easier to use than prevailing robot hands. Our robot hands serve as proof that robot hands can be practical for real-world manipulation. They are fully open-sourced with example code and infrastructure and are continuing to serve as a significant entry point for dexterous manipulation research.

Second, how can robot hands imitate the human brain's ability to complete dexterous tasks in a human-like fashion? For instance, we must firmly grasp a knife from the handle and not from the blade. While most robots used today have fewer than 10 degrees of freedom, a humanoid with two hands has over 50 degrees of freedom with many points of contact with the environment. This high dimensionality makes data-efficient learning extremely difficult. To rectify this, we leverage internet-scale human experience from the web as training data. Because robot hands have a similar morphology as human hands, we can directly learn from retargeting human motion to teach robots fluidly. In this thesis, we find that this unlocks the generalizable, human-like behavior we seek.

Acknowledgments

This thesis would not have been possible without everyone in Prof. Deepak's and Prof. Abhinav's labs. Deepak's lab is truly a bustling place where every student is always thinking, sharing and collaborating on new ideas. Many of the ideas and directions of my thesis would not be possible without the lab and I would like to thank everyone in it.

I would also like to thank my MSR advisor: Prof. Deepak Pathak. When I first joined CMU, I remember Deepak had that contagious excitement and optimism which I instantly wanted to be a part of. Despite COVID, our lab built fast.

Admittedly, the first year of my MSR was hard. It is easy to dream in one's head about robot hands working like human hands. But as Hans Moravec says, "the hard things are easy; easy things are hard." Easy-to-use robot hand hardware was non-existent. The high-dimensional, contact-rich nature of robot hands made robot learning incredibly hard. Many things that we believed would work, did not.

While many may have given up, Deepak kept fighting: constantly coming up with new ideas and solutions to even the lowest level of problems for me and Aravind Sivakumar. Many people say that the hardest thing about research is finding the right problem to solve. I think he has a sixth sense for knowing the next big idea to drive towards. The goal that he instills in us is to not just publish but to publish big results and ideas. Deepak is also a very curious person and open to wildly new things. We had no idea LEAP Hand would work when we started. Through all of this, he has been the most helpful advisor I could ask for.

Learning from internet videos is very difficult to do successfully, but Shikhar Bahl has always found clever ways to make these real-world robots truly work. He tirelessly made sure these ideas came into reality. He has a special knack for leading projects and can make sure they get to publication. I owe a lot to him in the learning from human motion part of my work.

It may seem that Aravind Sivakumar and I only collaborated on one paper, but it was a whole year of research exploration. We figured out key ideas about robot hands, their hardware and where machine learning works. Also, because he came from a stint in industry, he made great visualizations, figures and code comments that I learned a lot from.

Prof. Nancy Pollard's lab has been a great collaboration. From working with

them, I learned how to think about mechanism design and brainstorm properly. I never did mechanical engineering in my education, but I like to dabble in the area with their help. We leverage one of the greatest strengths of CMU with this, its wide variety of robotics research.

Brian Hutchinson, our lab assistant, also deserves a special shout-out. I send him all sorts of strange lab requests and he incredibly makes quick work of all of them. These hardware-intensive projects would not be possible without him.

I would also like to thank Prof. Sonia Chernova and Prof. Harish Ravichandar at Georgia Tech. That lab was the most supportive group of people ever. They helped me immensely as an undergrad and instilled that research bug in me. I learned so much from their kind mentorship and I really hope to collaborate with them again in the future.

Many of my current friends may not know this, but my first love for robotics formed with my advisors Edward Petrillo, Bob McKillip, Mark Hillman, and Raj Pejaver at Team 293 in Hopewell Valley Central High School. Dr. Petrillo had an academic research focus, constantly making our robot mechanisms and building processes better: CNCs, mills, lathes, laser cutters you name it. He mentored us high schoolers with patience and kindness. Mr. McKillip taught me a lot of electronics skills and always pushed me towards the right practices. Always document, always verify, always justify your results. A lot of my intuitions in hardware are owed to this robotics team.

Of course, I would also like to thank my family. Michelle Zhao is always my biggest supporter and helps me through the ups and downs of research. My dad has this intellectual curiosity that never settles down. He likes anything from history to computers. My mom can have a hyper-focus, always organized, and never ever gives up. While this may not be true, I like to think I got parts of both of them: The curiosity of my dad and the grit of my mother to drive me forward.

I would also like to thank the NSF Graduate Research Fellowship under Grant No. DGE2140739. The work was supported in part by NSF IIS-2024594, ONR N00014-22-1-2096, ONR MURI N00014-22-1-2773, GoodAI Research Award, DARPA Machine Common Sense, Samsung GRO Research Award, and Air Force Office of Scientific Research (AFOSR) FA9550-23-1-0747.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Think about your hands for a minute. Just this morning you probably brushed your teeth, cooked some eggs, poured a cup of coffee, typed on a keyboard and maybe did some pullups. Easy right? Our hands can move smoothly from fine-grained actions to lifting a heavy weight all in a day's work. Our fingers can close extremely tightly on a pin, but then stretch around a large bottle to grasp it. We can softly grasp an egg, and then firmly grasp a hammer. Our hands can even withstand years of hard use without breaking a bone or ligament. While this is amazing, our hands can also feel throughout their fingers and palms. This touch can be used to interact with tools, ensure we grasp objects tightly, or even identify objects.

Human hands are incredible biological "hardware" that has been developed from millions of years of evolution and Darwinism [79]. We have 14 phalanges for our fingers, and 13 bones in our wrist and palm carefully arranged. We have a variety of joints, including the MCP, PIP and DIP in the fingers held together with ligaments and lubricated with synovial fluid. For actuation, we have many muscles connected to tendons which are connected to bones. The sense of touch, temperature, pain and pressure is routed through the Radial nerve, Median nerve and Ulnar nerve. Together, this multitude of parts unlocks great hand capabilities [56, 120, 205].

Processing all of this touch and visual information and controlling our hands is the human brain, another marvel of evolution. In fact, the development of the hand helped

promote the development of our intelligence in evolution [233]. From a young age as babies, we are constantly exploring. As quickly as a few months old, we learn how to control our fingers together and hold tight onto things. Additionally, our parents can teach using demonstration in a short amount of time [58, 98, 148]. This learning through grasping helps develop our brains into adolescence [30]. We then generalize experiences to new scenarios, a key part of intelligence [99, 233]. With this combination of exploration and demonstration, our brains learn how to complete over 80 different grasps in everyday life [153].

It is easy to imagine this dexterity in robots, especially in robotic humanoids. C-3PO walks the galaxies to make witty banter with the Star Wars characters. Back in the 1960s, the Jetsons imagined Rosey the humanoid helper doing chores for a family.

However, real robots have been dominated by 2 finger grippers or suction cups on robot arms. These robots have seen much success in industrial settings such as in warehouses or factories. In this setting, most of these tasks are limited to pick and place which can be performed by a claw. The argument is that these end-effectors are easy to build and program, and in these limited settings this is all that is needed.

There has been recent interest in robotic humanoids that can complete real-world tasks in our homes. In this case, robot hands must often need hands to use tools, complete more complex manipulation and manipulate objects designed for humans. In the early 2000s, Asimo was often seen demoing soccer, walking up stairs or pouring coffee with its hands [6]. The Boston Dynamics Atlas, while showing incredible acrobatics, does not have any hands [7]. Tesla Optimus can balance on one foot, but its manipulation so far has been limited to sorting large toddler-sized LEGO blocks [20] . In general, the locomotion capabilities of these humanoids have been increasingly robust. However, the capabilities of their hands are still woefully behind the capabilities of human hands.

In this thesis, we try to make robot hands more capable. First, how can we efficiently emulate as much of the human hand ”hardware” as possible? How can we build robot hands that are anthropomorphic, strong, and inexpensive for anyone to use? Second, we must process sensing information as the human brain does. The solution here lies in improving the robot brain, learning from diverse human videos and real-world experience.

1.2 Building Robot Hands

The prevailing belief of the robotics community has been that replicating the human hand kinematic structure is extremely difficult. As discussed, the human hand is very complicated and is not easy to emulate in robot hardware. However, we still would like our robot hands to have a small size, many degrees of freedom, and incredible strength/durability. How can we design anthropomorphic robot hands intelligently to have dexterity?

The conventional robot hands available today have prevented ubiquitous usage. The main purchasable hands have been the Shadow Hand [3] and Allegro Hand [5] for quite a few years. They are $15k and $150k respectively but are still considered to be notoriously difficult to use except for a few experts. On the other hand, there are a number of custom robot hands that show results in their own lab but cannot be reproduced. This is because they are very complicated to build or are not open-source [60, 128, 235]. Finally, there are a few open-source robot hands such as Inmoov [16] and DexHand [10], but these are often too weak or have too few DOF.

I believe that good robot hands should be everywhere. How do we do this? Similar to the popular 2-finger gripper counterpart, robot hands must be strong and repairable. Robot hands must be low-cost and easy to obtain. We need them to be extremely dexterous and anthropomorphic so that they can complete a variety of manipulation tasks successfully.

Following this strong belief, I introduce two dexterous hands. LEAP Hand is a useful robot hand that costs under $2k, is 3D printable, easy to assemble, and contains a full simulation. I release not just the hardware, but assembly instructions and a suite of useful software tools. DASH hand is a tendon-driven smaller hand. While it has more parts, it is still relatively easy to assemble and is very good for teleoperation and fine-grained manipulation.

Just a few months after its release, LEAP Hand has a large community around it and is one of the leading robot hands in academic research. Students all around the world are using it for their machine learning research such as tool use, human-like grasping or in-hand reorientation. This has enabled many researchers who were not involved in dexterous manipulation to get their first experience with it at a lower cost without difficult engineering. We have created a whole ecosystem of useful tools and a community around this platform. LEAP Hand is proof that following these design principles can help truly democratize robot hands.

1.3 Learning Dexterity

How do we emulate the human brain to control robot hands? Conventional robotics says to use first principles in physics and create an intricate model of the robot and the world. While this works often in low level controls or locomotion, it is difficult to model the intricate contacts of the robot hand on objects. And how do we even estimate the object state and its parameters? While there has been some early work in this area going back to the 90s, this does not generalize to new objects or increasingly complex behaviors [125, 145, 146].

Recently, there have been great developments in robot learning especially in learning from demonstration or experience. In the most naive approach, a human demonstrates to the robot how to complete a task. The robot then learns a policy on the behavior seen in the data. While this often works, the key problem is that robots will struggle to generalize to test scenarios it hasn't seen before in training. This necessitates humans laboriously collecting demonstrations for every task and environment that the robot might encounter which does not scale. To get around this, learning from real-world autonomous experience can be useful, it is often too slow to learn in the real world with.

Our key insight is that we do not have to collect demonstration data for every task that we want the robot to complete. Dexterous hand behavior happens all the time by humans. To get these dexterous demonstrations, one way is through internet video. The internet contains a wide variety of videos of humans using their hands in a huge variety of scenarios. In this thesis, our robot hands learn from this human behavior.

While it would be appealing to learn from any human behavior, often there is a large embodiment gap between human hands and robots. However, because our robot hands are designed to be anthropomorphic, there is a much smaller gap. Still, we must learn how to retarget, or to traverse this gap between human and robot hands and convert the behavior from one to the other. Our key insight is that our retargeters can actually learn from human videos and experiences from the internet. This enables us to learn from noisy or hard-to-detect human behavior.

First, we learn a teleoperation system from a single RGB webcam. Our key insight is to use models trained from internet-data to guide our robot to perform useful behavior. We call this "Robotic Telekinesis". Second, we find that learning dexterous policies to complete tasks from direct human demonstration alone requires too many demos. How can we improve this sample efficiency? In VideoDex, we use internet videos and extract

robot trajectories from them to use as pseudo-robot experience. We then use this data as a prior to pretrain robot behavior. Finally, in DEFT, we learn an affordance prior from human motion in video. This affordance learns where and how to grasp objects only from human video. Then, we use real-world learning to learn a residual policy that fine-tunes our robot behavior. These learning from human methods and the data they leverage are integrally important to teaching dexterity.

Part I

Designing Dexterous Robot Hands

Chapter 2

LEAP Hand: Low-Cost, Efficient, and Anthropomorphic Hand for Robot Learning

2.1 Abstract

Dexterous manipulation has been a long-standing challenge in robotics. While machine learning techniques have shown some promise, results have largely been currently limited to simulation. This can be mostly attributed to the lack of suitable hardware. In this paper, we present LEAP Hand, a low-cost dexterous and anthropomorphic hand for machine learning research. In contrast to previous hands, LEAP Hand has a novel kinematic structure that allows maximal dexterity regardless of finger pose. LEAP Hand is low-cost and can be assembled in 4 hours at a cost of 2000 USD from readily available parts. It is capable of consistently exerting large torques over long durations of time. We show that LEAP Hand can be used to perform several manipulation tasks in the real world—from visual teleoperation to learning from passive video data and sim2real. LEAP Hand significantly outperforms its closest competitor Allegro Hand in all our experiments while being 1/8th of the cost. We release detailed assembly instructions, the Sim2Real pipeline and a development platform with useful APIs on our website at https://leap-hand.github.io/

Figure 2.1: **Relative size of popular robot hands to scale.** *Left to right,* adult human hand, Allegro Hand [5], LEAP-C Hand, LEAP Hand, Inmoov [16], D'Manus [51]. LEAP Hand is similar in size to Allegro and $\sim 30\%$ larger than a human hand. D'Manus is considerably larger than the rest. Because of the tendon-driven nature, Inmoov is the smallest robotic hand. The hands are accurate to scale.

2.2 Introduction

Hand dexterity has been critically responsible for human cognition through active manipulation, tool use, and governing how humans learn from the world [58, 98, 148]. Replicating the dexterity of the human hand with a robot hand has been a long-standing challenge in robotics. Machine learning techniques have recently shown promise in areas such as learning from humans. However, unlike the learning successes in locomotion [31, 170] across truly diverse terrains, robotic manipulation results in the real world have mostly been limited to one degree-of-freedom parallel jaw grippers [48, 82, 222]. In contrast, dexterous manipulation has largely been limited to simulation [71, 116] with comparatively fewer real-world results [36, 72, 110, 171, 214].

A major bottleneck in democratizing dexterous manipulation has been the hardware. Tendon-based hands like Shadow [3], while impressively capable [36], cost over 100K USD and often require significant maintenance [34] due their complicated nature. While Inmoov [16] is inexpensive and open source, it only has 5 actuators on weak tendons. Therefore, direct-driven hands have been the popular alternative for many applications [5, 138]. The Allegro Hand has been a popular direct-driven hand, but it is often unreliable, difficult to repair, and does not have an anthropomorphic kinematic structure, (see Fig 2.1) and is expensive at over $16K. Please see Section 2.3 for further analysis.

As a result, only a few labs have access to hardware capable of complex dexterous tasks. This is in stark contrast to 2-finger grasping or locomotion [4, 13, 14, 27] where

readily available hardware allows results to be easily reproduced and improved upon by the community. Following this analogy, good hand hardware for machine learning must be durable, repeatable, low-cost, versatile, and ideally anthropomorphic to enable easy transfer learning from humans.

We propose LEAP Hand– a dexterous, extremely *low-cost* and *robust* hand for robot learning, built from off-the-shelf or 3D-printed parts. Our hand can be assembled in under 4 hours at a cost of 2000 USD, which is 1/8th the cost of the Allegro Hand and 1/50th to that of ShadowHand. While we acknowledge this is still not affordable for all, we believe it is a step towards democratizing dexterous manipulation research. We show through a number of rigorous experiments that LEAP Hand is both robust, durable, and able to exert large torques over long periods of time. Additionally, our robot hand it is easily repairable in-house just using a standard $250 3D printer and does not need to be sent out for repair.

Although robustness and low-cost is critical, they should not come at the cost of dexterity and anthropomorphism. We believe a good versatile hand is the one that is both *dexterous* as well as *anthropomorphic* because much of the world around us, for instance, doors, kitchens, tools, or instruments, are designed with human hands in mind, making it easier to learn by watching humans act. In LEAP Hand, we aim to maximize dexterity while being kinematically similar to a human hand.

Since the ball joint at the human knuckle (Metacarpophalangeal aka MCP) cannot be replicated with direct-driven hands, it must instead be approximated using two separate motors. Prior work in direct driven hands has converged primarily to two designs, see Fig 2.2, one that allows abduction-adduction of fingers in open hand pose and the other that only allows with the finger flexed upwards. However, both of these lose one degree of freedom (DoF)—either in the flexed or extended position of the finger. In LEAP Hand, we propose a new kinematic mechanism to facilitate *universal abduction-adduction* for direct-driven hands that retain all degrees of freedom in all finger positions. We demonstrate that this leads to higher dexterity and for improved grasping and in-hand manipulation.

Finally, we show that LEAP Hand easily integrates with existing results in robot learning. For instance, YouTube video-based learned teleoperation and behavior cloning. In addition to the physical robot hardware, we also release an Isaac Gym-compatible simulator for the LEAP Hand and show sim2real transfer for a contact-rich task of blind in-hand rotation of a cube [160]. This shows that the hardware and simulation are accurate and that complex tasks trained in simulation can be transferred to the real hand. We **open**

source the URDF model, assembly instructions, ROS/Python API, mapping methods from human hands to LEAP Hand, and an Isaac Gym simulation environment at https://leap-hand.github.io/.

2.3 Related Work

Robot Hands Shadow [3] and ADROIT [137] hands paved the way to enable complex, contact-rich dexterous tasks with an anthropomorphic ball joint MCP. [34, 36]. However, they are costly (100k USD) and require constant maintenance. In contrast, the Inmoov hand is 3D printable, tendon-driven, and human-like[16]. It has only one DoF per finger and is reliant on tendon actuation which is difficult to calibrate and can be inaccurate. Bauer et. al. [47] present a soft tendon-driven hand that is very flexible for many configurations. Unfortunately, it is difficult to simulate due to its deformable nature [90, 93]. In contrast to tendon-driven hands, which have motors in the wrist, the Allegro Hand [5] has its motors in the finger joints. It is most popular in research labs [38, 109, 217, 220, 225] because it is relatively cheap (16k USD). However, users find the motors in the fingers to be weak for many everyday tasks. Moreover, the closed-source components are difficult to repair or replace. Additionally, its kinematic structure is not anthropomorphic or dexterous as we demonstrate. The ROBEL suite (which includes D'Manus) [51] is more robust, open-sourced, and easy to build. However, it has only two fingers and a thumb—making it significantly different from a human hand. Yuan et. al. [241] accomplish within-hand manipulation using rollers attached to the fingers. Humans lack this degree of freedom and manipulate objects in a very different manner. [8, 127, 128] have shown impressive results and hold a lot of promise by using fluids and linear actuators to move the fingers, but these hands are not readily available and too complicated to quickly produce, use, and maintain for robot learning.

Rapid Manufacturing Aluminum machining is traditionally used to create strong parts but is prohibitively difficult and expensive. Manufacturing plastic parts includes a cumbersome process of mold making, casting, curing, and support removal [21]. In contrast, additive manufacturing can be used to create parts very quickly for prototyping. In our paper, we leverage recent advancements in extruders, hot-ends, and motors made by the open-source Reprap Community [22]. This allows us to directly print soft flexible filaments like Ninjaflex [2]. We use this to create many parts for LEAP Hand like the hard

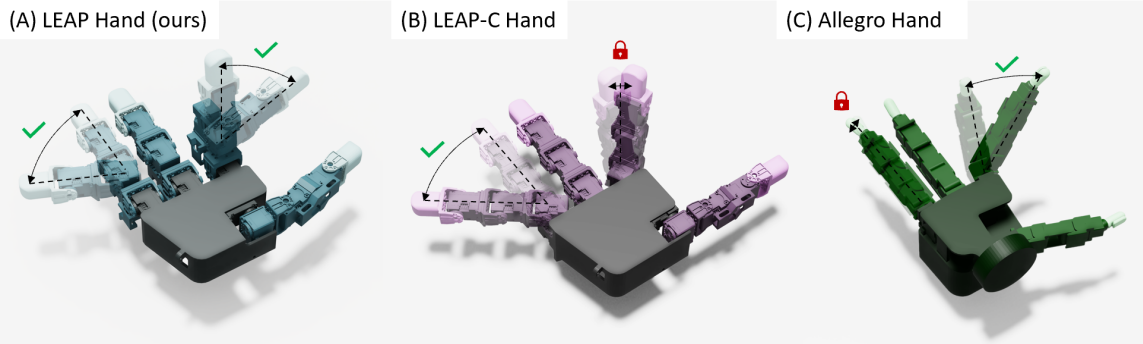

Figure 2.2: Comparison of MCP joints in different robot hands and their dexterity and two different positions. (A) In LEAP-C Hand there is a large range of motion at extended but not flexed position (B) In LEAP Hand, at flexed and extended positions, the fingertip has a large range of motion. (C) In Allegro, there is a large of motion at flexed but not extended position.

palm and soft rubber fingertips.

Learning Dexterity Using a Shadow hand and Sim2real, Andrychowicz et. al. [34, 36, 134] accomplish in-hand rotation for a variety of objects. Policies that scale to thousands of objects can also be trained in simulation [71, 115]. [175] uses the D'Hand to reposition a valve. In-hand rotation of Baoding Balls using the Shadow Hand trained purely in the real-world [173], and pipe insertion using the D'Hand [103] are other notable examples of dexterous manipulation.

Several recent works focus on supervising policies of robot hands [95, 123, 200, 230]. from MANO [199] parameters which parameterize a human hand. Closely related is the teleoperation of robot hands from real-time video [109, 216], which can be used to guide learning and improve sample-efficiency [192, 196]. Hand poses can be extracted from video data available on the web to learn manipulation policies [163, 192]. Large-scale pre-training using internet videos is helpful for efficiently training robot hands for downstream tasks using a few task specific-demos [212] and also on non-dexterous manipulation [45, 179].

2.4 Kinematic Design and Analysis

The kinematic structure of a hand refers to the arrangement of its joints which determines the different poses and forces of motion it can apply. First, LEAP Hand should be as anthropomorphic as possible so that data from humans can be used to learn skills [45, 192, 212] with machine learning. This can be done using methods such as teleoperation with

VR gloves or extracting keypoints from videos of human hands [216]. In addition, LEAP Hand should be dexterous for tasks such as in-hand manipulation from sim2real. In this section, we propose a robot hand design that is both anthropomorphic and dexterous.

In the human hand, there are four main degrees of freedom in each finger (Fig. 2.3). The knuckle or metacarpophalangeal (MCP) is a ball joint with two degrees of freedom that allows abduction/adduction and flexion/extension. The joint closest to the knuckle is called the proximal interphalangeal (PIP) joint. The last joint, closest to the fingertip, is the distal interphalangeal (DIP). The PIP and DIP are hinge joints, each with one degree of freedom. The human hand features an opposable thumb which allows the application of force in opposition to other fingers. This enables a variety of power and precision grasps [153]. To easily map motions, a robot hand must have analogous joints to a human hand.

To replicate this structure, it is alluring to use tendons like robot hands such as the ShadowHand [3]. Such tendon-driven hands can store the large motors needed to drive them in the wrist, enabling greater flexibility in joint design and introduce ball joints. However, they are very expensive (100K USD), complicated or hard to maintain. As a result, cheaper direct-driven alternatives [5, 138] have been more popular.

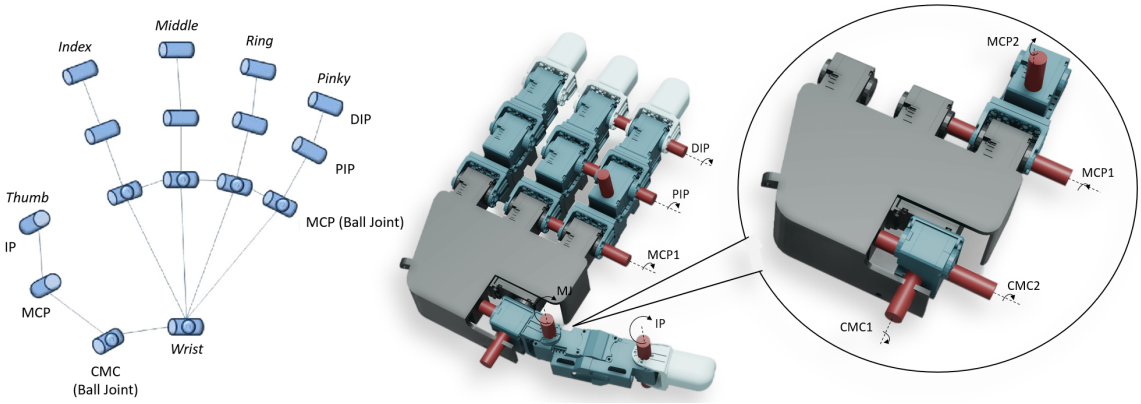

Figure 2.3: The human hand kinematics above has ball joints at the MCP and CMC joints. These are difficult joints for low-cost hands to include. Left Figure from [65]. Comparison of MCP joints in different robot hands. (A) In LEAP-C Hand there is a large range of motion at extended but not flexed position (B) In LEAP Hand, at flexed and extended positions, the fingertip has a large range of motion. (C) In Allegro, there is a large of motion at flexed but not extended position.

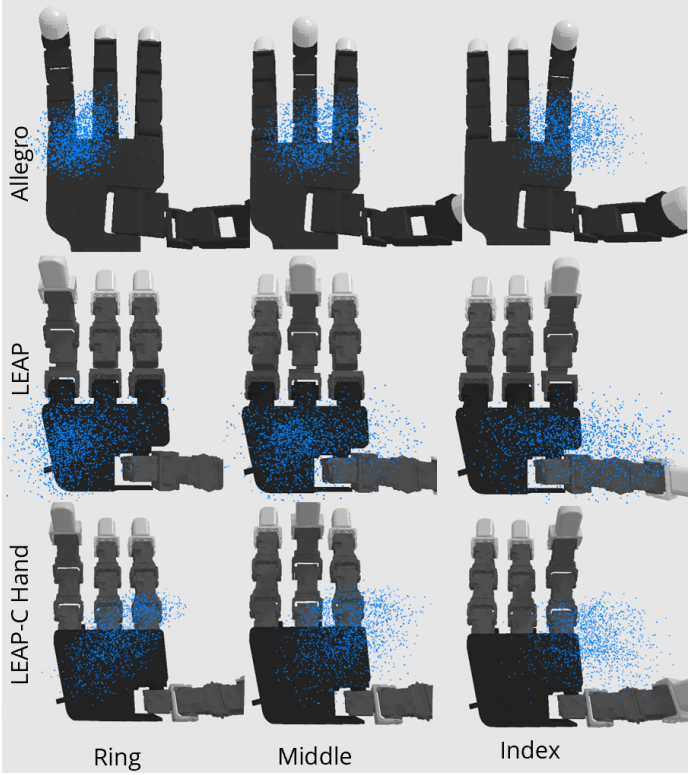

Figure 2.4: We compare the possible positions of opposability of the thumb and each of the other fingers on each of the three hands. We find that LEAP Hand has the best even spread on top of the palm and a very large contact area.

2.4.1 Universal Abduction-Adduction Mechanism

Direct-driven hands must store the motors inside the fingers so they are limited in kinematic structure and cannot precisely imitate the human hand. Since the PIP and DIP joints are hinge joints, they are easily modeled, each with a single actuator. A ball joint cannot be modeled in this way and is typically approximated using two motors (MCP-1, MCP-2) arranged close together [138]. Prior seminal work has proposed two designs for this (Fig. 2.2). However, both of these designs, Allegro and LEAP-C Hand, lose one degree of freedom in either the extended or closed position. As a result, Allegro is less dexterous when extended whereas LEAP-C Hand (like C-Hand in [138]) is less dexterous when closed.

The reason for the lost dexterity in both LEAP-C Hand and Allegro is that the axis of the motor responsible for adduction-abduction (MCP-2) is fixed to the palm of the hand. In

LEAP-C Hand, the axis is perpendicular to the plane of the palm, whereas, in Allegro, it lies in the plane of the palm. Thus, when the finger becomes parallel to this axis, that DoF is ineffective. Please see Figure 2.2 and the kinematic tree in the supplemental.

In LEAP Hand, we propose a new ***universal abduction-adduction mechanism*** for the fingers such that they can retain all degrees of freedom at all MCP positions. Instead of the MCP-2 axis being fixed to the palm (i.e., of motor responsible for adduction-abduction), the key idea is to bring the axis to the frame of reference of the first *finger* joint and arrange it such that it is always perpendicular to it. This allows the finger to have adduction-abduction in all positions (Fig. 2.2). Thus, LEAP Hand has adduction-abduction in the extended position (similar to LEAP-C Hand) as well as pronation/supination in the flexed position (similar to Allegro).

2.4.2 Evaluating Manipulability via Thumb Opposability

Chalon et.al. [66] and Lee et.al. [138], have shown that what makes a hand more versatile is not merely the degree of abduction-adduction but also its thumb opposability volume. We test our design against Allegro and LEAP-C Hand, a baseline hand we manufacture with the same motors and parts as LEAP Hand. In Fig. 2.4, we plot the intersection of the thumb and finger workspaces for each hand and compute a thumb opposability metric [66]. In Table 2.2, we show that LEAP Hand combined with the new MCP joints in the secondary fingers is better placed and is more dexterous compared to other available hands because of the increased opposable volume.

Next, manipulability measures the ease with which the fingertip can be moved in various directions at a particular joint pose. We use metrics introduced by Yoshikawa et. al. [239]. To evaluate this, many calculate the manipulability ellipsoid from the end-effector Jacobian which models the directions in which the end-effector can move. We compute the volume of this ellipsoid using the following equation:

$$w = \sqrt{\det\left(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^{\mathrm{T}}\right)}$$

where $\boldsymbol{q}$ is the joint configuration and $\boldsymbol{J}$ is the Jacobian of the end-effector. Note that because we are calculating volumes, a hand that can only move in one or two cartesian directions will have a volume close to zero at that pose. In three key poses, we show that LEAP Hand has consistently larger ellipsoids of greater volume for both the cartesian and

Robot/Position	Down (m^3)	Up (m^3)	Curled (m^3)
Allegro Hand			
Linear	8.11×10^{-9}	3.98×10^{-13}	2.39×10^{-5}
Angular	0	0	0
LEAP-C Hand			
Linear	1.60×10^{-12}	1.23×10^{-10}	9.28×10^{-5}
Angular	1.02×10^{-13}	1.02×10^{-9}	2.02×10^{-13}
LEAP Hand (ours)			
Linear	2.02×10^{-6}	2.42×10^{-6}	4.51×10^{-5}
Angular	1.20×10^{-5}	1.20×10^{-5}	1.20×10^{-5}

Table 2.1: We show the manipulability ellipsoid volume for both the linear and angular component at three different finger positions, down, all the way up, and then halfway/curled. We find that LEAP Hand has a large manipulability ellipsoid at all three configurations.

angular components of the jacobian. This means that LEAP Hand has better movement at the fingertips in these few poses and a higher manipulability metric leading to more dexterity (Table 2.2).

Finally, we show that increased dexterity leads to practical benefits as well. In the grasping test (Sec. 2.7.1), we find that LEAP Hand is able to grasp more objects tightly. In the blind in-hand cube rotation task (Sec. 2.7.4), we find that LEAP Hand is able to rotate the cube much faster than Allegro.

2.5 Hand Design Principles

A good kinematic design must be realized effectively in hardware. In particular, the hardware should be low-cost, easy to repair, and robust.

2.5.1 Low-cost and Easy to Repair

In contrast to locomotion or manipulation with two-fingered grippers, real-world research in dexterous manipulation has been limited. This can be attributed in large part to the lack of suitable dexterous hand hardware. Commonly used dexterous hands such as ShadowHand [3] and AllegroHand [5] cost 100K and 16K USD, respectively, and must be

Opposability Vol.	Index (mm^3)	Middle (mm^3)	Ring (mm^3)
Allegro	409,135	348,809	204,281
LEAP-C Hand	834,516	743,764	638,605
LEAP Hand (ours)	**1,125,556**	**1,056,746**	**804,618**

Table 2.2: We show the finger-to-thumb opposability volume in mm^3 by randomly sampling 25,000 joint configurations and finding the instances at which both fingers touch and recording that contact point. The volume of this area of contact is calculated and reported.

sent back to manufacturers for repair in case of damage. This hardware is out-of-budget or impossible to maintain for many researchers allowing only a small fraction to work on real-world dexterous manipulation. On the other hand, due to the availability of cheap and reliable locomotion [4, 14] and manipulation [13, 25, 27] hardware, a large community of researchers is able to build off of each others' work and drive progress.

A suitable hand should therefore be as accessible as possible. This implies that it should be low-cost and easy to repair. In LEAP Hand, we accomplish this by using as many off-the-shelf parts as possible and fabricating the rest using only a commodity 3D printer that costs around 200 USD. It can be assembled in under 4 hours.

LEAP Hand is designed to be modular. This allows key features of the robot hand to be changed, such as the length or number of fingers and the distances between each of the fingers in the palm for particular learning tasks or for analysis. Additionally, the modularity makes the hand easily repairable with only a few distinct parts.

2.5.2 Robustness

Learning, whether via teleoperation, behavior cloning, or reinforcement learning, on a robot hand can be notoriously harsh on hardware, especially when it is placed on a robot arm [109, 217]. Due to the movement of the arm, the hand may repeatedly collide with the table and objects it is trying to grasp. A robot hand should be robust to such treatment and continue to function reliably without breaking. In addition, a robot hand must be able to impart large torques. This is required for lifting heavy objects or using heavy tools like drills or hammers.

While 3D printing is fast and inexpensive, the resulting parts are often not strong enough. One alternative is custom metal machined parts. However, we avoid these as they add

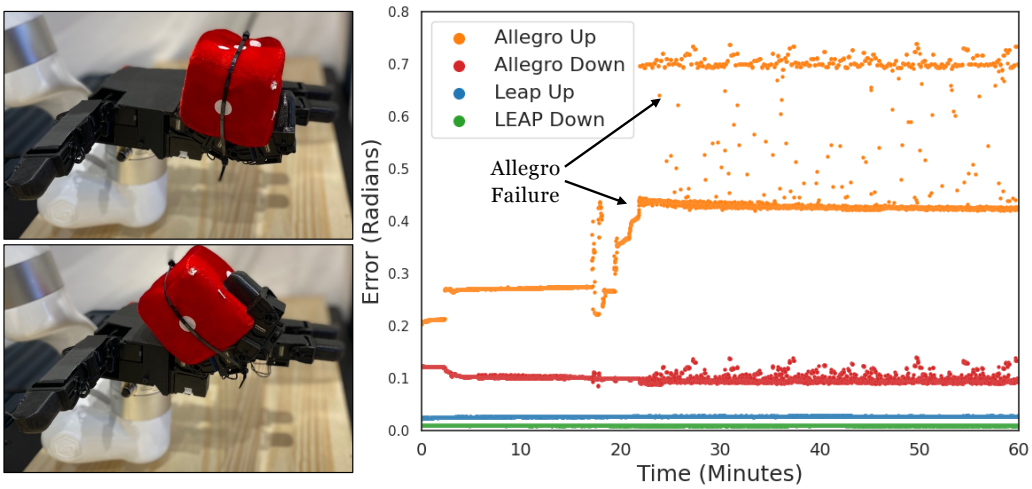

Figure 2.5: **Repeatability test**. *Left:* An illustration of LEAP-C Hand performing repeating grasp-ungrasp on a small plush dice using one joint for an hour. *Right:* Comparison of LEAP-C Hand and Allegro [5]. After just 15 minutes, the Allegro Hand cannot maintain movement. In contrast, LEAP Hand continues to maintain movement with minimal joint error (< 0.05 rad). Videos at https://leap-hand.github.io/

significant cost and require specialized skill and equipment to manufacture. We instead rely on inexpensive ($10) off-the-shelf professionally extruded reinforced plastic brackets from Robotis [23] that are designed to withstand wear and tear. We only print the palm and smaller wire guide spacers using a commercial 3D printer.

The joints in LEAP Hand are designed to exceed the strength of the human hand. We choose motors geared to high torque output for their size while still being capable of a hand-like joint movement velocity of around 8 rad/sec. The amount of motor mass inside the hand is maximized compared to the size of the hand, and every other component is minimized. This enables the hand to be as strong as possible for its human-like form factor. Because these motors are so powerful, we support current- or torque-limiting them as in Section 2.6 to manipulate fragile objects and increase the durability of the hand.

Endurance test

To test the strength of the hand over a long period of time, we hang a 2kg weight on one of the fingertips. This pose is similar to if a person was holding a half gallon of milk up with one finger without using their palm as support. We find that LEAP Hand is able to continuously hold the grasp for an hour with only a small angle error. While the current

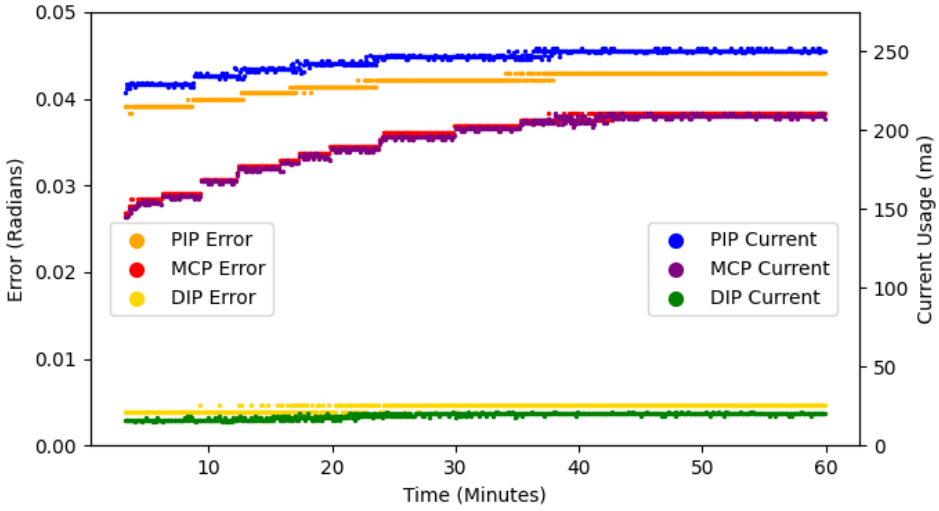

Figure 2.6: **Endurance Test**. We balance a heavy 2kg weight on only one fingertip of LEAP Hand for one hour. On the left axis, we show that the angle error of commanded vs actual remains small. The right axis shows that the current use does initially increase with the temperature of the motor. However, it still withholds the weight and uses less than half of its maximum possible current of 600ma.

usage gradually increases initially, it stabilizes along with the temperature of the motors, which remain cool. The current usage of the top motor reaches 250mA, which is still less than half of the maximum possible. The Allegro Hand is not powerful enough to complete this test. See Figure 2.6 for a plot of the results.

Repeatability test

We test the consistency and accuracy of LEAP Hand against Allegro Hand by running them continuously for 1 hour in a grasping scenario as in Figure 2.5. We continuously raise and lower a small (25g) plush dice strapped onto the finger by commanding one of the base finger joints up and down at 5Hz. The error of the desired joint angle compared to the actual joint angle is graphed through time.

LEAP Hand has a consistent error of 0.025 radians in the up position and 0.005 in the down position (Fig. 2.5), which is reasonable given the PID controller and the 750mA current limit. On the other hand, the Allegro hand starts at a much higher error. After 15 minutes, it begins to fail and then completely fails to move on one out of three grasps. This was not a failure of the position sensor in the motor. The strain of the continuous grasping

Hand	**Strength** (N)	**Power Density** $N \times DOF/(cm^2)$
Bauer et. al [47]	37.4	0.677
Allegro Hand [5]	8.5	0.35
D'Manus [51]	27.8	0.313
Inmoov Hand [16]	5.8	0.116
Adult Human Hand	26.5	2.199
LEAP Hand	19.5	**1.045**
LEAP-C Hand	21.5	**1.15**

Table 2.3: **Pullout Test**. A resistance comparison of each hand to pullout force which correlates to grasping strength. Power density is the total amount of motor force per square area of the hand.

on the motor caused overheating such that the motor was not able to apply required torques.

Pull-out force test

This test measures the amount of momentary outward force that can be resisted by a flexed finger from a hooked force gauge before failure. Failure is defined as a motor or gear slipping or a finger deviating more than 15 degrees from its commanded position (Fig.2.7). The force returned correlates with the grip strength.

Tab. 2.3 compares force for robot and human hands. D'Manus [51] is the strongest due to its large motors. Of the anthropomorphic hands, LEAP Hand performs the best, exceeding the grip strength of a human. The Allegro Hand is weak because of smaller motors explaining why it struggles in many grasping tasks. The tendon-driven hands, Bauer et. al, and Inmoov do not perform that much better in this test even though they can store large motors in their wrists and arms. We find that these tendons often slip and cannot provide that much force at the end-effector.

2.6 Fabrication and Software

Fabrication First, each of the 3D printed components must be fabricated (Fig. 6.1). A $200 Ender 3 3D printer [12] was used with PLA plastic over a 2 day period, but any consumer-grade FDM printer will suffice. Each of the two palm pieces is printed along with fingertips and finger spacers. We collect the 3D printed parts, plastic extruded brackets, the Dynamixel motors [11], U2D2 controller, and assorted cabling. The fingers are assembled

individually using brackets, 3D-printed finger spacers, and motors. The assembly process for LEAP Hand takes around 4 hours. Then each of the fingers is mounted onto the palm, and their firmware is flashed for control. The hand interfaces with the computer using a USB cable and ROS, Python, or C++. The 4-finger LEAP Hand weighs 595g and can be easily mounted to a variety of robot arms. Full video instructions of the assembly process is on our website.

Software A variety of control modes are supported on LEAP Hand: position control, current control, current-based position control, and velocity control. Position control enables the hand to create torques to match a desired position on the motors which is typical of many PID-based controllers. Current control mode enables a desired torque to be applied to the motors. Current-based position control mode enables PID-based position control but also caps the maximum current and torque. This enables the hand to follow position commands but also prevents large torques, which can be unsafe for the robot and the environment around it.

Figure 2.7: **Pullout Test**. A pull-out force is applied and the maximum force is recorded before the hand has a 15 error or slipping.

Simulation We construct a detailed 3D assembly of the hand as used in (Fig. 2.4) on Pybullet. This will enable anyone to 3D print and design their own version of LEAP Hand. In addition to hardware, we release an Isaac Gym and Pybullet-based simulator for LEAP Hand. Its faithfulness to the real world is verified by performing sim2real in Sec. 2.7.4. We release the sim2real platform to jumpstart lab research with LEAP Hand.

2.7 LEAP Hand Applications

First, we compare all of the hands in a grasping test with various everyday objects. Next, we compare the two most robust, human-like hands, the 4-finger LEAP Hand and the Allegro Hand [5] against each other in a variety of machine learning tasks. Teleoperation from human video demonstrates grasping capabilities and human-like form factor. Next, leveraging internet video shows the capability of learning from humans. Finally, we show

Object	Grasp Type [153]	LEAP	LEAP-C	Allegro	D'Manus	Inmoov
Power:						
Mustard	Med. Palm+Pad	20	20	13	8	Y
Toy Kick Ball	Lrg. Palm+Pad	20	20	9	20	N
Golf Ball	Small Pad	16	20	7	0	Y
Softball	Large Pad	20	20	10	15	N
Drill	Trigger Press	20	20	15	0	N
Pringle Can	Power Palm	19	20	20	0	Y
Pan (from rim)	Disk Grasp	20	20	20	14	N
Intermediate:						
Chopsticks	Tripod Grasp	16	13	0	0	N
Wood Cylinder	Cigarette Grasp	4	5	0	0	N
Precision:						
1" Cube	2 Finger Precision	20	20	20	0	N
M&M	Tip Pinch Grasp	Y	Y	Y	N	N
Wine Glass	Flat Hand Cupping	20	20	4	0	N
Credit Card	Lateral Pinch	20	20	8	0	N

Table 2.4: We test each robot hand on a variety of objects and grasps types and see how much perturbation force they can resist (in newtons). The dexterous morphology of LEAP Hand as well as its strong motors enables the cigarette and flat-hand cupping grasps.

LEAP Hand on in-hand manipulation via sim2real, which demonstrates that the simulation and hardware are precise. In this task, LEAP Hand is able to rotate the cube faster and is more robust to disturbances. Please see the supplemental and our website for videos of these results.

2.7.1 Grasping Test using Teleoperation

We compare each of the hands and their ability to perform different types of grasp when holding objects. To quickly experiment and find these poses, we use the Manus Meta VR glove [17] to accurately teleoperate the first three hands (see appendix for details). Since D'Manus is not anthropomorphic enough to teleoperate from human motion, we manually control keyframes for it. We show various types of grasps that each hand can perform, and the amount of perturbation force they can resist (up to 20N). Once the object is grasped we push on it with the force gauge until it slips or the force gauge crosses 20N. Because InMoov is too fragile to teleoperate and apply perturbation forces to, we only test if it can grasp the object securely and report these results in the table.

Table 2.4 shows LEAP Hand can grasp all objects and can perform both many power and precision grasps. While LEAP Hand and LEAP-C Hand perform similarly, the latter has weaker grasps because its MCP side motors cannot be used to adjust the grasp. Allegro's motors are significantly weaker which leads to objects like the golf ball or the soccer ball to easily slip out. Additionally, because its kinematics lacks adduction/abduction in an extended position, it cannot perform the tripod grasp for the chopsticks, the cigarette grasp for the wooden cylinder, or the flat hand cupping grasp for the wine glass properly. D'Manus can complete extremely strong power grasps on larger objects, but its lack of opposability and inability to provide resistive force on all 4 sides of the objects hurts its performance. Due to its large size, it fails to grasp smaller objects.

2.7.2 Teleoperation from Uncalibrated Human Video

Teleoperation enables control of high DOF robots in real-time via human feedback. This is also a useful method for collection demonstrations. Because the Allegro Hand's morphology does not have a human-like MCP joint we must borrow the human-to-robot re-targeting

#	Teleoperated Task	Success Rate		Completion Time (in *s*)	
		LEAP Hand	Allegro	LEAP Hand	Allegro
1	Pickup Dice Toy	**1.0**	0.9	**6.5 (1.7)**	8.6 (2.65)
2	Pickup Dino Doll	**1.0**	0.9	**6.0 (1.5)**	8.2 (3.49)
3	Box Rotation	**0.7**	0.6	**28.2 (15.7)**	37.2 (12.6)
4	Scissor Pickup	0.6	**0.7**	32.4 (7.8)	**28.6 (9.4)**
5	Cup Stack	**0.8**	0.6	**15.4 (7.0)**	21.5 (7.6)
6	Two Cup Stacking	**0.6**	0.3	**18.2 (9.2)**	27.3 (11.0)
7	Pour Cubes in Plate	**0.8**	0.7	**30.2 (15.2)**	36.8 (17.7)
8	Cup Into Plate	**0.8**	**0.8**	**6.2 (2.5)**	10.6 (4.4)
9	Open Drawer	**0.9**	**0.9**	**18.2 (11.2)**	23.6 (12.3)
10	Open Drawer & Pick	**0.7**	0.6	37.2 (10.2)	**33.7 (8.1)**
	Outperform rate	**9/10**	3/10	**8/10**	2/10

Table 2.5: **Teleoperation—comparing LEAP Hand and Allegro.** Success rate and average completion time of a trained operator completing a variety of teleoperated tasks. LEAP Hand outperforms or matches the Allegro performance on 9/10 tasks.

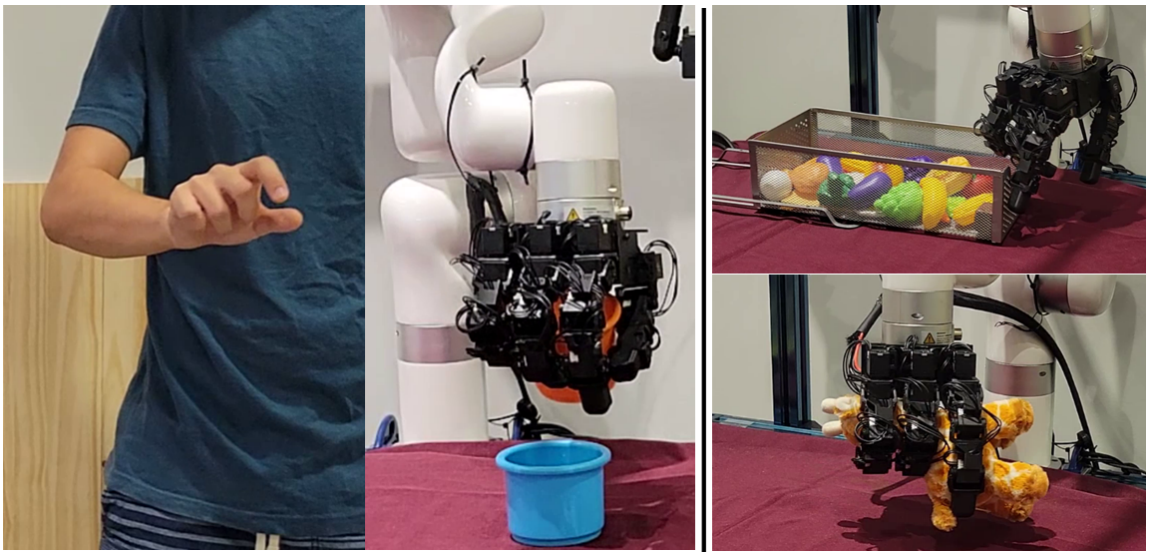

Figure 2.8: **Teleoperation and Behavior Cloning.** *Left*: We perform dexterous teloperation using Telekinesis [216] with a single view color camera. *Right*: We perform behavior cloning from internet video and teeloperated demonstrations using Videodex [212].

method from Robotic Telekinesis [216] (Fig. 2.8 (left)), that manually defines key vectors between palms and fingertips on both robot v_i^r and human hand v_i^h. These vectors define an energy function E_π which minimizes the distance between human hand poses (parameterized by the tuple (β_h, θ_h)) and the robot hand poses q scaled by c_i:

$$E_\pi(\ (\beta_h, \theta_h),\ q\ ) = \sum_{i=1}^{10} ||v_i^h - (c_i \cdot v_i^r)||_2^2 \tag{2.1}$$

[216] trains an MLP $H_R(.)$ to implicitly minimize the energy function described in Equation 5.2.

Because LEAP Hand includes similar joints to a human, we can directly map joint angles between the human and robot. In table 2.5, we observe that LEAP Hand performs better than Allegro Hand on 9/10 of these teleoperated tasks. LEAP Hand is easier to control due to its better morphology, accuracy, and responsiveness to hand input. As users in [216] mention, it is difficult to teleoperate a robot hand with an energy function. Additionally, the better opposability and strength of LEAP Hand allows the operator to reliably grasp objects that are difficult to grasp on the Allegro Hand. While Allegro Hand needed to take breaks to avoid overheating like in [110], LEAP Hand kept running without a degredation

	Pick		Rotate		Open		Cover		Uncover		Place		Push		**Overall**
	train	test	train	test	train	test	train	test	train	test	train	test	train	test	
Allegro Hand	0.81	0.75	**0.89**	0.69	0.90	**0.80**	0.78	0.67	**1.00**	**0.90**	0.90	0.70	**1.00**	**1.00**	6/14
LEAP Hand	**0.92**	**0.84**	**0.89**	**0.72**	**0.94**	0.76	**0.80**	**0.75**	0.96	**0.90**	**0.94**	**0.75**	**1.00**	**1.00**	**12/14**

Table 2.6: **Learning from videos via VideoDex [212].** Hand policies are pretrained on internet videos of humans and finetuned using minimal (≈ 100) teleoperated demos. On this practical use case, LEAP Hand performs better on 12 of 14 $\{\text{task}\}\times\{\text{train, test}\}$ pairs.

of performance.

2.7.3 Behavior Cloning from Demonstrations

Behavior cloning enables agents to learn a policy for a particular task given demonstrations. However, collecting demonstrations for behavior cloning is expensive. We utilize video from Epic-Kitchens [78] as pre-training for our policy using VideoDex [212] along with NDP [42], see Fig. 2.8 (right). We also only use demonstrations collected from prior work [212] on Allegro Hand and map those to LEAP Hand. LEAP Hand still outperforms the Allegro Hand in task performance as in Table 2.6. This is because of its consistency and strength while completing these tasks.

2.7.4 Sim2Real In-Hand Manipulation

We perform in-hand rotation of a cube along an axis perpendicular to the palm. LEAP Hand can return current joint position, velocity, and torque and can be controlled from both torque or position commands. In this case, the robot infers the object pose through the history of observed joint angles alone. This is a challenging task since it is contact-rich, and the policy cannot directly observe the pose of the cube. The policy receives joint angles (16 values) from the motors and outputs the target joint angles (16) at 20 Hz which is passed as position commands to the motors.

We choose a GRU [74] architecture for our policy. We first generate a cache of stable grasps similar to [191]. The policy is then rewarded for turning the cube $r_{\text{rot}} = \text{clip}(\omega_z, -0.25, 0.25)$, where ω_z is the angular velocity along the vertical axis. We add additional penalties for deviation from the stable grasp pose, mechanical work done, motor torques, and object linear velocity. The scale for the rotation reward is 1.25. The scales for

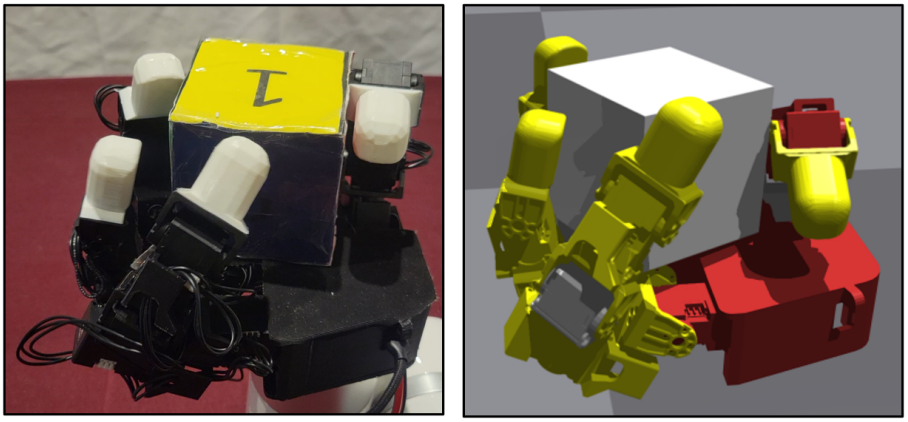

Figure 2.9: **Sim2Real transfer.** *Left*: Simulated LEAP Hand in Isaac Gym [162] completing an in-Hand manipulation task. *Right:* LEAP Hand completing the same task in the real world. Please see our website https://leap-hand.github.io/ for our open source pipeline.

the penalties are -0.1, -1, -0.1, -0.3 respectively. We train PPO [204] with BPPT [232] in IsaacGym [162].

In simulation, we compare the average angular velocity of a cube for different hands and find that LEAP Hand leads to faster rotations (Tab. 2.7) than Allegro. This is because the joint structure of LEAP Hand allows it to support the cube from the sides, whereas since Allegro does not have adduction/abduction it must let go of the cube periodically in order to re-orient it.

Hand	Angular velocity (rad/s)
Allegro	0.0828
LEAP-C Hand	0.2205
LEAP Hand (ours)	**0.2288**

Table 2.7: Comparison of angular velocity for the blind in-hand rotation of a cube in simulation. Since the Allegro Hand lacks abduction/adduction at its extended position, it has low angular velocity. However, LEAP Hand and LEAP-C Hand have this ability and have better performance.

2.8 Conclusion and Future Work

We introduce LEAP Hand and its core design principles. Following these principles, we demonstrate that LEAP Hand can perform exceedingly well compared to other hands on

the market in strength, grasping, and durability. We show its usefulness in a variety of real-world tasks, including teleoperation, behavior cloning, and sim2real. We **open source** the URDF model, 3D CAD files, and a development platform with useful APIs. In future work, we plan to develop and integrate LEAP Hand with low-cost touch sensors.

Chapter 3

DASH: Designing Anthropomorphic Soft Hands through Interaction

3.1 Abstract

Modeling and simulating soft robot hands can aid in design iteration for complex and high degree-of-freedom (DoF) morphologies. This can be further supplemented by iterating on the design based on its performance in real world manipulation tasks. However, iterating in the real world requires an approach that allows us to test new designs quickly at low costs. In this paper, we leverage rapid prototyping of the hand using 3D-printing, and utilize teleoperation to evaluate the hand in real world manipulation tasks. Using this method, we design a 3D-printed 16-DoF dexterous anthropomorphic soft hand (DASH) and iteratively improve its design over five iterations. Rapid prototyping techniques such as 3D-printing allow us to directly evaluate the fabricated hand without modeling it in simulation. We show that the design improves over five design iterations through evaluating the hand's performance in 30 real-world teleoperated manipulation tasks. Testing over 900 demonstrations shows that our final version of DASH can solve 19 of the 30 tasks compared to Allegro, a popular rigid hand in the market, which can only solve 7 tasks. We open-source our CAD models as well as the teleoperated dataset for further study. They are made available on our website https://dash-through-interaction.github.io.

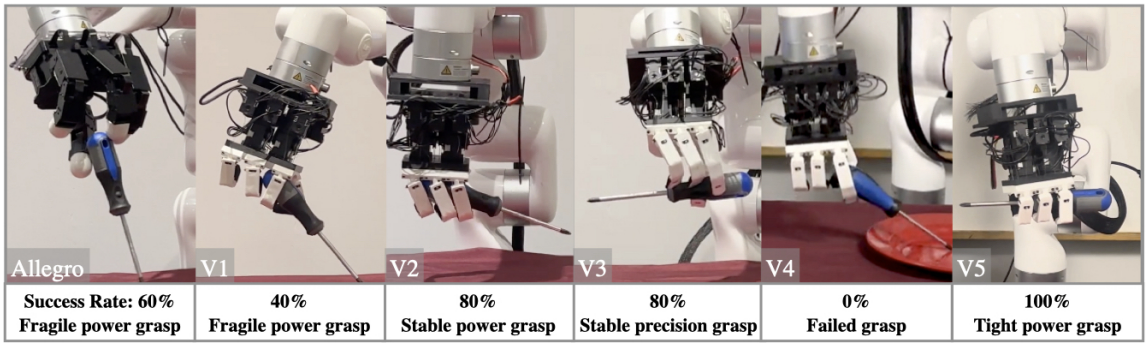

Figure 3.1: Manipulation task performance over five iterations of DASH designed through rapid prototyping and real-world evaluation on tasks alongside task performance of our baseline hand Allegro.

3.2 Introduction

Rapid prototyping technologies have advanced significantly, making way for designers to build new systems at a fast pace. These techniques, such as 3D-printing, allow for quick turnaround between design iterations to test and evaluate systems quickly. This is especially useful for systems with dynamics that are difficult to predict or model, such as soft robot manipulators.

Iterating for dexterous soft hand designs is a laborious process. The complex design space and the infinite degrees of freedom make it difficult to predict the effects of incremental design changes. Unlike rigid robot hands, state-of-the-art soft body simulators are not able to provide accurate, efficient, and robust evaluation of soft designs [69]. Hands such as the BRL/Pisa/IIT SoftHand [87] or the RBO hand [190] have evolved over years to incorporate more adaptive synergies and dexterity. To speed up development times and reduce fabrication overhead many works have recently turned towards 3D-printing to either directly print soft hands [47] or to quickly create complex molds [244]. While this has significantly reduced the cycle time for fabrication, designing dexterous soft hands still requires a lot of expertise, and trial and error due to the continuously deformable nature of soft robots. The lack of appropriate simulators means the evaluation of soft hand designs has to be done on the real prototype by using hand-crafted policies [29] or sequential keyframed open-loop poses [47].

Our key insight is that we can evaluate these systems beyond hand-crafted policies or

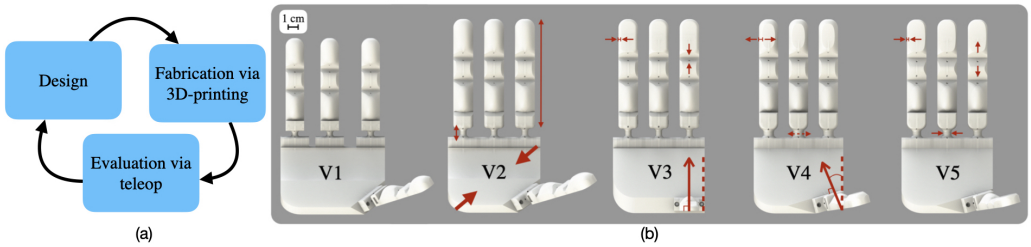

Figure 3.2: (a) Our soft robotic hand design process involving rapid prototyping and real-world evaluation (b) CAD models and differences across DASH iterations V1 through V5, as explained in Section 3.5.

sequential keyframed open-loop poses using recent advancements in teleoperation systems. Improvements in hand tracking and pose estimation [200] have led to the development of vision-based teleoperation approaches including using a single RGB camera for real-time tracking of human hand poses [217]. Teleoperation offers valuable insights into system performance and enables the identification of robust strategies in real-world scenarios. Simulation often falls short in capturing system nuances, accurately modeling soft materials, and adapting strategies. Therefore, tasks that succeed in simulation can still fail when observed and evaluated through teleoperation, providing a clearer understanding of the system's capabilities and limitations.

In this paper, we 3D-print soft robotic hands to test iteratively using teleoperation on a designated manipulation task set, modify the design, and repeat this process shown in Figure 3.2(a). While manually testing and revising designs to improve systems is not a new concept [244], recent technologies allow us to repeatedly iterate the entire framework of designing, fabricating, and evaluating in a matter of *days*. This framework is versatile and can be implemented at any stage of the design process. Its purpose is to bridge the gap between the real world and simulation by adjusting for discrepancies or expediting fine-tuning for real-world scenarios. We envision this framework as a valuable augmentation to existing design frameworks, enhancing the overall design process for soft anthropomorphic robotic hands.

Using our framework, we present a case study to design a 16-DoF tendon-driven 3D-printed soft hand DASH, shown in Figure 3.1. This hand has a small form factor similar in size to a human hand, 3D-printable parts that are easily replaceable, and a modular customizable design that allows for easy iteration. Through teleoperation, we explore the capabilities of the soft hand in order to inform our design iterations across five hands:

DASH v1, v2, v3, v4, and v5. In order to evaluate the dexterity of the hand, we designed a suite of 30 manipulation tasks with varying grasp types and objects that are inspired by human hand capabilities, which allows us to test the capabilities of our robot hand. Our hands show improvements across iterations, albeit not monotonically, and each iteration, except v1, required less than 100 hours to design, fabricate, and test. DASH v1, v2, v3, v4, and v5 succeed on 70%, 82%, 83%, 75%, and 87% of executions across all tasks, respectively. We also outperform a commercial dexterous robotic hand, Allegro [138], which has a success rate of 60% on the same 30 tasks.

The contributions of this paper are

- A detailed report of our process for designing soft hands that leverages rapid prototyping techniques and uses a teleoperated real robot for evaluation, instead of simulation.
- The design of a state-of-the-art dexterous anthropomorphic soft hand using our framework, that outperforms a commercial robotic hand on real world manipulation tasks.
- The release of open-source CAD models and data corresponding to 900 teleoperated human demonstrations to democratize access to low-cost dexterous hands.

3.3 Related Work

Soft robotic hands such as RBO [190] utilize intrinsic compliance, rapid prototyping, and actuated palms for a modular, highly compliant, high degree-of-freedom, and low cost manipulator in order to perform a large variety of in-hand manipulation tasks and object grasps. However, existing robotic hands are still far from achieving human-like dexterity for manipulation [76]. It is necessary to continuously improve and refine the design of both existing and new robotic hands in order to achieve greater dexterity and functionality for performing more complex manipulation tasks.

Although there is currently no unified framework for designing iterations of soft anthropomorphic hands, design iteration methods for robotic systems have been explored. Typically, Finite Element Method (FEM) is used to assess optimal geometries and morphologies before fabricating the final design for real-world evaluation [91, 94]. For instance, the SOFA soft robot simulator has been used to co-optimize control and design of soft

hands [86]. However, these approaches are primarily limited to simulated environments and do not address the sim-to-real gap or real-world design iteration. A related framework in robotic fish design utilizes simulation-based FEM testing, real-world design iteration, and proposes a modular design for easier iterations [244]. Nevertheless, optimizing soft robots is challenging due to their complex geometries, impeding the development of efficient optimization algorithms. Furthermore, the lack of efficient simulation tools that can rapidly evaluate design candidates further compounds the challenge [69].

Learning control policies for dexterous manipulation is challenging due to the high DoFs and complex interactions [36, 173]. In contrast, teleoperation offers a swift and natural way to control robot hands, beyond pick-and-place scenarios. It has been used for human demonstrations in imitation learning [40, 212]. Teleoperation is particularly valuable during the design process of dexterous hands, enabling quick evaluation of nuanced capabilities. Mapping human to robot hand morphology can be categorized as *Joint-to-Joint*, *Point-to-Point*, or *Pose-based* [143]. For our soft hand, we adopt a similar joint-to-joint mapping technique as Liarokapis et al. [147].

Our work extends the design and fabrication methodology presented by Bauer et al. [47], consisting of simplifying the design complexity of soft hands by incorporating geometric features such as bumps or creases to achieve 'joint-like' deformations. They perform kinematic testing on designs before fabricating and evaluate a single design. Similar to Bauer et al. [47], we utilize 3D printing, creases for 'joint-like' deformations, and tendon-driven actuation to curl the soft fingers. In addition to these features, DASH's design incorporates three additional tendons in all fingers, enabling adduction, abduction, and folding of the fingers towards the palm, thereby enhancing dexterity.

3.4 Experiment Setup

3.4.1 Robot Hand design

Finger Joints

All iterations of DASH consist of four fingers: the thumb, index, middle, and ring fingers (see Figure 3.3(a)). In order to achieve modularity, each finger, including the thumb, is designed identically. Each finger has three joints (from the base of the finger to the

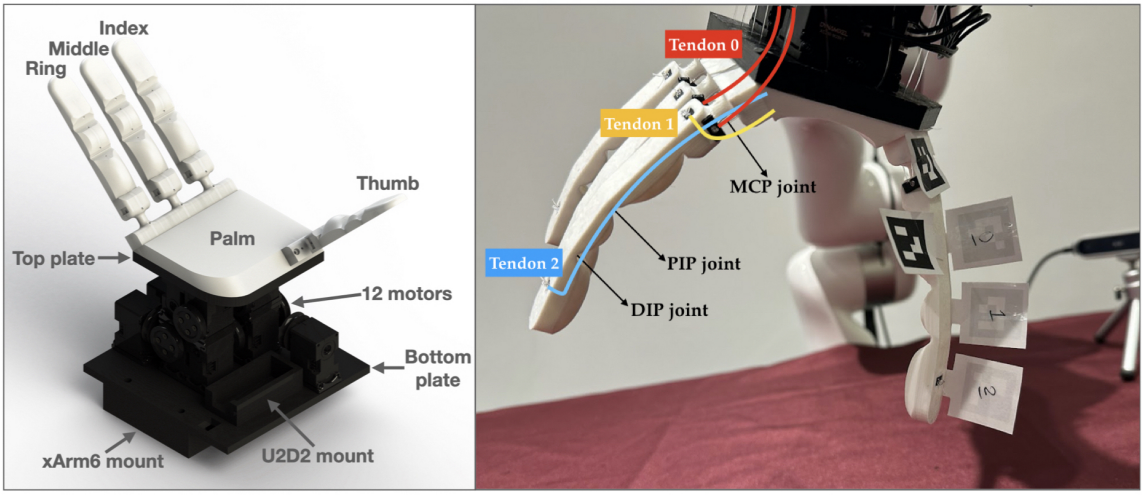

Figure 3.3: (a) Assembly of DASH-v3 (top to bottom) including fingers, palm, top plate, motors, bottom plate, xArm6 mount. (b) Calibration procedure to map motor angles to finger joint positions, where tendon 0 actuates MCP side-to-side, tendon 1 actuates MCP forward folding motion, and tendon 2 curls the finger controlling both PIP and DIP joints.

fingertip): the metacarpophalangeal (MCP) joint, proximal interphalangeal (PIP) joint, and the distal interphalangeal (DIP) joint. The joints for each finger are shown in Figure 3.3(b).

Tendons

Each finger is controlled by four tendons shown in Figure 3.3(b). Two tendons run along the sides of the MCP joint, closest to the palm, for abduction and adduction, which allows the fingers to move closer together and farther apart. These two tendons are controlled by a single motor, so we refer to them as tendon 0. A single tendon, tendon 1, is used to flex the finger forward at the MCP joint, orthogonally to the axis of motion of the abduction-adduction tendons. The last tendon, tendon 2, runs through the entire length of the finger to enable completely curling into itself.

3.4.2 Fabrication using 3D-printing

The hand assembly, shown in Figure 3.3(a), is the same for all hand iterations and consists of 4 soft fingers attached to the soft palm. The rigid components include a top plate below the palm, 12 motors, a bottom plate which also houses the Dynamixel U2D2 motor controller,

and a xArm6 mount.

The soft dexterous hand's mounts, motor housing, and motor pulleys are all 3D-printed from PLA (rigid material depicted in black in Figure 3.3(a)) while the soft hand was printed with Ninjaflex Edge (83A shore hardness) [18] using an Ender 3 S1 Plus. For all the robot experiments, the hand was mounted onto a xArm6 [28] robot arm. DASH costs approximately $1500 to build with the majority of the cost consisting of 3D-printer ($500) and the twelve motors ($1000) required. For comparison, the Allegro hand is also 16-DoF and costs around $15000 [248].

3.4.3 Hand Evaluation using Teleoperation

Learning Kinematics

To approximate the kinematics of the finger joints without real-time feedback, we learn a model from collecting offline paired motor and joint angle data from a single finger. Through small increments of 3 degrees of actuation and joint angle tracking across 1000 finger configurations using AR tags and RGB cameras, we obtain a collection of tuples (joint angles, motor angles) that are normalized to $[0, 1]$ for independent kinematic calibration, assuming fixed joint lengths and bending at the creases.

A linear model is learned using the collected data to map finger joint angles to motor angle outputs. We refer to the motors controlling tendons 0, 1, and 2, as shown in Figure 3.3(b), as motors 0, 1, and 2, respectively. The equations for MCP, PIP, and DIP joint angles are shown below. In equation 3.1, we learn the MCP joint angles $\theta_{\mathsf{MCP}_{\mathsf{side}}}, \theta_{\mathsf{MCP}_{\mathsf{fwd}}}$ jointly since the amount of side-to-side angle at the MCP joint can restrict the forward folding motion of the finger. In equation 3.2, the Motor 2 angle $\theta_{\mathsf{motor}_2}$ is an average measure of the motor angle for the desired PIP and DIP joint angles $\theta_{\mathsf{PIP}}, \theta_{\mathsf{DIP}}$ since the same tendon controls both the PIP and DIP joints. The weights in equations 3.1 and 3.2 are found by fitting our data using linear functions. We collect training data for almost two hours for each iteration of the hand to calibrate new models, using the weights shown in Table 3.1.

$$\begin{bmatrix} \theta_{\mathsf{motor}_0} \\ \theta_{\mathsf{motor}_1} \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} \cdot \begin{bmatrix} \theta_{\mathsf{MCP}_{\mathsf{side}}} \\ \theta_{\mathsf{MCP}_{\mathsf{fwd}}} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{3.1}$$

Hand Design	*v1*	*v2*	*v3*	*v4*	*v5*
w_1	-1.05	-0.43	-0.43	-0.59	-0.59
w_2	0.01	0.2	0.2	-0.12	-0.19
w_3	0.1	0.51	0.51	0.26	-0.32
w_4	0.83	0.54	0.54	0.38	0.72
w_5	0.67	0.6	0.6	0.62	0.63
w_6	0.99	0.76	0.76	1.69	0.65
b_1	0.47	0.38	0.38	0.45	0.58
b_2	-0.07	0.01	0.01	0.44	-0.03
b_3	0.03	-0.04	-0.04	-0.05	-0.09
b_4	-0.01	-0.16	-0.16	-0.3	-0.07

Table 3.1: Calibration weights for all five iterations of DASH mapping from finger joint angles to motor angles.

$$\theta_{\mathsf{motor}_2} = \frac{\theta_{\mathsf{PIP}} w_5 + b_3}{2} + \frac{\theta_{\mathsf{DIP}} w_6 + b_4}{2} \tag{3.2}$$

Teleoperation System

We use Manus Meta Quantum Metagloves [17] designed for VR tracking and Mocap Use, as shown in Figure 3.4 (costs $\sim$ \$8000). The Manus glove is worn on the operator's hand and tracks fingertip positions within a 0.1-degree accuracy using hall effect sensors. Each finger returns 4 angles $\theta_{\mathsf{MCP}_{\mathsf{side}}}, \theta_{\mathsf{MCP}_{\mathsf{fwd}}}, \theta_{\mathsf{PIP}}, \theta_{\mathsf{DIP}}$ in real time, which are mapped one-to-one to the robot hand. Then, we convert to motor angles using our kinematics models.

To control the robot arm shown in Figure 3.4, we employ wearable SteamVR trackers [24], utilizing time-of-flight lasers emitted from SteamVR Lighthouses positioned around and above the operator. One tracker is worn on the glove, while another is placed around the waist. We align the waist tracker's rotation with the robot's base frame and adjust the end-effector pose to match the orientation of the human wrist. Then, the human wrist poses are scaled up to cover the robot's larger workspace, making necessary adjustments to ensure user comfort. Safety checks, including dynamic force feedback on the arm, prevent accidental damage to the robot or its surrounding environment.

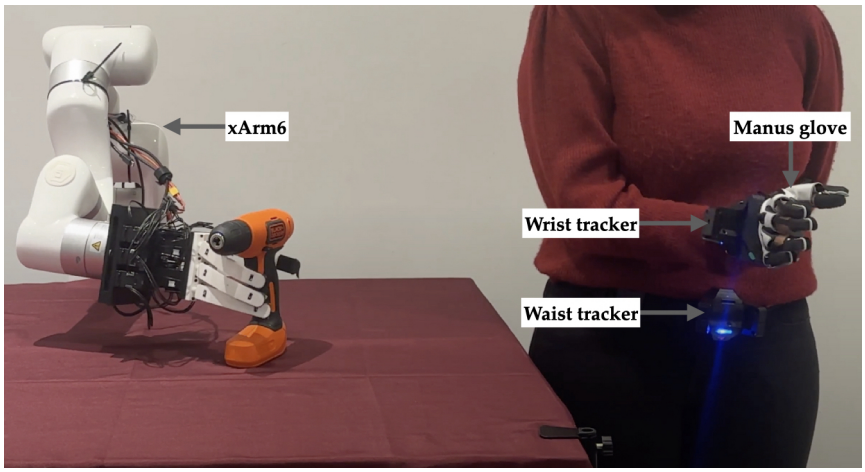

Figure 3.4: Manus Meta Quantum Metagloves used for tracking the hand for teleoperating the robot arm and DASH.

3.4.4 Manipulation Tasks

Each DASH iteration is tested on a suite of tasks, named DASH-30, listed in Table 3.2 which were inspired by the different types of grasps defined in Liu et al. [154] and tasks from previous teleoperation works [107, 217]. These tasks are categorized by the type of grasp or force necessary. Categories like Hold include a greater number of tasks aimed at testing different grasping techniques and objects. On the other hand, skills like Lever or Twist involve fewer tasks specifically designed to assess whether a particular hand design can successfully perform these skills. Additionally, some tasks were hand-picked as tasks where compliance of the hand may be advantageous.

The feedback from the manipulation task evaluation combines observations from the following metrics: task success, performance across five repetitions of the same task, trends in tasks the hand fails to complete, type of grasps possible or used, opposability of fingers, and reachability of fingertips.

Hold
1) Scissor, 2) Hammer, 3) Chopsticks (single), 4) Pen, 5) Wooden cylinder (using adduction/abduction), 6) Screwdriver, 7) Drill, 8) (Plastic) Egg*, 9) (Plastic) Chip*, 10) M&M*
Pick (and place)
11) Dry-Erase Board Eraser, 12) Tennis Ball, 13) Softball, 14) Cloth*, 15) Plush Broccoli, 16) Plush Dinosaur,17) Pringles Can, 18) Spam Box, 19) Mustard Bottle, 20) Wine Glass, 21) Bin picking
Lever
22) Cube flip, 23) Card pickup from deck
Twist
24) Dice rotation in-hand, 25) Grape off of stem*
Open
26) Plastic bag*, 27) Drawer
Put in/on
28) Cup Pouring (onto plate), 29) Cup Stacking & unstacking, 30) 1 inch Block stacking

Table 3.2: DASH-30: task set of 30 manipulation experiments. Tasks with the asterisk (*) were hand-picked as tasks where compliance of the hand may be advantageous.

3.5 DASH Iterative Design Studies

3.5.1 Iteration v1

DASH-v1 performs best on pick and place tasks and tasks using adduction-abduction such as the grape task, shown in Figure 3.5 due to the larger hand size and space between fingers as illustrated in Figure 3.2(b).

Design & Fabrication

We start by designing the shape of the finger to enable curling fully into itself, incorporating four total tendons per finger, and redesigning the MCP joint as a multi-axis flexure for

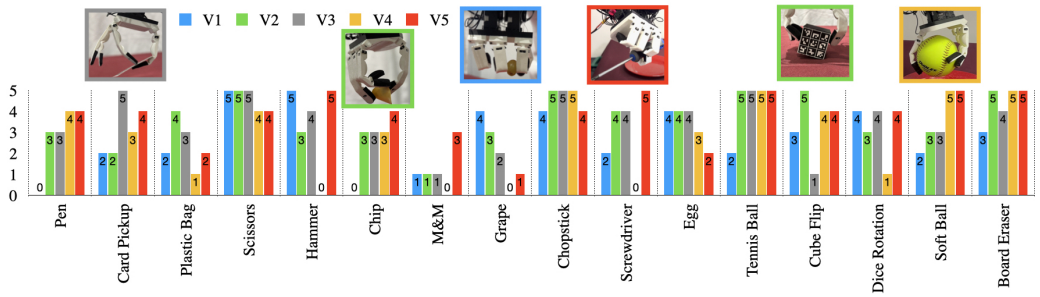

Figure 3.5: Subset of tasks with different performance success across V1 to V5 on specific tasks used to inform design iteration. The top row of inset images shows representative tasks of successful tasks for each hand.

increased dexterity in our desired hand. We iterated on the finger design for approximately four months and use this design for all four fingers in DASH-V1.

Designing V1 included considerations such as tendon anchors, 3D-printing settings, and material stiffness. For example, printing the hand with more infill makes it stiffer but requires more torque than our motors can supply for the joint's full range of motion. To better understand the stiffness of the fingers, we test the finger strength by curling the finger completely and using a force gauge to pull on the finger until it uncurls (see Table 3.3 for results). DASH-V1 hand design is shown on the left in Figure 3.2(b). We designed the full hand assembly for V1 in 1 month.

Evaluation

We test DASH-V1 on the 30 manipulation tasks from Section 3.4.4, repeating each task five times. Over 150 repetitions, DASH-V1 succeeded on all 5 repetitions for 10 of the 30 tasks, as shown in Figure 3.6. For V1, these tasks were Scissors, Hammer, Wooden cylinder, Cloth, Plush Broccoli, Plush Dinosaur, Pringles can, Mustard bottle, Wine glass, and Cup stacking. V1 struggled to grasp small objects such as the M&M, Pen, and Chip since the fingers were not able to reach and properly oppose each other. For tasks involving precise motions such as picking the Grape off a stem and opening the Plastic bag, V1 uses the abduction-adduction capability. The abduction-adduction grip strength of V1 is high and enables picking up objects such as the grape off the stem with ease, as shown in Figure 3.5 inset.

V1 succeeds on five out of five repetitions on 10 tasks but shows room for improvement.

Grasps that require all four fingers such as picking up a tennis ball would be more successful if the thumb could reach and oppose the rest of the fingers. The best opposability to the thumb was to the ring finger, hence pinch (or precision) grasps were easiest to execute with those two fingers. Improving the reachability and opposability of the fingertips requires a smaller palm or longer fingers. We explore these design options in v2 in order to have more overlap in the workspace of the fingers.

3.5.2 Iteration v2

DASH-v2 performance improves on pick and place and hold tasks requiring power grasps. Furthermore, v2 excels at the levering task of cube flip on table, shown in Figure 3.5 inset, due to higher finger strength and fingertip reachability from a smaller palm and longer finger hand design shown in Table 3.3 and Figure 3.2(b).

Design & Fabrication

The second iteration of DASH consists of changes to the size of the hand and the MCP joint of the finger. To allow for more reachability among the fingers as well as opposability, the fingers were made longer and the palm was made smaller as shown in Figure 3.2(b). For comparison, DASH-v2 is similar in size to the average male hand which is 88.9mm wide and 193mm long (wrist to fingertip) [15] . Compared to DASH-v1, there is more than a 25% reduction in area of the palm and the finger length increased by 11% in v2 which is shown in Figure 3.2(b).

The MCP joint was improved to achieve a larger range of motion. The underlying structure of the MCP joint is a cylinder to act as a multi-axis flexure, thus we increase the height of the cylinder to increase the joint angle range for the side-to-side and forward motion of the fingers. The design changes also resulted in a higher maximum load of a single finger as shown under finger strength in Table 3.3. Thus, v2 achieves increased range of side-to-side and forward motion for the fingers by redesigning the MCP joint, and has a larger overlap in the workspace of the fingers solving the reachability and opposability issues in v1.

Designing, printing, and assembling v2 took 5, 83, and 6.5 hours, respectively. Printing v2 required us to not only re-print the soft hand, but also the rigid motor housing as the motor arrangement differs from v1. In total, making v2 from v1 took 94 hours.

Evaluation

With larger range of motion at the MCP joint and better reachability, we expect V2 to achieve better performance on tasks involving smaller objects like M&M, Pen, and Chip. As shown in Figure 3.5, V2 did improve performance on Pen and Chip. M&M and Card pickup were tasks that did not improve from V1. Both of these tasks require fine manipulation which is still a limitation in V2. Instead, our main improvement from V1 to V2 is in achieving better power grasps. Tripod grasps or using more than two fingers was necessary to have stable grasps, especially for the holding tasks such as Hammer, Screwdriver, and Chopstick. However, observations during teleoperation included difficulty using precision grasps with two fingers.

V2 performs better than V1 in 14 tasks (refer to Figure 3.6), including tasks involving Soft ball, Screw driver, Tennis ball, Dry-erase board eraser, and Spam box that all require power grasps. As shown in Figure 3.5, the most significant improvements are seen for Pen, Chip, Tennis ball and Cube Flip. The inset in Figure 3.5 shows V2 grasping Chip with the ring finger and thumb finger, and V2 succeeding at all five repetitions of Cube Flip. These improvements are possible with better reachability and opposability of the thumb with the rest of the fingertips.

Having more space between the fingers made abduction-adduction tasks such as picking Grape off of a stem and Wooden cylinder easier for V1 compared to V2, but V2 still performs reasonably well. Out of the 150 repetitions, V2 is successful in 123 repetitions, which is 18 more when compared to V1. Additionally, the number of tasks where all five repetitions were successful increased from 10 tasks using V1 to 14 tasks using V2.

3.5.3 Iteration V3

DASH-V3 has the best thumb opposability and thinnest fingertip design out of all of our hand iterations, yielding in the best score for Card Pickup as shown in Figure 3.5. Thinner fingertips, however, led to weaker finger strength which decreased task success for tasks such as Dry-erase Board Eraser and Grape off stem.

Hand design	*v1*	*v2*	*v3*	*v4*	*v5*
Palm size	94x102	84x84	84x84	84x84	84x84
Finger length	90	100	100	100	100
MCP diameter	6	6	6	10	8
MCP height	6	8	8	8	8
DIP crease width	10.3	10.3	8.9	10.3	13.0
Thumb angle	45°	45°	0°	22.5°	22.5°
Fingertip edge	3.5	3.5	1.73	3.5	3.5
Fingertip thickness	13.21	13.22	7.98	11.22	8.75
Finger strength	37.8	47.6	34.5	51.8	27.4

Table 3.3: Hand design parameters where finger length refers to the distances in millimeters from the top of the MCP joint to the fingertip and finger strength (N) is measured by pulling on a fully curled finger with a digital force gauge.

Design & Fabrication

The changes from DASH-V2 to V3 involve changing the thumb placement and fingertip shape. In order to make grasps with only two or three fingers more stable, the thumb has to be directly opposable to the rest of the fingers, most importantly the index finger. In V2, the thumb has a 45-degree angle to the palm which we change to be parallel to the index finger in V3, as shown in the middle of Figure 3.2(b) and in Table 3.3. In addition to the thumb placement, the fingertip shape was changed from a rounded surface to a thinner wedge-like surface (see Figure 3.2). The rounded surface in V2 presented a point contact when interacting with objects. In contrast, the wedge-like surface will have a larger contact area and thinner fingertip (similar to fingernail) in order to get under objects to grasp. This results in a thin fingertip edge, almost half the size of V1 and V2's fingertip edge (see Table 3.3). We also move the tendon routing farther away from the center axis of the MCP joint so that we can exert more torque when folding the finger forward about the MCP joint.

Designing, printing, and assembling V3 took 4, 67.25, and 4.25 hours, respectively. Similar to V2, we reprinted the motor housing again due to the new thumb placement. In total, making V3 took almost 83.75 hours.

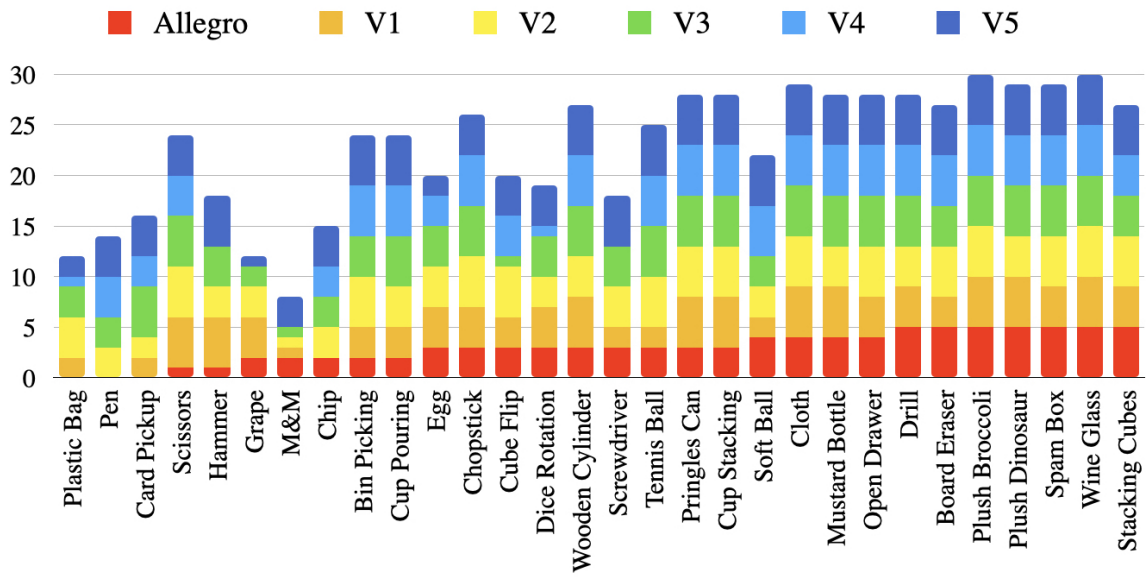

Figure 3.6: Task performance over 5 repetitions of each task across V1, V2, V3, V4, V5, and Allegro as baseline. The tasks are ordered difficult to easy from left to right, according to task performance of Allegro.

Evaluation

As shown in Figure 3.6, DASH-V3 has more successful tasks than the previous hand iterations and our baseline, completing 16 tasks successfully in all repetitions as opposed to the 14 tasks V2 successfully executed. V3 succeeded on all repetitions of Wooden Cylinder, Card Pickup, Cup Pouring, Drill, Plush Dinosaur, and Mustard Bottle, which are tasks V2 did not master. The task improvement was due to better thumb opposability compared to V2. In total, V3 succeeded on 124 repetitions which is 1 more than the number of repetitions V2 is successful at. With V3, we observe higher grasp stability during power grasps and handling of delicate objects, during teleoperation. Additionally, we find that the fingertip shape makes a large difference for specific tasks. We clearly see this effect occurring in Cube flip and Card pickup (see inset images of V2 Cube Flip and V3 Card Pickup in Figure 3.5). The flat fingertips of V3 are ideal for thin delicate card pickup but not for the cube flip. Reorienting the cube in-hand in Cube flip is better suited to the rounded fingertip on V2, keeping a stable point contact while the object rotates on the table.

3.5.4 Iteration v4

DASH-v4 was optimized for strength as we found that lacking for tasks such as Cube Flip for v3. This allowed for heavy objects like Soft Ball to have great success with v4 as shown in Figure 3.5 but decreased finger folding motion resulted in decreased performance for tasks such as Hammer, Screwdriver, and M&M.

Design & Fabrication

The fourth iteration of DASH was designed to optimize for strength. We focused on redesigning the MCP joint to be thicker, providing increased stiffness for folding the fingers into the palm. Achieving the right balance was challenging, as we aimed to maintain the range of motion for MCP forward motion within the torque limits of our motors. While a simple solution would be to use larger motors to increase force and stiffness at the MCP joint, this would result in a larger and heavier hand.

Additionally, we made changes to the fingertips and thumb placement, creating a hybrid design influenced by DASH-v2 and DASH-v3. Thicker fingertips proved useful for tasks involving rotation, such as Cube flipping, while thinner fingertips were beneficial for pinch grasps like Card pickup. The result was rounded edges with a flat surface in the center of the fingertip, providing versatility for pinch to power grasps. Similarly, the thumb placement was positioned between v2 at $45°$ and v3 at $0°$, settling at $22.5°$ relative to the palm. While v2 excelled in power grasps and v3 in pinch grasps like Card pickup, we aimed for v4 to perform equally well in both types of grasping.

Designing, printing, and assembling v4 took 8.5, 82, and 5 hours, respectively. Similar to v3, we reprinted the motor housing to accommodate the new thumb placement.

Evaluation

DASH-v4 successfully completed all five repetitions of 17 tasks, surpassing the task performance of v3. v4 maintained its performance in most of these tasks, with the exception of Scissors, as shown in Figure 3.5. However, it outperformed v3 in tasks involving the Dry-erase board eraser and performed better than any previous hand iteration in the Soft ball task. This was attributed to the stronger MCP joint, which enhanced the finger strength, as indicated in Table 3.3. Nevertheless, the limited range of motion in

the MCP forward joint resulted in poor reachability, causing objects like Scissors to slip between the fingertips.

Overall, the hybrid thumb position and fingertip shape, combining features from V2 and V3, proved advantageous in achieving a greater number of tasks. However, the next iteration should address the loss of range of forward folding motion to improve reachability. The limited reachability of V4 also led to zero successes out of five repetitions in four tasks, including Hammer, Screwdriver, M&M, and Grape off stem. All of these limitations can be attributed to the restricted range of motion in the MCP forward joint.

3.5.5 Iteration V5

DASH-V5 aimed to be a combination of all previous hand design features with respect to joint and fingertip thicknesses. V5 generally outperformed all previous design iterations and excelled at the Screwdriver task as shown in Figure 3.1.

Design & Fabrication

The fifth iteration of DASH features a stiffer MCP joint compared to V3, but it is more compliant than the MCP joint of V4. By increasing the compliance at the MCP joint, we were able to achieve a greater range of motion at the joint compared to V4, which had limited folding capabilities. Furthermore, we made the fingertip thinner than that of V4, and widened the DIP crease (as shown in Table 3.3), in order to improve the curling of the finger. As a result, DASH-V5 exhibits the most extensive curling motion among all the previous iterations.

Designing, printing, and assembling V5 took 2, 24, and 2.75 hours, respectively. Unlike the previous versions, we kept the motor assembly unchanged and only replaced the fingers of V4. Consequently, the total time required for iteration was the lowest for V5, totaling 28.75 hours.

Evaluation

Among all the design iterations of DASH, DASH-V5 performed the best. V5 succeeded on five out of five repetitions on 19 tasks and achieved a completion rate of 131 out of 150 total task repetitions. In addition to the tasks that V4 succeeded on, V5 also completed five out of five repetitions on the Hammer and Stacking cubes tasks. This improvement

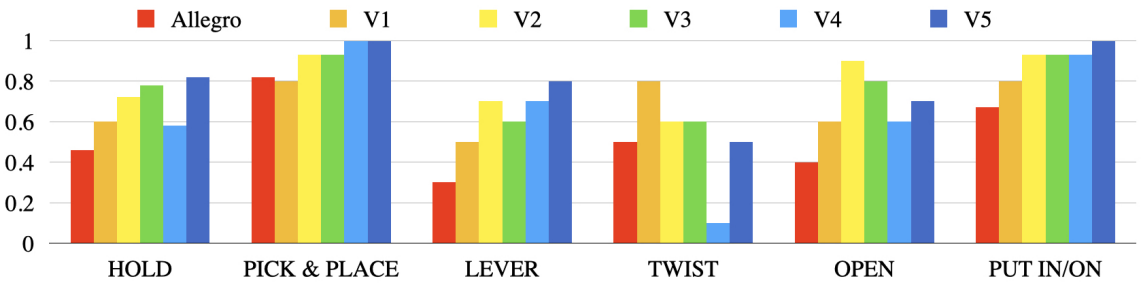

Figure 3.7: Task performance across V1, V2, V3, V4, V5, and Allegro as baseline on each category of tasks from Table 3.2.

indicates that we have made incremental progress on the hand design. V5 had the most curling range of motion than previous hands which made picking objects easier for the teleoperating user due to stable grasps enveloping objects into the palm.

As shown in Figure 3.5, V5 showed improved task performance for Hammer, Screwdriver, Chip, M&M, Grape off stem, and Plastic Bag. However, it performed worse for the Chopsticks and Egg tasks. Although V5 has the lowest finger strength among all DASH iterations due to thinner joints and thinner fingertips (as shown in Table 3.3), its enhanced finger curling abilities even enabled a single finger to hold objects. However, when completely curled, thin objects such as the chopsticks were prone to falling between the thumb and fingers. This issue could be addressed by introducing longer fingers to allow for more overlap between the fingertips. Overall, V5 outperformed all previous iterations of DASH across all 30 tasks. However, it is worth noting that certain hands may specialize in specific tasks. For instance, V5 excelled at picking up Screwdriver, while V3 was the most suitable for Card pickup, as shown in Figure 3.5. One interesting result involved the V5 screwdriver hold, which aligned perfectly in the groove on the tool handle.

3.6 Baseline Study: Allegro Dexterous Hand

Allegro [138] is an off-the-shelf gripper that we use as a baseline. Allegro is a dexterous robotic hand that has four fingers with motors at the joints, rigid structure, and large rubber spherical fingertips. We perform the same 30 manipulation tasks from DASH-30 (Table 3.2) with Allegro to compare the performance against all iterations of DASH.

Allegro succeeded on all five repetitions on 7 out of 30 tasks. These tasks included manipulating the Drill, Dry-Erase board eraser, Plush broccoli, Plush Dinosaur, Spam box,

Wine glass, and Stacking cubes, as shown as the rightmost tasks in Figure 3.6. Allegro performed best on pick and place tasks compared to other types of tasks as shown in Figure 3.7. However, all iterations of DASH, except V1, were also successful at these tasks.

The Allegro robotic hand and fingers had difficulty with tasks such as picking up the Pen, Card, Plastic bag, Scissors, and Hammer which required precision.While both DASH and Allegro hands lack sensing capabilities, this disproportionately affected Allegro because lack of compliance made it easy to grasp too tightly or not enough, especially for rigid objects. Similarly, the Cup pouring grasp was unstable due to the spherical fingertips rotating the cup in-hand during the task. The side-to-side motion (or abduction-adduction) of the fingers was limited, making Dice Rotation coarse and unpredictable. However, Allegro had stable grasps for larger and softer objects such as the Drill, Softball, Plush Dinosaur, Plush Broccoli, and Wine Glass (see rightmost tasks in Figure 3.6).

3.7 Discussion

Across the 30 tasks, we observe that V5 has the best performance solving 19 tasks successfully completing all repetitions, while V4, V3, V2, V1, and Allegro solve 17, 16, 14, 10, and 7 tasks, respectively. In Figure 3.7, we see that all iterations of DASH outperform the Allegro baseline on most categories of tasks listed in Table 3.2 as well as steady improvement in DASH iterations except for twisting and opening which are the most difficult categories of tasks. The two twisting tasks were Dice Rotation and Grape which both more successful with the larger palm and space between fingers for V1 compared to other iterations. For opening tasks, V2 had more success on opening Plastic Bag due to its rounder fingertips and higher finger strength. Through our suite of varied manipulation tasks and human-in-the-loop design iteration, we validate our framework's ability to use real world evaluation to iteratively design soft robot hands through rapid prototyping and teleoperation.

From our case study in Section 3.5, we draw three crucial observations regarding our proposed framework. Firstly, the direct feedback from the designer performing real world manipulation tasks with DASH was crucial for us in informing the design changes required to improve performance across iterations. In contrast, testing in simulation can result in design changes that do not necessarily translate to performance improvement in the real

world. Secondly, using teleoperation removed the necessity of designing different control policies for 30 various tasks across six robot hand morphologies in our case study, and allowed us to adjust grasps in real-time during task execution, which is often not feasible in simulation or by using keyframed poses. Lastly, despite using real robot hands in the design iteration process, our framework has a short iteration time, consisting mostly of printing time (about 80% of total time), by leveraging 3D-printing and the use of teleoperation to evaluate the design in the real world.

Our framework can extend to testing other soft robotic hands in the real world for rapid design iteration. There are three stages of our framework, as shown in Figure 3.2(a), including design, fabrication, and evaluation. Some best practices include incorporating a modular design to facilitate easier iteration, adopting rapid prototyping methods for seamless fabrication, and favoring incremental design changes to allow for targeted iteration on specific design features. Our method of evaluating using teleoperation also allowed for minimal changes in control when the hand design changed. Our framework can be used to test easily prototyped hands, such as those by Bauer et al. [47] or RBO [190], using the same setup used for DASH iteration, similar to our Manus [17] VR teleoperation system. Additionally, DASH-30, our suite of 30 varied manipulation tasks can be used to benchmark other dexterous hands in the community.

Observing that our robot hand has similar structure and size to human hands, we note a crucial limitation of our framework, shown in Figure 3.2(a), for robot hand morphologies that diverge from human hand morphology as teleoperation might not be feasible in such cases. Additionally, calibration or mapping of the teleoperator's hand to the robot hand can have a significant impact on the robot hand's performance in real-world manipulation tasks. For example, an inaccurate mapping from the teleoperator's hand to the robot hand can incorrectly evaluate the robot hand to be incapable of some tasks. Another limitation for this framework is that it can result in longer turnover times for designs that cannot be made with rapid prototyping techniques such as 3D-printing. Lastly, monotonic improvement is difficult to guarantee due to the manual design iteration process in our framework.

3.8 Conclusion and Future Work

This paper presents a design iteration process that can supplement existing design iteration techniques by leveraging 3D-printing and teleoperation. We exhibit the potential of this

framework through a case study of designing a 16-DoF 3D-printed dexterous anthropomorphic soft hand DASH. By 3D-printing the new design at each iteration, and evaluating it on real-world manipulation tasks using teleoperation to inform future hand designs, we consistently improve its performance over the baseline Allegro hand and across successive iterations of DASH. We open-sourced our DASH CAD models and teleoperated demonstration data at https://dash-through-interaction.github.io.

Future directions include automatic design iteration by singling out features of the CAD design and correlating them with capabilities of the hand. Further study would be required to automate this process and use collected data to learn what properties of the hand should be improved for better task performance. Currently, the process of design iteration in our case study was manual in that we chose parameters to change based on task performance and observations from real-world manipulation experiments.

Part II

Learning Human-like Dexterity

Chapter 4

Robotic Telekinesis: Learning a Robotic Hand Imitator by Watching Humans on YouTube

4.1 Abstract

We build a system that enables any human to control a robot hand and arm, simply by demonstrating motions with their own hand. The robot observes the human operator via a *single RGB camera* and imitates their actions *in real-time*. Human hands and robot hands differ in shape, size, and joint structure, and performing this translation from a single uncalibrated camera is a highly underconstrained problem. Moreover, the retargeted trajectories must effectively execute tasks on a physical robot, which requires them to be temporally smooth and free of self-collisions. Our key insight is that while paired human-robot correspondence data is expensive to collect, the internet contains a massive corpus of rich and diverse human hand videos. We leverage this data to train a system that understands human hands and retargets a human video stream into a robot hand-arm trajectory that is smooth, swift, safe, and semantically similar to the guiding demonstration. We demonstrate that it enables previously untrained people to teleoperate a robot on various dexterous manipulation tasks. Our low-cost, glove-free, marker-free remote teleoperation system makes robot teaching more accessible and we hope that it can aid robots in learning to act autonomously in the real world. Video demos can be found at: https://robotic-telekinesis.github.io

Figure 4.1: Our system leverages passive data from the internet to enable robotic real-time imitation in-the-wild. This low-cost system does not require any special gloves, mocap markers or even camera calibration and works from a single RGB camera.

4.2 Introduction

Mimicking human behavior with robots has been a central component of robotics research for decades. This paradigm, known as teleoperation, has successfully been used to enable robots to perform tasks that were unsafe or impossible for humans to perform, such as handling nuclear materials [228] or deactivating explosives [85]. Teleoperation has also been used to enable the robotic automation of tasks that are easy for humans to demonstrate but difficult to program. In industrial robotics, for example, teleoperation can be used to demonstrate a single trajectory (e.g. picking a box from a conveyor belt) that the robot *overfits* to and repeats verbatim for months or years thereafter. Teleoperation can alternatively be used as a means to collect a large dataset of demonstrations, which can then be used to learn a policy that *generalizes* to new tasks in unseen environments [195, 196].

In this paper, we specifically study the problem of teleoperation for dexterous robotic manipulation. While there are many promising current techniques (e.g. Kinesthetic Control [52], Virtual Reality devices [19, 26], haptic gloves [1] and MoCap [245]), each of them

[1]Equal Contributions

suffers from some shortcoming that has limited its applicability. These setups typically involve expensive hardware and specialized engineering, expert operators, or an apparatus that impedes the natural fluid motion of the demonstrator's hand.

The challenge of overcoming these shortcomings grows exponentially with the complexity of the robot to be controlled; multi-fingered hands are far more difficult to teleoperate than two-finger grippers. Despite recent advancements, building an easy-to-use, performant and low-cost teleoperation system for high-dimensional dexterous manipulation has remained elusive. Handa *et al.* recently proposed DexPilot [108], a low-cost system for vision-based teleoperation that is free of markers or hand-held devices. It lowers the cost and usability barrier, but relies on a custom setup with multiple calibrated depth cameras, and uses neural networks trained on images collected in this controlled environment, which limits its use to a specific lab setting.

The objective of this paper is to enable teleoperation of a dexterous robotic hand, *in the wild*. This means our system should be low-cost, work for any untrained operator, in any environment, with only a single uncalibrated color camera. One should be able to simply look into a monocular camera of their phone or tablet and control the robot without relying on any bulky motion capture or multi-camera rigs for accurate 3D estimation. We call our system *Robotic Telekinesis*, as it provides a human the ability to control a dexterous robot from a distance without any physical interaction.

Unfortunately, building such a system poses a chicken-and-egg problem: to train a teleoperation system that can work in the wild, we need a rich and diverse dataset of paired human-robot pose correspondences, but to collect this kind of data, we need an in-the-wild teleoperation system. However, while we lack paired human-robot data, there is no shortage of rich human data, and **our key insight is to leverage a massive unlabeled corpus of internet human videos** at training time. These videos capture many different people from different viewpoints doing different tasks in several environments, ensuring generalization by design.

We propose a method that conquers the human-to-robot problem using two subsystems. The first subsystem uses powerful computer vision algorithms trained to estimate 3D human poses from 2D images, and the second subsystem uses a novel motion-retargeting algorithm to generate a physically plausible robot hand-arm action that is consistent with a given human pose. During training, our method only uses *passive data* readily available online and does not require any *active* fine-tuning on our robot in our lab setup.

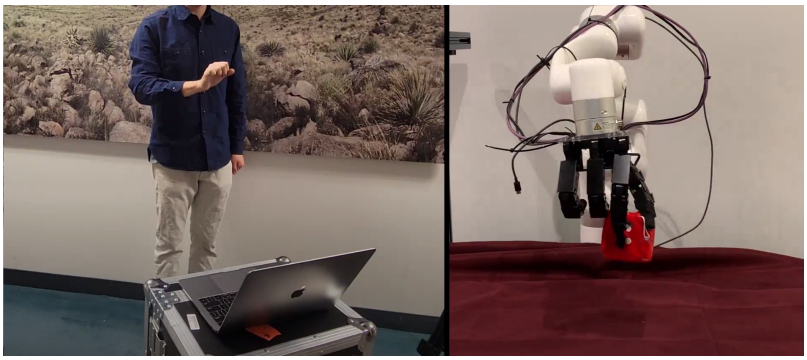

Figure 4.2: An operator completing a dice pickup task while watching the robot through a video conference. Video demos are at https://robotic-telekinesis.github.io/.

Our system is low-cost, glove-free, and marker-free, and it requires only a single uncalibrated color camera with which to view the operator. It allows any operator to control a four-finger 16 Degree-of-Freedom (DoF) Allegro hand, mounted on a robotic arm, simply by moving their own hand and arm, as illustrated in Figure 6.1. We demonstrate the usability and versatility of our system on ten challenging dexterous manipulation tasks. We further demonstrate the generality and robustness of our system by performing a systematic study across ten previously untrained human operators.

4.3 Related Work

The first section reviews relevant research in 3D human pose estimation and the second section discusses related work in kinematic motion retargeting, with a particular focus on cross-embodiment retargeting and teleoperation.

Understanding Human Hands and Bodies Modeling human bodies and estimating their poses are widely studied problems, with applications in graphics, virtual reality and robotics. The recent research most relevant to our work can roughly be divided into four sub-areas. (1) *Hand and body modeling.* MANO [199] is a low-dimensional parametric model of a human hand, and SMPL [158] is an analogous model for the human body. SMPL-X [182] is a single unified model of both the body and hands. (2) *Monocular human hand and body pose estimation*. Recent works in human pose estimation typically estimate 2D quantities like bounding boxes [207] or skeletons [62], or perform a full 3D reconstruction [95, 123, 230]. Rong *et al.* [200] propose a method for integrated 3D reconstruction of human hands and bodies. (3) *Dataset curation*: The advances in

human pose estimation crucially rely on large datasets of human hand and body poses. FreiHand [249], Human3.6M [118] and the CMU Mocap Database [9] are examples of densely-annotated datasets in clean indoor settings. On the other end of the spectrum, the 100 Days of Hands [207] and Epic Kitchens [78] datasets are massive collections of raw videos that span a rich and diverse set of hand poses and motions but don't contain pose annotations. (4) *Understanding human hand function*. Brahmbhatt *et al.* [55] use thermal imaging to capture impact heatmaps that reveal patterns in the ways human hands interact with everyday objects. Taheri *et al.* [223] study the problem of how human hands and bodies behave while grasping and manipulating objects, and propose a method to generate plausible human grasps for novel objects. Hasson *et al.* [111] and Cao *et al.* [63] propose methods for joint hand and object reconstruction, and Hampali et al. [105] present a dataset of hand-object interactions with 3D annotations. Liu *et al.* [152] builds a taxonomy of human grasps to understand the cognitive patterns of human hand behavior [149].

Kinematic Retargeting and Visual Teleoperation Human pose estimation only solves half of the visual teleoperation problem. Mapping human poses to robot poses is itself a difficult challenge, because humans and robots have very different kinematic structures. Li *et al* [144] train a deep network to map human hand depth images to joint angles in the robotic Shadow Hand, and Antotsiou *et al* [37] combine inverse kinematics and Particle Swarm Optimization to retarget human hand poses to a high-dimensional robot hand model. Our system follows the method of DexPilot [108], which minimizes a cost function that captures the functional similarity between a human and a robot hand.

The general problem of kinematically retargeting motion in one morphology into another is also studied outside of robotic manipulation. Villegas *et al.* [229] propose a cycle consistency objective to transform motion between animated humanoid characters of different body shapes. Peng *et al* [184] use an approach based on keypoint matching to learn robotic locomotion behaviors from demonstrations of walking dogs. Zakka *et al.* [243] learn a visual reward function that allows reinforcement learning agents to learn from demonstrators with different embodiments.

4.4 Robotic Telekinesis

Robotic Telekinesis is a system for real-time, remote visual teleoperation of a dexterous robotic hand and arm. A human demonstrates tasks to the robot just by moving their hand,

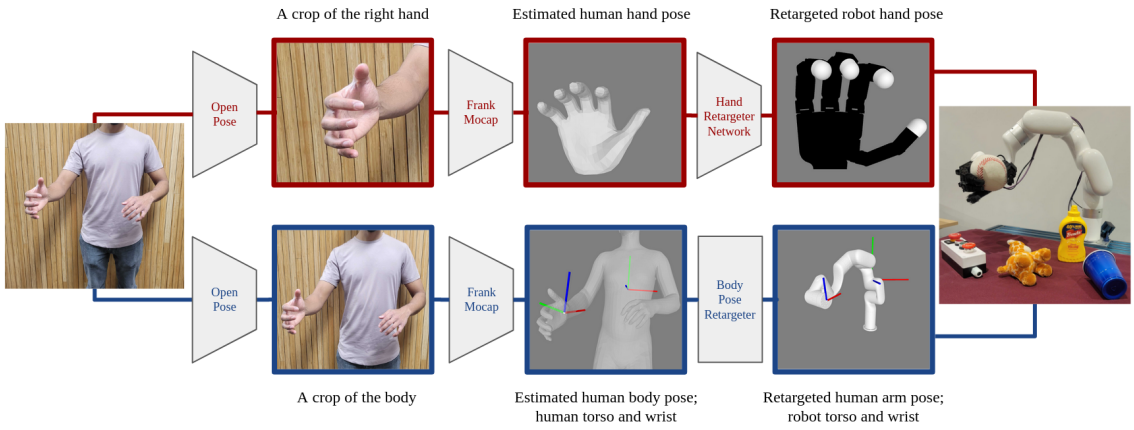

Figure 4.3: A graphical description of our visual teleoperation pipeline. First, a color camera captures an image of the operator. Top: to command the robot hand, a crop of the operator's hand is passed to a hand pose estimator, and the hand retargeting network maps the estimated human hand pose to a robot hand pose. Bottom: to command the robot arm, a crop of the operator's body is passed to a body pose estimator and cross-body correspondences are used to determine the desired pose of the robot's end-effector from the estimated human body pose. Finally, commands are sent to both the robot hand and arm.

and the robot mimics the actions instantaneously. Our system consists of an xArm6 robot arm, a 16-DoF Allegro robot hand, and a single uncalibrated RGB camera capturing a stream of images of the human operator. The operator must be in the field of view of the camera and must be able to see the robot to guide it, either in real life or through a video conference feed. Each image is *retargeted* into two commands which place the robot hand and arm in poses that match the hand-arm poses of the human operator in real time. Figure 4.2 illustrates an operator solving a grasping task while monitoring the robot through a video conference feed.

The problem of remote teleoperation from a single camera is severely under-constrained for two reasons. One reason is that the input images are in 2D while the robot is controlled in 3D: mapping from 2D to 3D is an ill-defined problem. While this issue could be addressed with a multi-camera setup, in this work our goal is to build a system usable with any one uncalibrated camera, from a cell-phone camera to a cheap webcam. The use of a single color-only camera leads to certain failure modes, typically related to inter-hand occlusions and ambiguous depth perception, but these are issues we attempt to mitigate using our neural network based retargeter.

The second reason is the ambiguity caused by the differences in morphology, shape and

functionality, between human hands and robot hands. To address both of these problems, we rely on deep neural networks to learn priors from passively-collected internet-scale human datasets to enable powerful human pose estimation and human-to-robot transfer.

In the rest of this section, we describe our visual teleoperation pipeline. As shown in Figure 5.1, we group the pipeline into two branches: one branch for hand retargeting and the other one for arm retargeting.

4.4.1 Hand Teleoperation: Human Hand to Robot Hand Pose

The problem of retargeting 2D human images to robot hand control commands is broken into two sub-problems. The first is to estimate the 3D pose of the human hand from a 2D image, and the second is to map the extracted 3D human hand parameters to robot joint control commands. We discuss each of these two sub-problems below.

2D Hand Image to 3D Human Hand Pose

The first step in hand retargeting is to detect the operator's hand in a 2D image and infer its 3D pose. To this end, we exploit recent advances in computer vision. We rely on large paired 2D-3D datasets, and high-quality models that leverage this data to produce physically plausible 3D human pose estimates from 2D images.

Our method first computes a "crop" around the operator's hand, based on a bounding box computed using an off-the-shelf detector derived from OpenPose [62]. The resulting image crop goes to a pose estimator from FrankMocap [200] to obtain hand shape and pose parameters of a 3D MANO model [199] of the operator's right hand. See the "OpenPose" and "FrankMocap" modules in the top row of Figure 5.1 for a graphical depiction of this phase of the pipeline, and see the appendix for further implementation details.

We emphasize that our human hand pose estimation module works for *any human operator*, with *any camera* in *any environment*, without any further fine-tuning. This strong generalization is due to the diversity of millions of images on which the neural network and pose estimators are trained.

3D Human Hand to Robot Hand Pose

Next, estimated 3D human hand poses are retargeted to the 16 Allegro hand joint angles

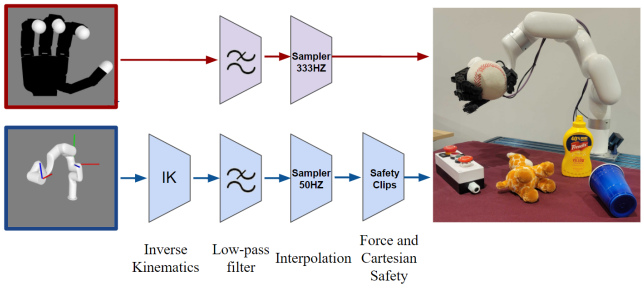

Figure 4.4: **Control Pipeline**. Description of our control stack. Raw target poses are received from the visual retargeting modules, then Inverse kinematics, low-pass filtering, sampling, and safety clipping are performed. The smoothed commands are sent to the robot.

to place it in an analogous hand pose (see the third panel on the top branch of Figure 5.1). This has three challenges:

- Underconstrained: The Allegro hand and the human hand have many DOF and very different embodiments: they differ greatly in shape, size and joint structure.
- Generality: Our retargeter must work for *any human operator* trying to perform *any kind of task* in *any environment*.
- Efficiency: We require a real-time solution (>15 Hz).

A natural way to address these three challenges would be to train a supervised learning model on a diverse dataset of paired human-robot hand pose examples. However, collecting this large scale dataset would be prohibitively expensive. Instead, we train a deep human-to-robot hand retargeter network that uses human data alone.

Dataset of YouTube Videos of Human Interaction We leverage a massive internet-scale dataset of human hand images and videos. We gather about 20 million images from the Epic Kitchens [78] dataset, which captures ego-centric videos of humans performing daily household tasks, and the 100 Days of Hands [207] dataset, which is a collection of YouTube videos. We run the hand pose estimator from [200] (the same one we use at deployment time in our pipeline) to estimate human hand poses for each image frame in these videos. We augment this massive noisy dataset of estimated human hand poses with the small and clean FreiHand dataset [249], which contains ground-truth human hand poses for a diverse collection of realistic hand configurations.

A Lack of Paired Data Our dataset contains millions of human hand poses, but no ground-truth target robot poses to regress onto. In most neural network regimes, at training

time, the network has access to *paired* examples $(x \in \mathcal{X}, y \in \mathcal{Y})$, where $\mathcal{X}$ is the source domain, and $\mathcal{Y}$ is the target domain. In our case, the source domain $\mathcal{X}$ is the set of all human hand poses and the target domain $\mathcal{Y}$ is the set of all robot hand poses, *but we only have training data from the source domain.* Hence, we can not perform a direct regression.

Energy Function Formulation Instead, we formulate the retargeting problem using a feasibility objective. We posit that the optimal corresponding robot hand pose is the one that best mimics the *functional intent* of the human. In order for the robot hand to effectively mimic human actions, the relative positions between the robot's fingertips should match those of the human's. Following [108], we define a set of five hand keypoints (4 fingertips (no pinky) and a palm), and ten keyvectors which connect all pairs of keypoints.

These keyvectors are used in an *energy function* that captures *dis*similarity between a human hand pose (parameterized by the MANO model parameters (β_h, θ_h)) and an Allegro hand pose (parameterized by the joint angles q_a). First, for each $i \in \{1, \ldots, 10\}$, the i-th keyvector is computed on the human hand (call it $\mathbf{v_i^h}$) and the Allegro hand (call it $\mathbf{v_i^a}$). Then, each Allegro hand keyvector $\mathbf{v_i^a}$ is scaled by a constant c_i. The i-th term in the energy function is then the Euclidean difference between $\mathbf{v_i^h}$ and $c_i \cdot \mathbf{v_i^a}$:

$$E(\ (\beta_h, \theta_h),\ q_a\ ) = \sum_{i=1}^{10} ||\mathbf{v_i^h} - (c_i \cdot \mathbf{v_i^a})||_2^2 \tag{4.1}$$

where the scaling constants $\{c_i\}$ are hyperparameters. Critically, this energy function is a fully differentiable function of the Allegro joint angles (because of the differentiability of the forward kinematics operation), which allows us to train the hand retargeter network f via gradient descent, using the energy function as a loss function.

This energy optimization formulation is different than the prototypical IK problem, which solves for joint angles that achieve target fingertip poses relative to a fixed base. Our end-effector constraints are all relative to each other, which makes it difficult to adopt open-source IK solvers such as [64].

Retargeter Network Our hand retargeter network is a Multi-Layer Perceptron (MLP), $f(.)$, with two hidden layers. It takes as input a human hand pose (a vector $x \in \mathbb{R}^{55}$ that denotes the MANO hand shape and pose parameters) and outputs a vector of Allegro joint angles $y \in \mathbb{R}^{16}$. Since there is no paired labels available for human to robot hand, our network is trained to minimize the energy function $E(x, y)$ in Equation 5.2 that captures

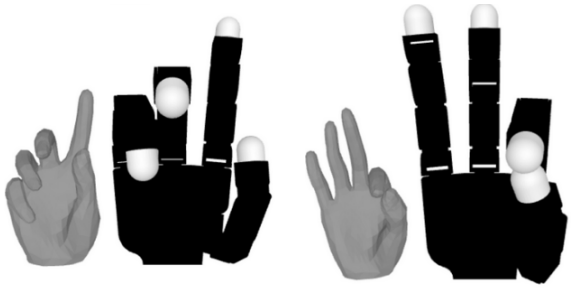

Figure 4.5: **Human-to-robot Translations**. The inputs and outputs of our hand retargeting network. Each of the pairs depicts a human hand pose, and the retargeted Allegro hand pose.

the dissimilarity between the input 3D human hand pose x and the network's predicted robot hand pose y. Per convention, a high energy means the two poses are highly *dis*similar. Formally, the neural network optimizes the following objective:

$$\underset{f}{\operatorname{argmin}}\ \mathbb{E}_{x\in\mathcal{X}}\left[E(x, f(x))\right], \tag{4.2}$$

At inference time, we simply pass the estimated hand shape/pose vector to the network, which directly outputs Allegro joint angles that we can command to the robot. A key benefit of using a neural network is speed: the network's forward pass takes about 3ms (333Hz) – this is critical for smooth real-time teleoperation.

Collision Avoidance via Adversarial Training

Using a neural network to perform human-to-robot hand retargeting has another subtler advantage over an online optimization approach: we can augment the energy function with terms that are slow to compute. When training a neural network, we can run expensive operations in order to compute the loss at each iteration. This allows us to use any energy function, as long as it is differentiable. We simply absorb the computation cost during training instead of incurring it at deployment time.

We exploit this idea to address the problem of self-collisions. Minimizing the keyvector-similarity energy function described above can sometimes yield robot hand joint configurations which orient the hand such that fingers collide with each other or with the palm. It is difficult to add a term to the energy function that penalizes such configurations, since "self-collision-ness" is not a differentiable function of the robot's joint angles.

To address this, we first train a classifier that takes in an Allegro joint angle vector, and

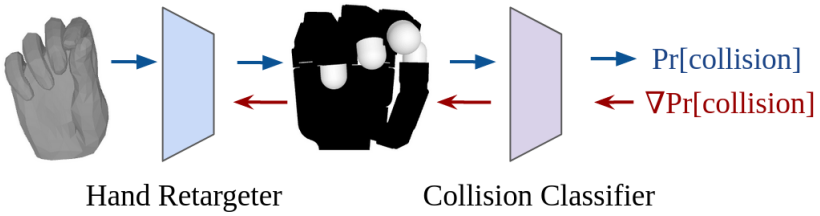

Figure 4.6: A trained self-collision classifier is used as an adversary that penalizes self-colliding joint configurations. The blue arrows denote the forward pass, and the red arrows denote the flow of gradients during the backward pass.

tries to classify whether or not the joint configuration yields a self-collision. This classifier is an MLP, and we generate its training data programatically by repeatedly sampling a joint angle vector within the legal joint limits, and querying a (non-differentiable) self-collision checker to generate a ground-truth binary self-collision label.

Once our self-collision classifier is trained, we use it as a "discriminator" to train our retargeter network. At every training iteration, we pass the retargeter's predicted robot joint angle vectors to the self-collision classifier. Intuitively, we want the predicted self-collision score to be as low as possible, and therefore we use it as a term in the loss function for the retargeter network. *The gradient of the self-collision score from the collision network is backpropagated through the self-collision classifier, and used to update the weights of the retargeter network*, as shown in Figure 4.6. This leads the retargeter network to avoid outputting Allegro joint angle configurations that the self-collision classifier believes to be illegal. Our retargeter network and self-collision classifier are akin to the generator and discriminator respectively in a Generative Adversarial Network (GAN), though in our case, we pretrain and freeze the self-collision classifier so we don't suffer the instability of jointly optimizing a discriminator and a generator, notorious in GAN training.

4.4.2 Arm Teleoperation: Human Body to Robot Arm Poses

A hand that can flex its fingers but does not have the mobility of an arm will not be able to solve many useful tasks. The second branch of our retargeting pipeline therefore focuses on computing the correct pose for the robot arm from images of the human operator.

At each timestep, we first compute a crop of the operator's body using a bounding box detector derived from OpenPose [62], then pass the crop to the body pose estimator from FrankMocap [200]. We model the human body using the parametric SMPL-X model, and

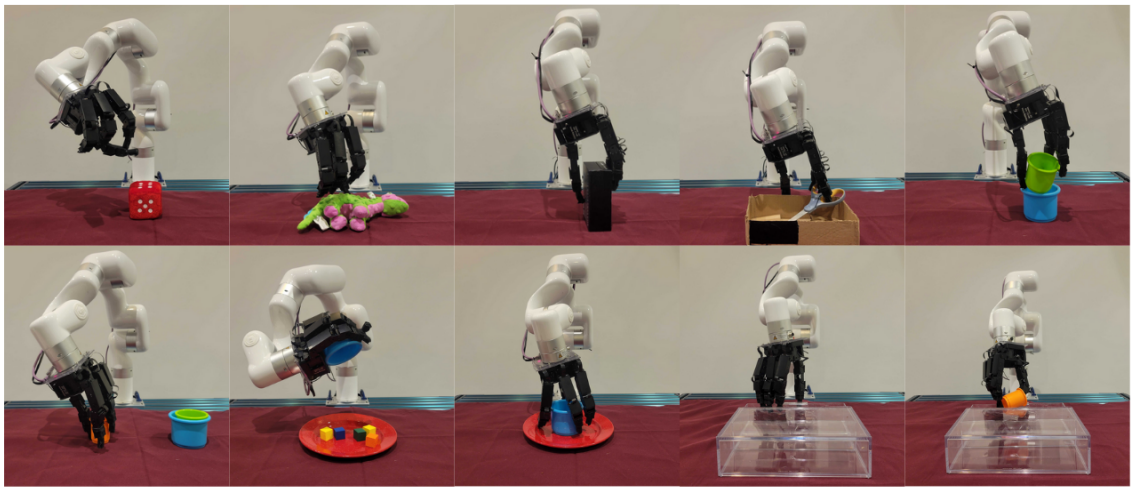

Figure 4.7: Ten different teleoperation tasks. Top row, left to right: Pickup Dice Toy, Pickup Dinosaur Doll, Box Rotation, Scissor Pickup, Cup Stack. Bottom row, left to right: two cup stacking, pouring cubes onto plate, cup into plate, open drawer and open drawer and pickup cup. Please see videos at https://robotic-telekinesis.github.io/.

the body pose estimator predicts the 3D positions of the joints on the human kinematic chain.

Because we aim to build a system that operates from a single "floating" color camera, there are two main problems that arise. (1) Without a depth sensor or camera intrinsics, we cannot accurately estimate how far from the camera the human's hand is. (2) Without camera-to-robot calibration, there is no known transformation between the camera, robot and human. (With a calibrated depth camera, we would simply mount a camera with a fixed known transformation relative to the robot's base frame, localize the position and orientation of the operator's wrist in the 3D camera coordinate frame, and use the known camera extrinsics to determine how the robot's wrist should be positioned and oriented.)

Instead, we *estimate the relative transformation between the human wrist and an anchor point on the human's body*. We define the human's torso as the origin, and manually choose a suitable point to serve as the "robot's torso". We posit that the relative transformation between the human's right hand wrist and torso should be the same as the relative transformation between the robot's wrist link and the robot's torso. By traversing the kinematic chain from the torso joint to the right hand wrist joint, we compute the relative position and orientation between the human's right hand wrist and torso (see Figure 5.1). The bottom

Task	Success (rate) Ours	Success (rate) DexPilot-Mono*	Completion Time (sec) Ours	Completion Time (sec) DexPilot-Mono*	Description
Pickup Dice Toy	**0.9**	0.7	**8.6 (2.65)**	13.5 (5.47)	Pickup Plush dice from table.
Pickup Dinosaur Doll	**0.9**	0.6	**8.2 (3.49)**	11.00 (3.95)	Pickup Plush dinosaur from table.
Box Rotation	**0.6**	0.3	37.2 (12.6)	**16.33 (10.69)**	Rotate box 90 degrees onto the smaller side
Scissor Pickup	**0.7**	0.5	28.6 (9.4)	**27.66 (11.09)**	Remove Scissors from the box using fingers
Cup Stack	0.6	**0.7**	**21.5 (7.6)**	22.85 (16.57)	Smaller cup must be placed inside the large cup.
Two Cup Stacking	**0.3**	0.1	**27.3 (11.0)**	45.00 (0.0)	Small cup placed into medium cup into large cup.
Pouring Cubes onto Plate	**0.7**	0.5	36.80 (17.7)	**13.8 (4.02)**	Five cubes in a cup must be poured onto a plate.
Cup Into Plate	**0.8**	0.7	**10.6 (4.4)**	13.71 (5.44)	Place cup on the plate.
Open Drawer	**0.9**	**0.9**	23.6 (12.3)	**14.88 (4.40)**	Open clear drawer.
Open Drawer and Pickup Cup	0.6	**0.7**	33.7 (8.1)	**28.14 (11.48)**	Open clear drawer and pickup cup inside.

Table 4.1: Success rate and completion time (mean and standard deviation) of a trained operator completing a variety of tasks using two different methods. The `DexPilot-Monocular*` baseline is nearly identical to our system, but uses online gradient descent for hand pose retargeting (inspired by DexPilot [108]). Our system, which uses a neural network retargeter, outperforms the baseline in 7 out of 10 tasks.

row in Figure 5.1 depicts this pipeline visually. This simple correspondence trick works surprisingly well in practice and provides a natural user experience for even a moving operator.

To handle minor errors in human body pose estimation and ensure smooth motion, we reject outliers, and apply a low-pass filter on the stream of estimated wrist poses. We then use an IK solver [59] to compute arm joint angles that place the robot's end-effector (i.e. "wrist") at the correct relative transformation relative to the "robot torso" coordinate frame. See the bottom row of figure 4.4 for a depiction of our control stack, and see the appendix for further details about the arm retargeting modules.

4.5 Experimental Results

We evaluate the strengths and limitations of our system through experiments on a diverse suite of dexterous manipulation tasks with an expert operator. We also demonstrate the usability and robustness of the system through a smaller set of tasks on a group of ten previously untrained operators. Videos can be found at https://robotic-telekinesis.github.io/.

Baseline Our hand retargeter neural network is compared to an *online optimization* procedure that runs online gradient descent to minimize the energy function between the human and robot hand. We call this baseline `DexPilot-Monocular*`: the use of online optimization for retargeting is modeled after DexPilot [108], but the overall system

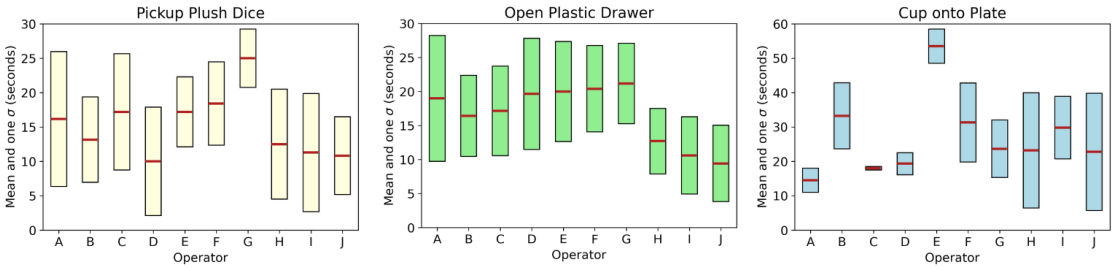

Figure 4.8: Ten novice operators were asked to complete tasks: (1) picking up a plush dice, (2) opening a plastic drawer, and (3) placing a cup onto a plate. For each task, the mean and standard deviation completion times were computed over seven trials.

(including the single-camera setup) is held constant between the baseline and our method. At each timestep, given an estimated human hand pose x, a solver iteratively searches for the robot pose y^* that minimizes the energy (cost) function $\mathcal{L}$ with respect to x, i.e.

$$y^* = \underset{y}{\operatorname{argmin}}\ \mathcal{L}(x, y). \tag{4.3}$$

The code for DexPilot's [108] kinematic retargeting module is not available, so we implement their online optimization solver using the Jax GPU-accelerated auto-differentiation engine [54].

We do not compare our system to the full DexPilot system. DexPilot is designed for use in a specific multi-camera rig, but our system is designed to run anywhere. The `DexPilot-Monocular`* baseline is meant to enable analysis of the tradeoffs between online optimization and neural networks for kinematic retargeting, within a single-camera setup. It uses the retargeting module from DexPilot [108], but is otherwise identical to our system.

4.5.1 Success Rate: Trained Operator Study

A trained operator attempted a diverse set of tasks to test the capabilities of our system and the `DexPilot-Monocular`* baseline. These tasks are shown in Figure 5.5. They span a diverse spectrum of arm and hand motions and involved interacting with a variety of different objects. Each of the ten tasks was run for ten trials with a timeout period of one minute. This rigorously tested the system's capabilities and limitations. These tasks

Pipeline Stage	Ours (Hz)
Open Pose Body (input from camera)	29
Open Pose Hand (input from camera)	29
Frank Mocap Body	16
Frank Mocap Hand	27
Body Pose Retargeter (output to robot)	16
Hand Retargeter (output to robot)	24

Table 4.2: Runtime of each stage of our pipeline. Our hand retargeter NN runs at 24 Hz (the online gradient-descent baseline runs at 10Hz). Both systems use an AMD 3960x CPU and two 3080 Ti GPU's.

are described in Table 4.1. The operator achieved good success on all tasks – our system outperformed the baseline on 7 out of 10 tasks, and performed similarly on the other 3 tasks. Grasping plush objects proved easy as these grasps do not require much precision, but we observed that fine-grained grasps of smaller, more slippery objects like plastic cups occasionally proved difficult. See videos of the trained operator completing these tasks at: https://robotic-telekinesis.github.io/. During experiments, the expert found that our system was easier to use and performed better than `DexPilot-Monocular`*. The online gradient descent solver in the baseline occasionally stayed stuck in local minima because it would use the previous pose as a seed. This meant that the hand would often output unnatural poses with the fingers digging into the palm, an issue that the authors of DexPilot also noted. Our method, because it was trained on YouTube data, learned to always output natural hand poses which was useful for operators to use. Since it is not seeded, our method did not get stuck in minima. This data also masked the ambiguities and errors from our single camera constrained setup. Our method also produced occasional errors on uncommon hand poses unseen in the training set, but these one-off errors did not propagate forward through time. Additionally, the baseline ran at a slower rate and felt delayed to the operator's movements as benchmarked in table 4.2. This was jarring and hard to compensate for when trying to complete dexterous tasks. Our system maintained fluidity and felt very responsive when opening and closing the hand.

4.5.2 Usability: Human-Subject Study

To test usability and generality, we conducted a human-subject study in which 10 previously untrained operators each completed a set of 3 tasks, 7 times each. The first task was a plush

dice pickup task (30 second timeout), the second was drawer opening (30 second timeout), and last was to place a cup onto a plate (60 second timeout). The total time for one human subject to learn about the system and complete all tasks took approximately 15 minutes. Figure 4.8 reports the completion times of each operator on each of the three tasks.

Although the underlying technology is complex, the user interface was easy to understand and use for all operators. Each operator differed in their style of motion, stances, and appearances, but there were no noticeable discrepancies in the behavior of the system or the distribution of results.

We found that subjects often struggled during the first few trials. However, all subjects found it easy to adjust and learn how to use the system very quickly. Our system was often complimented on its responsiveness and fluidity: subjects did not notice with any lag or jitter in the robot's imitation. Subjects enjoyed participating in the study, and some said that teleoperation of the robot was similar to a video-game. Additionally, subjects noted they felt safe and comfortable during teleoperation.

The largest frustrations with the system was in periodic errors in the retargeting of the human fingers to the Allegro robot hand. Many subjects noted instances when they were attempting complicated hand poses, but our system failed to accurately imitate them. In particular, we noticed systematic errors of our system in handling the flexion of the thumb. The shape and joint axes of the Allegro hand thumb are particularly different from that of the human thumb, and we suspect that our energy function does not place enough weight on accurate thumb retargeting. Some subjects observed that the system was worse at tracking their hand when it was all the way open with their palm parallel to the camera, this is a particular issue that we cannot get around with a single camera setup.

4.6 Analysis

4.6.1 Accuracy of retargeter network

We compare the accuracy of `DexPilot-Monocular*`'s *online optimization* with our neural network retargeter that relies on *offline optimization* during training. We gather a test set of 500 sequences from the DexYCB video dataset [67], which contains videos with annotated ground-truth human hand poses. For each video, at each timestep, the poses are fed to both our neural network and `DexPilot-Monocular*` with a (generous) time

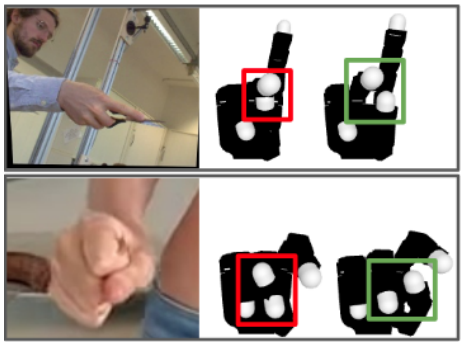

Figure 4.9: **The contribution of an adversarial self-collision loss**. The red boxes highlight instances where the vanilla retargeting network outputs Allegro hand poses that result in self-collision. The green boxes depict the predictions of the network trained with self-collision loss. These robot hand poses maintain functional similarity to the human's hand pose, but avoid self-collision.

budget of 40ms to solve. We emphasize that both retargeters optimize the same energy function, but in different ways.

We do not, however, have access to "ground-truth" Allegro joint angles against which to compare the output of the two retargeters. To circumvent this, we design a version of `DexPilot-Monocular`* that is allowed an *infinite time budget* to run until convergence. We call this the pseudo-ground-truth oracle, and our assumption is that its final output is as close to optimal as possible.

We compare the root mean squared error (RMSE) between the oracle's outputs and the outputs of each of our two retargeters on the dataset. Our neural network retargeter outperforms `DexPilot-Monocular`* in matching the oracle. The neural network retargeter achieves an RMSE of **0.17** radians (about 10 degrees) while `DexPilot-Monocular`* achieves an RMSE of **0.25** radians per joint (about 14 degrees).

4.6.2 Self-Collision Avoidance

We perform an ablation on the weight of the self-collision classifier (Section 4.4.1) in the energy function, to see how it affects the behavior of the hand retargeter. We use a test set of 3000 held-out hand poses from the FreiHand dataset [249] and consider 6 different hand retargeter networks, trained with collision-loss weights of 0, 0.2, 0.4, 0.6, 0.8 and 1. (A weight of 0.8, for example, means that the self-collision loss is weighed 0.8 as heavily as the sum of all the other key-vector matching loss terms in the energy function.) Each

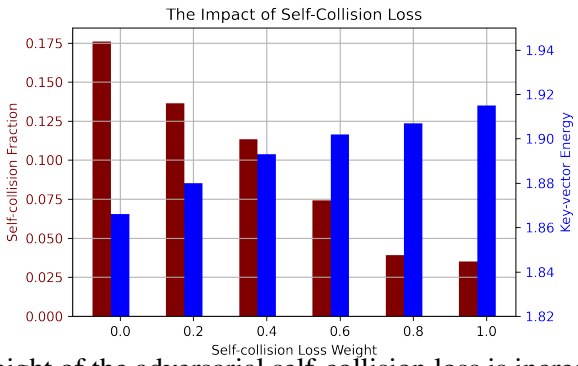

Figure 4.10: As the weight of the adversarial self-collision loss is increased, the hand retargeter network produces fewer self-colliding joint configurations (maroon), but incurs a higher energy with less similar poses (blue). A higher energy means that the predicted robot hand pose is dissimilar to the operator's hand pose.

network makes predictions on the data and we compute (1) the fraction of resulting Allegro joint angle vectors that result in self-collision, and (2) the average value of the key-vector energy terms over the dataset.

We summarize the results in Figure 4.10. The plot shows there is a trade off between minimizing self-collisions, and minimizing key-vector dissimilarity. As we increase the weighting term of the self-collision avoidance loss term in the energy function, we produce fewer offending joint configurations but minimization performance degrades for the other terms in the energy function. We depict this trade off visually in Figure 4.9. It is difficult to confidently assert that one is more valuable than the other, and in practice, we find that a middle ground works very effectively for the user.

4.7 Conclusion

We present *Robotic Telekinesis*, a system for in-the-wild, real-time, remote visual teleoperation of a dexterous robotic hand and arm, in which a human operator demonstrates tasks to the robot just by moving their own hands. We leverage the latest advancements in 3D human pose estimation and thousands of hours of raw day-to-day human footage on the internet to train a system that can understand human motion, and retarget it to corresponding robot actions. Our method requires only a single color camera, and can be used out-of-the box by any operator on any task, without any actively collected robot training data. We show that our system enables experts and novices alike to successfully perform a number of different dexterous manipulation tasks. We hope that our system is used as a starting

point for future research, rather than as an end product. We believe a powerful use of visual teleoperation is to bootstrap autonomous robot learning, and by building an intuitive and low-cost platform for humans to provide task demonstrations, we hope to contribute to the democratization of robot learning.

Chapter 5

VideoDex: Learning Dexterity from Internet Videos

5.1 Abstract

To build general robotic agents that can operate in many environments, it is often imperative for the robot to collect experience in the real world. However, this is often not feasible due to safety, time, and hardware restrictions. We thus propose leveraging the next best thing as real-world experience: internet videos of humans using their hands. Visual priors, such as visual features, are often learned from videos, but we believe that more information from videos can be utilized as a stronger prior. We build a learning algorithm, VideoDex, that leverages *visual*, *action*, and *physical* priors from human video datasets to guide robot behavior. These actions and physical priors in the neural network dictate the typical human behavior for a particular robot task. We test our approach on a robot arm and dexterous hand-based system and show strong results on various manipulation tasks, outperforming various state-of-the-art methods. For videos and supplemental material visit our website at https://video-dex.github.io

5.2 Introduction

The long-standing dream of many roboticists is to see robots autonomously perform diverse tasks in diverse environments. To build a robot that can operate anywhere, many methods rely on successful robotic interaction data to train on. However, deploying inexperienced,

real-world robots to collect experience may require constant supervision which is infeasible. This poses a chicken-and-egg problem for robot learning because to collect experience safely, the robot already needs to be experienced. How do we get around this deadlock?

Fortunately, there is plenty of real-world human interaction videos on the internet. This data can potentially help bootstrap robot learning by side-stepping the data collection-training loop. This insight of leveraging human videos to aid robotics is not new and has seen immense attention from the community at large [78, 101, 102]. However, most of the prior work tends to use human data as a mechanism for pretraining just the visual representation [176, 178, 186, 206, 234], much like how deep learning has been used as a pretraining tool in related areas of computer vision [73, 113] and natural language processing [57, 89]. Although pretraining visual representations can aid in efficiency, we believe that a large part of the inefficiency stems from very large action spaces. For continuous control, learning this is exponential in the number of actions and timesteps, and even more difficult for high degree-of-freedom robots (shown in Figure 5.1). Dexterous hands are one such class of high degree of freedom robots that have the possibility to provide great contact for the grasping and manipulation of different objects. Their similarity to human hands makes learning from human video advantageous.

In this work, we study how to go beyond using internet human videos merely as a source of visual pretraining (i.e. **visual priors**), and leverage the information of how humans move their limbs to guide train robots on how they should move (i.e. **action priors**). However, guiding robot motions using human videos requires understanding the scene in 3D, figuring out human intent, and transferring from human to robot embodiment. First, 3D human estimation works decently well in general human videos which we can leverage to gather 3D understanding. Second, there have been large-scale datasets that break down the human intent via crowdsourcing labels [78, 102]. Finally, to handle the embodiment transfer, we use human hand to robot hand retargeting as an energy function to pretrain the robot action policy. Our key insight is to combine these visual and action priors from human videos with a prior on how robot should move in the world [43, 44] (i.e., **physical prior**, using a second order dynamical system) to obtain dexterous robot policies that can act in the real world. We call this approach, VideoDex. To enhance real-world performance, we mix the experience obtained from massive internet data with a few in-domain demonstrations.

In summary, VideoDex is a robot learning algorithm that incorporates visual, action, and physical priors into a single open-loop policy by learning from passive videos contained

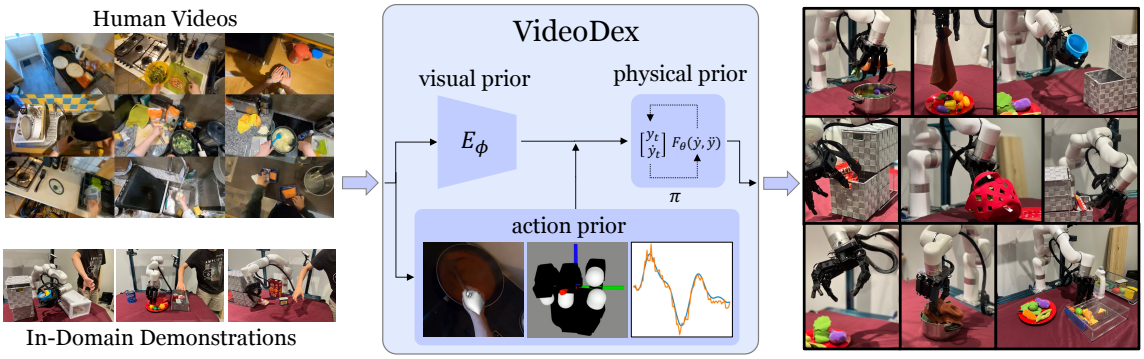

Figure 5.1: We re-target human videos as an action prior, use pretrainined embeddings as a visual prior, and use Neural Dynamical Policies (NDPs) [43] as a physical prior to complete many different tasks on a robotic hand.

in human activity datasets from the internet. VideoDex then only needs to adapt to real world tasks using a few in-domain examples. We find that VideoDex outperforms many state-of-the-art robot learning methods on seven different real-world manipulation tasks on a high DOF multi-fingered robotic arm-hand system as well as on a 1-DOf gripper robotic arm system.

5.3 Related Work

Learning for Dexterity Reinforcement learning (RL) with an engineered reward function can show dexterous simulation results [121, 141] but requires lots of data, especially in high DOF dexterous manipulation. This requires simulators [162, 226], which cannot model physics properly, making real-world transfer difficult. Behavior cloning is an approach [53, 187] that can work safely. DIME [39] involves using nearest neighbor matching of image representations with demonstrations to determine actions. Qin et al. [193] teleoperates and learns policies in simulation, followed by Sim2Real transfer. DexMV[192] uses collected human hand videos for robot hand imitation learning. DexVIP [164] learns hand-object affordances and priors for RL initialization using curated video datasets.

Learning from Videos and Large-Scale Datasets There are many curated datasets from internet human videos, for example, FreiHand [249] for hand poses, 100 Days of Hands [207] for hand-object interactions, Something-Something [101] for semantically similar interactions, Human3.6M [118] and the CMU Mocap Database [9] for Human pose estimation. Epic Kitchens [78], ActivityNet datasets [92], or YouCook [81] are

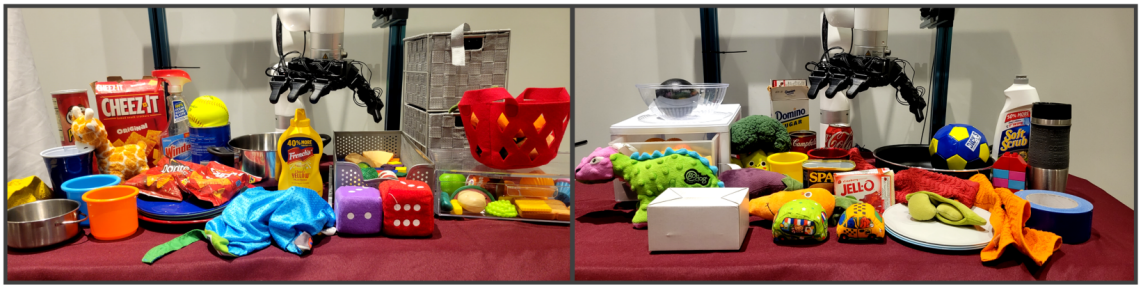

Figure 5.2: The collection of train objects (left) and test objects (right) used for experimentation.

action-driven datasets we focus on for dexterous manipulation.

Learning Action from Videos Detecting humans, estimating poses of different body parts, or understanding the dynamics and interactions related to human motion is a commonly studied problem. One can model human hands using the MANO [199] model and the human body using SMPL, SMPL-X [158, 182] models. There are many efforts in human pose estimation such as [123, 200, 230]. We focus on FrankMocap [200] for our project as it is robust for online videos. Traditionally, teleoperation approaches have employed hand markers with gloves for motion capture [106] or VR settings [136]. Without gloves, Li et. al. [144] used depth images and a paired human-robot dataset for teleoperation, and Handa et. al. [108] designed a system that mimics the functional intent of the human operator to perform object manipulation tasks.

Robot Learning by Watching Humans Recent works have leveraged human datasets to learn cost functions [45, 68, 208], learn action correspondences [202] both in a paired [209] and unpaired manner [218]. This data can also be used to extract explicit actions by leveraging structure in the collection (such as reacher-grabber tools [240]) or prediction of future hand and object locations [139], as well as keypoint detectors [80]. This can also be used to build representations for robot learning [178, 179]. R3M [178] trains on the Ego4D [102] dataset using a temporal alignment loss between language labels and video frames. We build on top of previous efforts in this area, where we combine visual representations trained on human activity data, with *action* driven representations.

5.4 Background

5.4.1 Neural Dynamic Policies

Neural Dynamic Policies (NDPs) [43, 44, 83], produce smooth and safe open-loop trajectories. When using them as a network backbone, they can be rolled out to trajectories of arbitrary lengths which enables the use of varying-length human videos. NDPs can be described with the Dynamic Movement Primitive equation [117, 180, 189, 201]:

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f_w(x, g), \tag{5.1}$$

where y is the coordinate frame of the robot, g is the desired goal in the given coordinate frame, f_w is a radial basis forcing function, x is a time variable, and α, β are global constants. NDPs use the robot state, scene, and a NN to output the goal g and shape parameters w of the forcing function f_w.

5.4.2 Learning from Watching Humans

Recently, Sivakumar et al. [216] introduced Robotic Telekinesis, a pipeline that teleoperates the Allegro Hand [5] using a single RGB camera. Leveraging work in monocular human hand and body pose estimation [200], hand and body modeling [158, 182, 199], and human internet data, Robotic Telekinesis real-time re-targets the human hand and body to the robot hand and arm. Due to its efficiency and ease of use, we leverage Sivakumar et al. [216]'s approach for demonstration collection.

We borrow the human hand to robot hand retargeting method from Robotic Telekinesis [216] that manually defines key vectors v_i^h and v_i^r between palms and fingertips on both the human and robot hand. They build an energy function E_π which minimizes the distance between human hand poses (β, θ) and robot hand poses q. c_i is a scale parameter. Therefore, the energy function is defined as:

$$E_\pi(\ (\beta_h, \theta_h),\ q\ ) = \sum_{i=1}^{10} ||v_i^h - (c_i \cdot v_i^r)||_2^2 \tag{5.2}$$

Sivakumar et al. [216] train an MLP $H_R(.)$ to implicitly minimize this energy function in 5.2, conditioned on knowing human poses (β, θ). For more details, we refer the readers to Sivakumar et al. [216].

5.5 Learning Dexterity from Human Videos

We learn general-purpose manipulation by utilizing large-scale human hand action data as prior robot experience. We leverage not only visual priors of the scene's appearance but also leverage important aspects of the human hand's motion, intent, and interaction. To do this, we *re-target* the human video data to trajectories from the robot's embodiment and point of view. By pretraining policies with these human hand trajectories, we learn *action* priors on how the robot should behave. However, it's notoriously difficult to leverage these noisy human video detections. Therefore, we must also employ a policy with *physical* priors to learn smooth and robust policies that do not overfit to noise. We explain insights and our method used to leverage *action* priors in the sections below.

5.5.1 Visual Priors from Human Activity Data

Many previous works [176, 178, 234] have tackled visual priors and representations for robot learning. These networks often encode some form of semantic visual priors into the pretrained network from human video internet datasets. We use the encoder from Nair et al. [178] as a useful visual initialization for our policy. Nair et al. [178] is trained on a visual-language alignment as well as a temporal consistency loss. Our network takes human video frames and processes them using the publicly released ResNet18 [112] encoder, E_ϕ from R3M [178]. The output of this network is our visual representation for learning.

5.5.2 Action Priors from Human Activity Data

While visual pretraining aids in semantic understanding, human data contains a lot more information about how to interact with the world. VideoDex uses action information to pretrain an action prior, a network initialization that encodes information about the typical actions for a particular task.

However, training robot policies on human actions are difficult, as there is a large embodiment gap between humans and robots as described in Handa et al. [108] and Sivakumar et al. [216] Thus, we must re-target the motion of the human to the robot embodiment to use it in training. This problem is solved using three main components. First, we detect human hands in videos. Second, we project hand poses H to robot finger joints H_r. Finally, we convert human wrist pose P to robot arm pose P_r. H_r and P_r define the trajectory of the human in the robot's frame, from which we can extract actions to

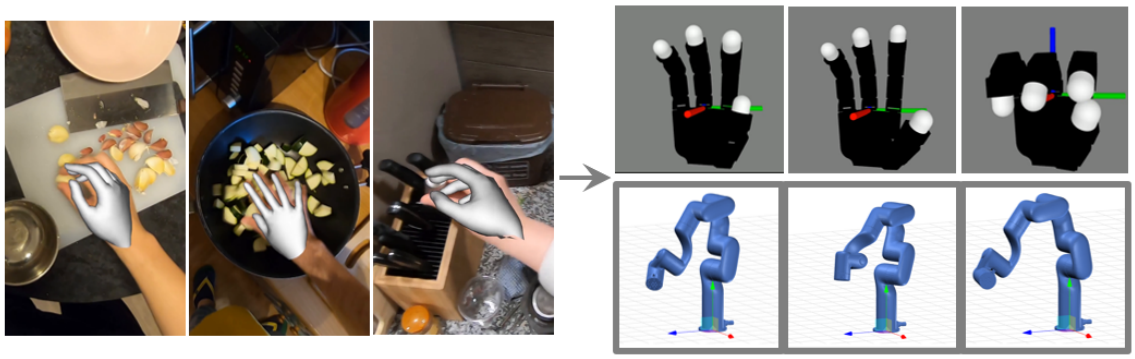

Figure 5.3: To use internet videos as pseudo-robot experience, we re-target human hand detections from the 3D MANO model [199] to 16 DoF robotic hand (LEAP) embodiment and we retarget the wrist from the moving camera to the xArm6 [27] embodiment. Videos at https://video-dex.github.io

pretrain our policy network with the action prior. See Figure 5.4 for a summary of the stages.

Action and Hand Detections First, we must detect the right actions the human is completing. To expedite development, we use the action annotations from the EpicKitchens dataset [78] but an action detection network such as [75] can be used. Now, we must detect the hand. VideoDex first computes a crop c around the operator's hand using OpenPose [62] and the result is passed to FrankMocap [200] to obtain hand shape (β) and pose parameters (θ) of the 3D MANO model [199]. These parameters are passed through a low pass filter and subsequently used in re-targeting to the robot.

Re-targeting Wrist Pose In this section, we show how to compute the transformation that describes the wrist pose in the robot frame denoted as M^{Wrist}_{Robot}. First, to calculate $M^{Wrist}_{C_t}$, where C_t is the camera frame at timestep t we leverage the Perspective-n-point algorithm [96]. This takes 2D keypoint outputs (u_i, v_i) by the hand detection model and 3D keypoints from the hand model (x_i, y_i, z_i) and computes $M^{Wrist}_{C_t}$. To accurately obtain camera intrinsics for PnP, COLMAP is used [203].

In human egocentric video datasets, the position of the camera is not fixed and we must compensate for this movement. Specifically, we compute the transformation between the camera pose in the first frame C_1 and all other frames in the trajectory, C_t. We call this transform $M^{C_t}_{C_1}$. To estimate this, we run monocular SLAM, specifically ORBSLAM3 [61].

Computing wrist poses in the first camera coordinate frame is important but this is still

not in the robot frame because the robot is always upright. To be able to transform the human trajectory in the robot's frame, we must find the vector that is parallel to gravity in the camera's frame, α_p. Thus recover object segmentations for surfaces that are parallel to the floor such as tables, floors, counters, and similar synonyms using a state-of-the-art object detector (Detic [246]). Then an estimated depth map from RGB frames only using Adabins [49] is computed. This way, the method does not rely on the long-term contiguity of a video like most SLAM approaches. We then use depth map portions that correspond to the relevant objects and calculate a surface normal vector. We estimate α_p using this normal vector and the following equations:

$$\text{pitch} = \tan^{-1}(x_{Acc}/\sqrt{y_{Acc}^2 + z_{Acc}^2}) \tag{5.3}$$

$$\text{roll} = \tan^{-1}(y_{Acc}/\sqrt{x_{Acc}^2 + z_{Acc}^2}) \tag{5.4}$$

Detailed ablations on the parameterization of the initial pitch of the predicted trajectory (α) are provided in Section 6.6. In SLAM, we also remove the dependency on gyroscope data by assuming that the scaling factor is 1.0. This is acceptable because the trajectory is rescaled to the robot frame later. Therefore, this wrist re-targeting approach uses only 2D images from human videos.

Since the robot has workspace limits, and we would also like to center the starting pose of the robot, we heuristically compute T_{Robot}^{World} which rescales and rotates the human trajectory in the world frame τ_W^{wrist} into the robot trajectory τ_R^{wrist}. The final function to obtain M_{Robot}^{Wrist} can be described as:

$$M_{Robot}^{Wrist} = T_{Robot}^{World} \cdot M_{World}^{C_1} \cdot M_{C_1}^{C_t} \cdot M_{C_t}^{wrist} \tag{5.5}$$

Re-targeting Hand Pose Human hands are also in a different *embodiment* compared to that of robot hands, like our 16 DOF LEAP Hand [210] . Similarly, to Sivakumar et al. [216], we use $H(.)$ to map hand poses to robot hand poses. Given human detected pose x_h, we obtain $x_r = H(x_h)$ using a similar re-targeting network to Sivakumar et al. [216], and get human hand trajectories: τ_R^{hand} in the robot's embodiment. We use τ_R to denote the combined hand and wrist trajectories: τ_R^{hand}, τ_R^{wrist}. See Figure 5.3 for a visualization.

Algorithm 1 Procedure for Videodex

Require: Human videos $V^H_{1:K}$ (length T), policy π_θ, demonstrations $\mathcal{D}_{1:N}$. Human detection f_{human} [200].
  for $k = 1...K$ **do**
    for $t = 1...T$ **do**
      Pose parameters $\theta_t, \beta_t = f_{\text{human}}(I_t)$
      Get wrist pose w_t from 5.3, 5.4 and 5.5,
      Hand pose $h_t = H(\theta_t, \beta_t)$
    end for
    Store all h_t, w_t into robot trajectory τ^k_R
    $\hat{\tau}^k_R = \pi_\theta(I^k_1, h^k_1, w^k_1)$
    Optimize $\mathcal{L}_\theta = ||\tau^k_R - \hat{\tau}^k_R||_1$
  end for
  Store policy weights θ_h to initialize π_θ
  while not converged **do**
    for $n = 1...N$ **do**
      $\tau_n, I^n_{1:T} = \mathcal{D}_n$
      $\hat{\tau}_n = \pi_\theta(I^n_1, h^n_1, w^n_1)$
      Optimize $\mathcal{L}_\theta = ||\tau_n - \hat{\tau}_n||_1$
    end for
  end while=0

5.5.3 Learning with Human Videos

We must design an open-loop policy π that learns first from the re-targeted human trajectories (the action prior) and then from real robot trajectories collected in teleoperation. Naively, training a neural network policy on τ_R will lead to overfitting to noisy hand detections. To circumvent this, we first use visual priors from the visual ResNet-based [112] encoder provided by Nair et al. [178], E_ϕ. Then, we introduce a *physical prior* to the network, the physically-inspired Neural Dynamic Policies [43, 44].

We construct π with the following setup. We first process the first scene image I with the visual encoder E_ϕ. Then the extracted features $E_\phi(I)$ are used to condition an NDP for the wrist and hand separately, f_{wrist} and f_{hand}. Concretely, each NDP operates by processing the input features with a small MLP which outputs w, g which are the trajectory shape and goal parameters. The forward integrator of the NDP outputs an open-loop trajectory for the hand and the wrist, $\hat{\tau}_R$. We use the following loss function:

$$\mathcal{L} = \sum_k \text{Loss}_{L1}(\tau_R - [f_{\text{hand}}(E_\phi(I_k)), f_{\text{wrist}}(E_\phi(I_k))])$$

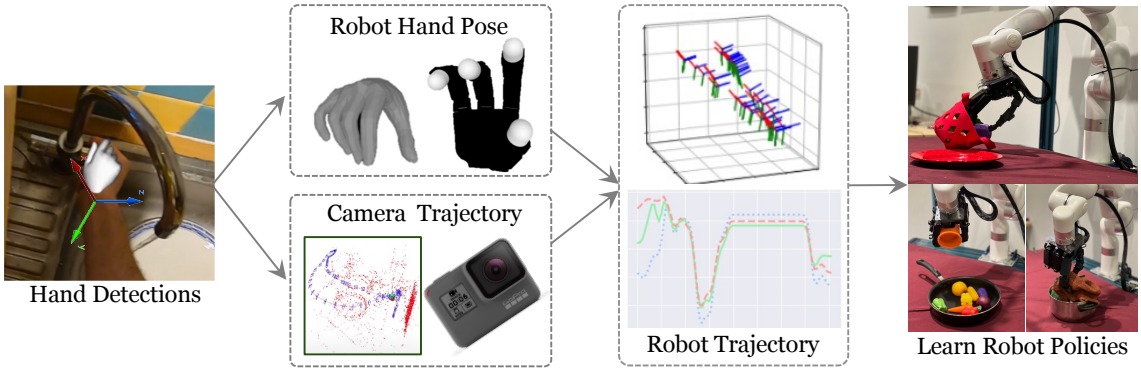

Figure 5.4: To use human videos as an action prior for training policies, we re-target them to the robot embodiment. The detected human fingers are converted to the robot fingers using a learned energy function. The wrist is re-targeted using the detections and camera trajectory and transformed to the robot arm.

Training Methodology: We use between 500-3000 video clips of humans completing the same task category as the robot will from the Epic Kitchens dataset [78]. For example, in pick, there are close to 3000 video clips of humans picking items. These are retargeted to the robot domain and used to pretrain the network with the human action prior of the pick task. Then, the final policy π is trained on a few teleoperated demonstrations of pick on the real robot. The full training takes about 10 hours on a single 2080Ti GPU. More training details can be found in the appendix and in Algorithm 1. Our network consists of the R3M [178] initialized ResNet-18 [112]. We process these features with a 3-layer MLP with a hidden layer size of 512, which are then processed by 2 NDP [43] networks.

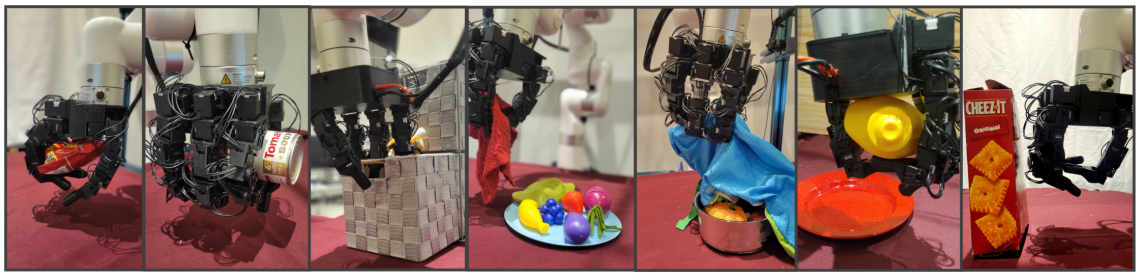

Figure 5.5: Tasks used in experiments. From left to right: pick, rotate, open, cover, uncover, place and push. See https://video-dex.github.io for videos of these tasks.

5.6 Experimental Setup

We perform thorough real world experiments on manipulation tasks, specifically many tasks that require dexterity. See our webpage for result videos. We aim to answer the following questions. (1) Is VideoDex able to perform general purpose open-loop manipulation? (2) How much does the action prior of VideoDex help? (3) How much does the physical prior of the NDPs in VideoDex help? (4) What important design choices are there (visual priors, physical priors, or training setup)?

Task Setup We pretrain action priors on retargeted Epic Kitchens data for seven robot tasks. Then, we collect about 120-175 demonstrations for each of these tasks on our setup to train the policy. In `pick`, the goal is to pickup an object. In `rotate`, the agent grasps and rotates the object in place. In `cover` and `uncover`, the goal is to cover or uncover a pan/plate with a soft cloth object. `Push` involves flicking/poking an object with the fingers. In `place`, the robot has to pick up an object and place it into a plate, pan or pot. In `open` we open three different drawers. Our testing procedure consists of unseen locations and objects. Details on the tasks and objects are in the supplemental.

While robot hands can provide great dexterity, we also investigate whether 2-finger grippers can benefit from action priors. The internet data is converted to where the closed human hand is a closed 2-finger gripper, and the open human hand is an open 2-finger gripper. We collect separate demonstrations on the real-robot using the 2-finger gripper from xArm [27]. Separate action priors are trained for the 16 DoF LEAP Hand and the 2-finger gripper.

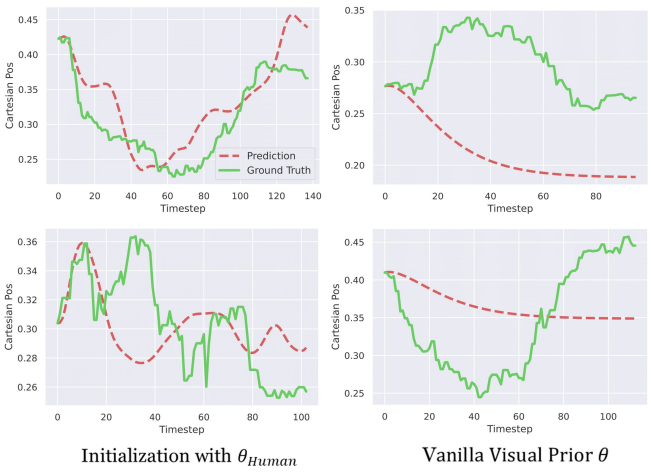

Figure 5.6: Networks initialized using action priors on human data without further training are closer to ground truth robot trajectories than networks only initialized using visual priors.

	Pick		Rotate		Open		Cover		Uncover		Place		Push	
	train	test	train	test	train	test	train	test	train	test	train	test	train	test
BC-NDP [44]	0.64	0.38	**0.94**	0.56	**0.90**	0.60	**0.78**	0.58	0.88	0.82	0.70	0.35	1.00	0.71
BC-Open[83]	0.50	0.44	0.72	0.38	0.80	0.40	0.44	0.58	**1.00**	**0.91**	0.40	0.25	1.00	0.93
BC-RNN [83]	0.56	0.31	0.78	0.50	**0.90**	0.50	0.56	0.42	0.88	0.75	0.70	0.50	1.00	**1.00**
VideoDex	**0.83**	**0.77**	0.85	**0.71**	**0.80**	**0.80**	**0.75**	**0.63**	**0.96**	0.92	**0.89**	**0.80**	1.00	**1.00**

Table 5.1: We present the results of train objects and test objects for Videodex and baselines as described above.

5.7 Results

First, we evaluate the need for initialization with the action priors obtained from the human internet videos. θ_h The baseline without internet pre-training is called BC-NDP. It uses the same physical prior and visual network initialization, without the initialization from θ_h. We also compare the effect of the action prior on 2-finger gripper policies. Second, we compare against two standard open-loop behavior cloning approaches introduced in recent benchmarks [83]. BC-open uses a 2 layer MLP instead of the NDP network. BC-RNN, uses an RNN to pre-process the visual features and then a two-stream, 2 layer MLP for wrist and hand trajectories. We try an offline RL ablation CQL [135], where we use the demonstrations as a sparse reward. We train a behavior cloning policy with the action prior from human videos without the physical prior of the NDP. We call this VideoDex-BC-Open. We ablate the type of visual representation and prior use by trying an initialization using the VGG16 network [215] (VideoDex-VGG) and the MVP network [234] [114] (VideoDex-MVP) based representation trained for robot learning. We ablate the need for a two stream policy, instead training a single NDP for both hand and wrist. (VideoDex-Single) To see if VideoDex works with fewer demonstrations (around 50 demonstrations, 5-7 per variant only), we train a policy called VideoDex-Constrained.

We analyze the results of our experiments and the guiding questions discussed in Section 5.6. We present the results of our findings as a 0-1 success rate in Table 6.2 and the result of the ablations we ran on the place task in Table 6.3.

Effect of Action Priors We firstly compare VideoDex against methods that do not employ an action prior trained on human data, as explained in Section 5.6. For almost all of the tasks, VideoDex either outperforms baselines or has a similar performance, especially for

held out objects/instances. We believe that one of the key aspects of VideoDex generalizing to test objects is the action prior pretraining on human videos. This can be seen in Figure 5.6. Without ever training on the robot demonstrations, the trajectories initialized using the action prior pretrained network θ_h (left) are much closer to the ground truth trajectories of a network that is initialized using only a visual prior such as the encoder from Nair et al. [178] (right). From the results, we see that `VideoDex-BC-Open` with action priors (Table 6.3) outperforms `BC-Open`. Having a physical prior added (`BC-NDP`) tends to help, but it is not the case for every task. We suspect that some tasks require smoother behavior than others. Additionally, in Table 6.3 our offline RL baseline, `CQL` [135] does not perform as well as the rest of the approaches, even under-performing the Behavior Cloning setup. Qualitatively, we see a much less smooth and less safe execution with this method, thus we only perform it on one task (`place`). Note that we use the same visual prior for this as well.

	Place	**Open**	**Pick**
`1-DOF BC-Open`[83]	0.62	0.69	0.71
`1-DOF VideoDex`	**0.69**	**0.82**	**0.77**

Table 5.2: We compare how the 1-DOF xArm gripper performs using Videodex. [27] Separate demonstrations were collected using this gripper.

Hand vs 2-Finger Gripper We compare whether the action priors from VideoDex also help in the more general 1-DOF gripper setting. In Table 5.2, we find that in the 1-DOF setting, VideoDex still improves performance on these tasks. This is because the priors from human internet videos still encode typical wrist trajectory behaviors as well as when the gripper should close for each task.

	Place	**Cover**	**Uncover**
`VideoDex-Fixed`	0.55	0.50	0.77
`VideoDex-Random`	0.45	0.63	0.85
`VideoDex-IMU`	0.70	**0.67**	0.90
`VideoDex`	**0.80**	0.63	**0.92**

Table 5.3: Ablations that compare the different ways of calculating the initial pitch of the camera with respect to gravity, on test objects. This enables us to transform human trajectories to be upright like the robot is.

Initial Pose Computation Comparison We compare three different ways to estimate α_p or $M^{C_1}_{World}$, the vector that points parallel to gravity. These methods contrast with VideoDex which uses the surface normal of objects that are typically parallel with the floor to calculate the direction of gravity. `VideoDex-Fixed`, assumes that α_p is [0,0]. This is reasonable as we are not relying on robots to exactly mimic the human but get a general action prior. `VideoDex-Random`, randomizesα_p in the range of 15-45 degrees, which is the typical

egocentric camera angle. `VideoDex-IMU` uses the internal image stabilization sensor data to estimate the upright vector. None of these approaches use gyroscope data in SLAM, as we assume that the scaling factor is 1.0. In Table 5.3, we present the results of these experiments. The performance degrades when randomizing or setting $M^{C_1}_{World}$ to a fixed value, in all three of the tasks, but it is still comparable to or better than our baselines that do not use any human action data. A possible explanation for the fact that `VideoDex-Surface` performed better than our `VideoDex-IMU` is that the sensor data may be noisy and estimating surface normals from visual features is more robust.

Effect of Physical Priors and Architectural Choices We compare different types of physical priors in Table 6.2 and in Table 6.3. In general (`BC-NDP`) tends to outperform baselines without a physical prior, except for `BC-RNN` in a couple of tasks. `BC-RNN` performs less aggressive behavior, which allowed it to efficiently grasp more objects. In Table 6.3 it's shown that an important physical prior is to treat the wrist and the hand in a more disentangled manner, as the performance for `VideoDex-Single` tends to drop compared to `BC-NDP` and `VideoDex-BC-Open` (Behavior Cloning with our action prior pretraining). The two stream architecture aids in learning, as it allows the policy to disentangle the actions of the wrist and the hand. This is important as the same grasp might be used for picking objects in many different locations, and similarly, it is possible to localize many objects and perform completely different types of interactions.

	Train	**Test**
Baselines:		
`BC-NDP` [44]	0.70	0.35
`BC-Open` [83]	0.40	0.25
`BC-RNN` [83]	0.70	0.50
`CQL` [135]	0.40	0.20
No Physical Prior:		
`VideoDex-BC-Open`	0.50	0.50
`VideoDex-Single`	0.50	0.30
Visual Prior Ablation:		
`VideoDex-VGG`	0.20	0.20
`VideoDex-MVP`	0.40	0.20
Constrained Data:		
`VideoDex-Const-5`	0.80	0.60
`VideoDex-Const-10`	0.50	0.30
`VideoDex` (ours)	**0.90**	**0.70**

Table 5.4: We present the results of the ablations discussed in Section 5.6. These are all performed on the `place` task.

Generalization with Less Data We limit VideoDex to a maximum of 5 and 10 teleoperated demonstrations per variant (we have 12-15 variants in our setup). As shown in Table 6.2, even with 5 instances per variant, we still see a 30% success rate for unseen objects. Empirically, the policies generally go to the right area but are not able to grasp

objects properly. With less robot experience, VideoDex outperforms which demonstrates that action priors also boosts sample efficiency.

Effect of Visual Priors We compared using our approach with MVP (`VideoDex-MVP`) [234] and VGG (`VideoDex-VGG`) [215] and their performance was below VideoDex using Nair et al. [178]. This is likely because both encoders are much larger than the ResNet18 [112] we use and require a lot more training time than feasible on human videos. However, `VideoDex-MVP` still performs better than `VideoDex-VGG`, which indicates that using a visual prior trained on human data does in fact help, as Xiao et al. [234] trained the representation in self-supervised fashion on videos and use the embeddings to perform robotics tasks in simulation. We see in Table 6.2, that while visual priors are important, action priors are more impactful.

Choice of Robotic Hand In our experiments, we also tried using the Allegro Hand [5]. We found that the Allegro had higher inaccuracy in control and more hardware failures as compared to LEAP Hand. LEAP Hand outperformed the Allegro Hand $7-12\%$ on average in all experiments, thus we use it for our setup [210] .

5.8 Discussion and Limitations

Although we see strong results on the held-out objects, VideoDex has several limitations and scope for future work. First, we focus on curated human video datasets, such as EpicKitchens [78], but only use these as a convenience to expedite our process. It is possible to filter internet videos of humans according to tasks using action detectors and then process them with VideoDex. We also use camera data in VideoDex but show that with a heuristic driven approach it is possible to obtain similar or better results. Second, we rely on off-the-shelf human hand detection modules that very often have erroneous 6D pose detections, especially when the hand is interacting with objects. Third, the action priors rely on the arm trajectory as well as the hand trajectory retargeting which must be recomputed for each different set of robot parameters and embodiment. Finally, our method of behavior cloning in the real world is currently open-loop, so it cannot react to changes in the environment. This is because closed-loop behavior cloning is difficult to keep safe in the real world. Similarly, when running closed-loop RL it is difficult to guarantee the safety of the system. We leave this to future work, to train policies that can react to real world

changes.

Chapter 6

DEFT: Dexterous Fine-Tuning for Real-World Hand Policies

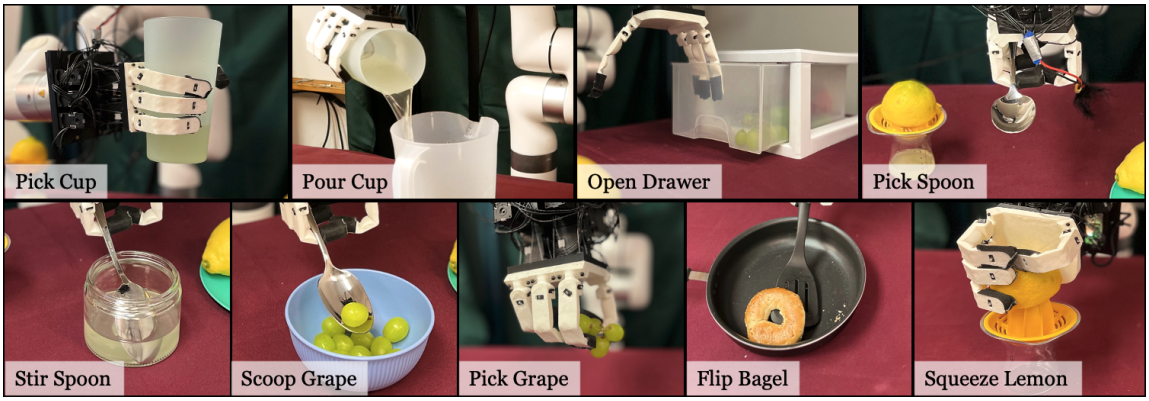

Figure 6.1: We present DEFT, a novel approach that can learn complex, dexterous tasks in the real world in an efficient manner. DEFT manipulates tools and soft objects without any robot demonstrations.

6.1 Abstract

Dexterity is often seen as a cornerstone of complex manipulation. Humans are able to perform a host of skills with their hands, from making food to operating tools. In this paper, we investigate these challenges, especially in the case of soft, deformable objects as well as complex, relatively long-horizon tasks. However, learning such behaviors from scratch can be data inefficient. To circumvent this, we propose a novel approach, DEFT (**DE**xterous **F**ine-**T**uning for Hand Policies), that leverages human-driven priors, which are executed

directly in the real world. In order to *improve* upon these priors, DEFT involves an efficient online optimization procedure. With the integration of human-based learning and online fine-tuning, coupled with a soft robotic hand, DEFT demonstrates success across various tasks, establishing a robust, data-efficient pathway toward general dexterous manipulation. Please see our website at https://dexterous-finetuning.github.io for video results.

6.2 Introduction

The longstanding goal of robot learning is to build robust agents that can perform long-horizon tasks autonomously. This could for example mean a self-improving robot that can build furniture or an agent that can cook for us. A key aspect of most tasks that humans would like to perform is that they require complex motions that are often only achievable by hands, such as hammering a nail or using a screwdriver. Therefore, we investigate dexterous manipulation and its challenges in the real world.

A key challenge in deploying policies in the real world, especially with robotic hands, is that there exist many failure modes. Controlling a dexterous hand is much harder than end-effectors due to larger action spaces and complex dynamics. To address this, one option is to *improve* directly in the real world via *practice*. Traditionally, reinforcement learning (RL) and imitation learning (IL) techniques have been used to deploy hands-on tasks such as in-hand rotation or grasping. This is the case as setups are often built so that it is either easy to simulate in the real world or robust to practice. However, the real world contains tasks that one cannot simulate (such as manipulation of soft objects like food) or difficult settings in which the robot cannot practice (sparse long-horizon tasks like assembly). How can we build an approach that can scale to such tasks?

There are several issues with current approaches for practice and improvement in the real world. Robot hardware often breaks, especially with the amount of contact to learn dexterous tasks like operating tools. We thus investigate using a *soft anthropomorphic hand* [166], which can easily run in the real world without failures or breaking. This soft anthropomorphic hand is well-suited to our approach as it is flexible and can gently handle object interactions. The hand does not get damaged by the environment and is robust to

*Equal contribution, order decided by coin flip.

continuous data collection. Due to its human-like proportions and morphology, retargeting human hand grasps to robot hand grasps is made simpler.

Unfortunately, this hand is difficult to simulate due to its softness. Directly learning from scratch is also difficult as we would like to build *generalizable policies*, and not practice for every new setting. To achieve efficient real-world learning, we must learn a prior for reasonable behavior to explore using useful actions. Due to recent advances in computer vision, we propose *leveraging human data to learn priors* for dexterous tasks, and improving on such priors in the real world. We aim to use the vast corpus of internet data to define this prior. What is the best way to combine human priors with online practice, especially for hand-based tasks? When manipulating an object, the first thing one thinks about is where on the object to make contact, and how to make this contact. Then, we think about how to move our hands *after the contact*. In fact, this type of prior has been studied in computer vision and robotics literature as *visual affordances* [46, 97, 100, 155, 157, 174, 207, 231]. Our approach, DEFT, builds grasp affordances that predict the contact point, hand pose at contact, and post contact trajectory. To improve upon these, we introduce a sampling-based approach similar to the Cross-Entropy Method (CEM) to fine-tune the grasp parameters in the real world for a variety of tasks. By learning a residual policy [84, 119], CEM enables iterative real-world improvement in less than an hour.

In summary, our approach (DEFT) executes real-world learning on a soft robot hand with only a few trials in the real world. To facilitate this efficiently, we train priors on human motion from internet videos. We introduce 9 challenging tasks (as seen in Figure 6.1) that are difficult even for trained operators to perform. While our method begins to show good success on these tasks with real-world fine-tuning, more investigation is required to complete these tasks more effectively.

6.3 Related Work

Real-world robot learning Real-world manipulation tasks can involve a blend of classical and learning-based methods. Classical approaches like control methods or path planning often use hand-crafted features or objectives and can often lack flexibility in unstructured settings [124, 133, 172]. On the other hand, data-driven approaches such as deep reinforcement learning (RL) can facilitate complex behaviors in various settings, but these

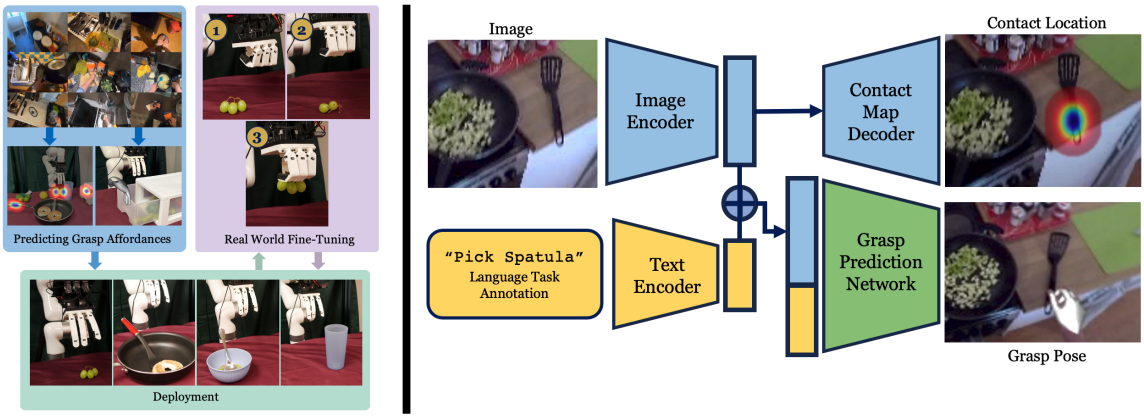

Figure 6.2: **Left:** DEFT consists of two phases: an affordance model that predicts grasp parameters followed by online fine-tuning with CEM. **Right:** Our affordance prediction setup predicts grasp location and pose.

methods frequently rely on lots of data, privileged reward information and struggle with sample efficiency [132, 150, 181, 185, 188]. Efforts have been made to scale end-to-end RL [33, 104, 121, 122, 142, 176] to the real world, but their approaches are not yet efficient enough for more complex tasks and action spaces and are reduced to mostly simple tasks even after a lot of real-world learning. Many approaches try to improve this efficiency such as by using different action spaces [168], goal relabeling [35], trajectory guidance [140], visual imagined goals [176], or curiosity-driven exploration [169]. Our work focuses on learning a prior from human videos in order to learn efficiently in the real world.

Learning from Human Motion The field of computer vision has seen much recent success in human and object interaction with deep neural networks. The human hand is often parametrized with MANO, a 45-dimensional vector [199] of axes aligned with the wrist and a 10-dimensional shape vector. MANOtorch from [236] aligns it with the anatomical joints. Many recent works detect MANO in monocular video [123, 200, 230]. Some also detect objects as well as the hand together [207, 237]. We use FrankMocap to detect the hand for this work. There are many recent datasets including the CMU Mocap Database [9] and Human3.6M [118] for human pose estimation, 100 Days of Hands [207] for hand-object interactions, FreiHand [249] for hand poses, Something-Something [101] for semantic interactions. ActivityNet datasets [92], or YouCook [81] are action-driven datasets that focus on dexterous manipulation. We use these three datasets: [102] is a large-scale dataset with human-object interactions, [156] for curated human-object interactions,

and [78] which has many household kitchen tasks. In addition to learning exact human motion, many others focus on learning priors from human motion. [161, 177] learn general priors using contrastive learning on human datasets.

Learning for Dexterous Manipulation With recent data-driven machine learning methods, roboticists are now beginning to learn dexterous policies from human data as well. Using the motion of a human can be directly used to control robots [107, 217, 225]. Moving further, human motion in internet datasets can be retargeted and used directly to pre-train robotic policies [165, 211]. Additionally, using human motion as a prior for RL can help with learning skills that are human-like [163, 183, 196]. Without using human data as priors, object reorientation using RL has been recently successful in a variety of settings [36, 72]. Similar to work in robot dogs which do not have an easy human analog to learn from, these methods rely on significant training data from simulation with zero-shot transfer [32, 167].

Soft Object Manipulation Manipulating soft and delicate objects in a robot's environment has been a long-standing problem. Using the torque output on motors, either by measuring current or through torque sensors, is useful feedback to find out how much force a robot is applying [41, 238]. Coupled with dynamics controllers, these robots can learn not to apply too much torque to the environment around them [126, 151, 159]. A variety of touch sensors [50, 213, 221, 242] have also been developed to feel the environment around it and can be used as control feedback. Our work does not rely on touch sensors. Instead, we practice in the real world to learn stable and precise grasps.

6.4 Fine-Tuning Affordance for Dexterity

The goal of DEFT is to learn useful, dexterous manipulation in the real world that can generalize to many objects and scenarios. DEFT learns in the real world and fine-tunes robot hand-to-object interaction in the real world using only a few samples. However, without any priors on useful behavior, the robot will explore inefficiently. Especially with a high-dimensional robotic hand, we need a strong prior to effectively explore the real world. We thus train an affordance model on human videos that leverages human behavior to learn reasonable behaviors the robot should perform.

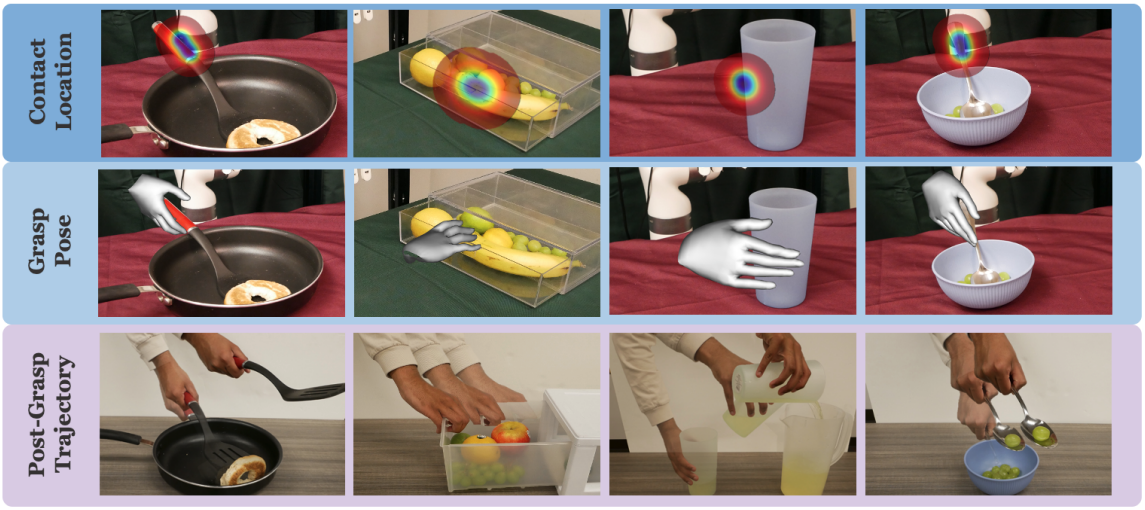

Figure 6.3: We produce three priors from human videos: the contact location (**top row**) and grasp pose (**middle row**) from the affordance prior; the post-grasp trajectory (**bottom row**) from a human demonstration of the task.

6.4.1 Learning grasping affordances

To learn from dexterous interaction in a sample efficient way, we use human hand motion as a prior for robot hand motion. We aim to answer the following: (1) What useful, actionable information can we extract from the human videos? (2) How can human motion be translated to the robot embodiment to guide the robot? In internet videos, humans frequently interact with a wide variety of objects. This data is especially useful in learning object affordances. Furthermore, one of the major obstacles in manipulating objects with few samples is accurately grasping the object. A model that can perform a strong grasp must learn *where* and *how* to grasp. Additionally, the task objective is important in determining object affordances–humans often grasp objects in different ways depending on their goal. Therefore, we extract three items from human videos: the grasp location, human grasp pose, and task.

Given a video clip $V = \{v_1, v_2, \ldots, v_T\}$, the first frame v_t where the hand touches the object is found using an off-the-shelf hand-object detection model [207]. Similar to previous approaches [46, 100, 155, 174], a set of contact points are extracted to fit a Gaussian Mixture Model (GMM) with centers $\mu = \{\mu_1, \mu_2, \ldots, \mu_k\}$. Detic [247] is used to obtain a cropped image v_1' containing just the object in the initial frame v_1 to condition the model. We use Frankmocap [200] to extract the hand grasp pose P in the contact frame

v_t as MANO parameters. We also obtain the wrist orientation θ_{wrist} in the camera frame. This guides our prior to output wrist rotations and hand joint angles that produce a stable grasp. Finally, we acquire a text description T describing the action occurring in V.

We extract affordances from three large-scale, egocentric datasets: Ego4D [102] for its large scale and the variety of different scenarios depicted, HOI4D [157] for high-quality human-object interactions, and EPIC Kitchens [78] for its focus on kitchen tasks similar to our robot's. We learn a task-conditioned affordance model f that produces $(\hat{\mu}, \hat{\theta}_{\text{wrist}}, \hat{P}) = f(v'_1, T)$. We predict $\hat{\mu}$ in similar fashion to [46]. First, we use a pre-trained visual model [178] to encode v'_1 into a latent vector z_v. Then we pass z_v through a set of deconvolutional layers to get a heatmap and use a spatial softmax to estimate $\hat{\mu}$.

To determine $\hat{\theta}_{\text{wrist}}$ and $\hat{P}$, we use z_v and an embedding of the text description $z_T = g(T)$, where g is the CLIP text encoder [194]. Because transformers have seen success in encoding various multiple modes of input, we use a transformer encoder $\mathcal{T}$ to predict $\hat{\theta}_{\text{wrist}}, \hat{P} = \mathcal{T}(z_v, z_T)$. Overall, we train our model to optimize

Parameter	Dimensions	Description
μ	3	XYZ grasp location in workspace
θ_{wrist}	3	Wrist grasp rotation (euler angles)
P	16	Finger joint angles in soft hand

Table 6.1: Parameters that are fine-tuned in the real world. The affordance model predicts a 45-dimensional hand joint pose for P, which is retargeted to a 16-dimensional soft hand pose.

$$\mathcal{L} = \lambda_\mu ||\mu - \hat{\mu}||_2 + \lambda_\theta ||\theta_{\text{wrist}} - \hat{\theta}_{\text{wrist}}||_2 + \lambda_P ||P - \hat{P}||_2 \tag{6.1}$$

At test time, we generate a crop of the object using Segment-Anything [131] and give our model a task description. The model generates contact points on the object, and we take the average as our contact point. Using a depth camera, we can determine the 3D contact point to navigate to. While the model outputs MANO parameters [199] that are designed to describe human hand joints, we retarget these values to produce similar grasping poses on our robot hand in a similar manner to previous approaches [109, 217]. For more details, we refer readers to the appendix.

In addition to these grasp priors, we need a task-specific post-contact trajectory to successfully execute a task. Because it is challenging to learn complex and high-frequency

Figure 6.4: **Left**: Workspace Setup. We place an Intel RealSense camera above the robot to maintain an egocentric viewpoint, consistent with the affordance model's training data. **Right**: Thirteen objects used in our experiments.

action information from purely offline videos, we collect one demo of the human doing the robot task (Figure 6.3) separate from the affordance model f. We extract the task-specific wrist trajectory after the grasp using [200]. We compute the change in wrist pose between adjacent timesteps for the first 40 timesteps. When deployed for fine-tuning, we execute these displacements for the post-grasp trajectory. Once we have this prior, how can the robot *improve* upon it?

6.4.2 Fine-tuning via Interaction

The affordance prior allows the robot to narrow down its learning behavior to a small subset of all possible behaviors. However, these affordances are not perfect and the robot will oftentimes still not complete the task. This is partially due to morphology differences between the human and robot hands, inaccurate detections of the human hands, or differences in the task setup. To improve upon the prior, we practice learning a residual policy for the grasp parameters in Table 6.1.

Residual policies have been used previously to efficiently explore in the real world [45, 119]. They use the prior as a starting point and explore nearby. Let the grasp location, wrist rotation, grasp pose, and trajectory from our affordance prior be ξ. During training we sample noise $\epsilon \sim \mathcal{D}$ where $\mathcal{D}$ is initialized to $\mathcal{N}(0, \sigma^2)$ (for a small σ). We rollout a trajectory parameterized by $\xi + \epsilon$. We collect R_i, the reward for each $\xi_i = f(v_i) + \epsilon_i$ where v_i is the image. First, we execute an initial number of M warmup episodes with actions

Algorithm 2 Fine-Tuning Procedure for DEFT

Require: Task-conditioned affordance model f, task description T, post-grasp trajectory τ, parameter distribution $\mathcal{D}$, residual cVAE policy π. E number of elites, M number of warm-up episodes, N total iterations.
$\mathcal{D} \leftarrow \mathcal{N}(\mathbf{0}, \sigma^2)$
for $k = 1 \dots N$ **do**
  $I_{k,0} \leftarrow$ initial image
  $\xi_k \leftarrow f(I_{k,0}, T)$
  Sample $\epsilon_k \sim D$
  Execute grasp from $\xi_k + \epsilon_k$, then trajectory τ
  Collect reward R_k; reset environment
  if $k > M$ **then**
    Order traj indices $i_1, i_2, \dots, i_k$ based on rewards
    $\Omega \leftarrow \{\epsilon_{i_1}, \epsilon_{i_2}, \dots, \epsilon_{i_E}\}$
    Fit $\mathcal{D}$ to distribution of residuals in Ω
  end if
end for
Fit $\pi(.)$ as a VAE to Ω
=0

sampled from $\mathcal{D}$, recording a reward R_i based on how well the trajectory completes the task. For each episode afterward, we rank the prior episodes based on the reward R_i and extract the sampled noise from the episodes with the highest reward (the 'elites' Ω). We fit $\mathcal{D}$ to the elite episodes to improve the sampled noise. Then we sample actions from $\mathcal{D}$, execute the episode, and record the reward. By repeating this process we can gradually narrow the distribution around the desired values. In practice, we use $M = 10$ warmup episodes and a total of $N = 30$ episodes total for each task. This procedure is shown in Algorithm 2. See Table C.1 for more information.

At test time, we could take the mean values of the top N trajectories for the rollout policy. However, this does not account for the appearance of different objects, previously unseen object configurations, or other properties in the environment. To generalize to different initializations, we train a VAE [130, 197, 198, 219] to output residuals δ_j conditioned on an encoding of the initial image $\phi(I_{j,0})$ and affordance model outputs ξ_j from the top ten trajectories. We train an encoder $q(z|\delta_j, c_j)$ where $c_j = (\phi(I_{j,0}), \xi_j)$, as well as a decoder $p(\delta_j|z, c_j)$ that learns to reconstruct residuals δ_j. At test time, our residual policy $\pi(I_0, \xi)$ samples the latent $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and predicts $\hat{\delta} = p(z, (I_0, \xi))$. Then we rollout the trajectory determined by the parameters $\xi + \hat{\delta}$. Because the VAE is conditioned on the initial image,

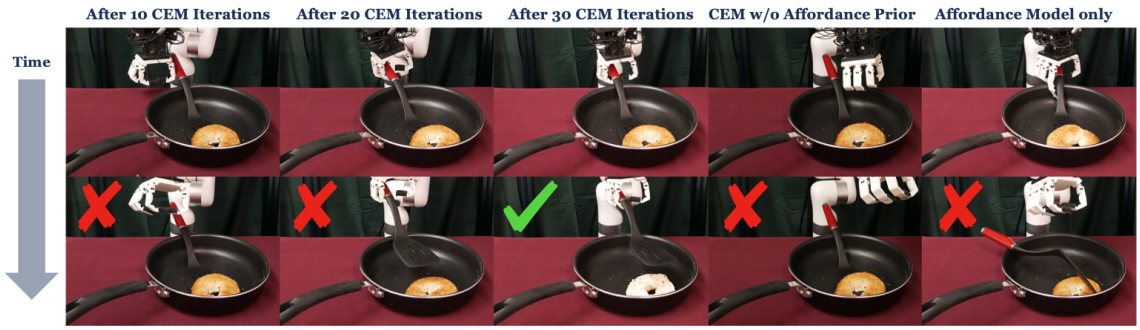

Figure 6.5: Qualitative results showing the finetuning procedure for DEFT. The model learns to hold the spatula and flip the bagel after 30 CEM iterations.

we generalize to different locations and configurations of the object.

6.5 Experiment Setup

We perform a variety of experiments to answer the following: 1) How well can DEFT learn and improve in the real world? 2) How good is our affordance model? 3) How can the experience collected by DEFT be distilled into a policy? 4) How can DEFT be used for complex, soft object manipulation? Please see our website at http://dexterous-finetuning.github.io for videos.

Task Setup We introduce 9 tabletop tasks, *Pick Cup*, *Pour Cup*, *Open Drawer*, *Pick Spoon*, *Scoop Grape*, *Stir Spoon*, *Pick Grape*, *Flip Bagel*, *Squeeze Lemon*. Robotic hands are especially well-suited for these tasks because most of them require holding curved objects or manipulating objects with tools to succeed. For all tasks, we randomize the position of the object on the table, as well as use train and test objects with different shapes and appearances to test for generalization. To achieve real-world learning with the soft robot hand, we pretrain an internet affordance model as a prior for robot behavior. As explained in Section 6.4, we train one language-conditioned model on all data. At test time, we use this as initialization for our real-world fine-tuning. The fine-tuning is done purely in the real world. An operator runs 10 warmup episodes of CEM, followed by 20 episodes that continually update the noise distribution, improving the policy. After this stage, we train a residual VAE policy that trains on the top ten CEM episodes to predict the noise given the image and affordance outputs. We evaluate how effectively the VAE predicts the residuals on each of the tasks by averaging over 10 trials. Because it takes less than an

Method	Pick cup		Pour cup		Open drawer		Pick spoon		Scoop Grape		Stir Spoon	
	train	test	train	test	train	test	train	test	train	test	train	test
`Real-World Only`	0.0	0.1	0.2	0.1	0.1	0.0	0.7	0.3	0.0	0.0	0.3	0.0
`Affordance Model Only`	0.1		0.4		**0.5**		0.5		0.0		0.3	
`DEFT`	**0.8**	**0.8**	**0.8**	**0.9**	**0.5**	**0.4**	**0.8**	**0.6**	**0.7**	**0.3**	**0.8**	**0.5**

Table 6.2: We present the results of our method as well as compare them to other baselines: Real-world learning without internet priors used as guidance and the affordance model outputs without real-world learning.We evaluate the success of the methods on the tasks over 10 trials.

hour to fine-tune for one task, we are able to thoroughly evaluate our method on 9 tasks, involving over 100 hours of real-world data collection.

Hardware Setup We use a 6-DOF UFactory xArm6 robot arm for all our experiments. We attach it to a 16-DOF Soft Hand using a custom, 3D-printed base. We use a single, egocentric RGBD camera to capture the 3D location of the object in the camera frame. We calibrate the camera so that the predictions of the affordance model can be converted to and executed in the robot frame. The flexibility of the robot hand also makes it robust to collisions with objects or unexpected contact with the environment. For the arm, we ensure that it stays above the tabletop. The job will be terminated if the arm's dynamics controller senses that the arm collided aggressively with the environment.

6.6 Results

Effect of affordance model We investigate the role of the affordance model and real-world fine-tuning (Table 6.2 and Figure 6.6). In the real-world only model, we provide a few heuristics in place of the affordance prior. We detect the object in the scene using a popular object detection model [131] and let the contact location prior be the center of the bounding box. We randomly sample the rotation angle and use a half-closed hand as the grasp pose prior. With these manually provided priors, the robot has difficulty finding stable grasps. The main challenge was finding the correct rotation angle for the hand. Hand rotation is very important for many tool manipulation tasks because it requires not only picking the tool but also grasping in a stable manner.

Zero-shot model execution We explore the zero-shot performance of our prior. Without applying any online fine-tuning to our affordance model, we rollout the trajectory parameter-

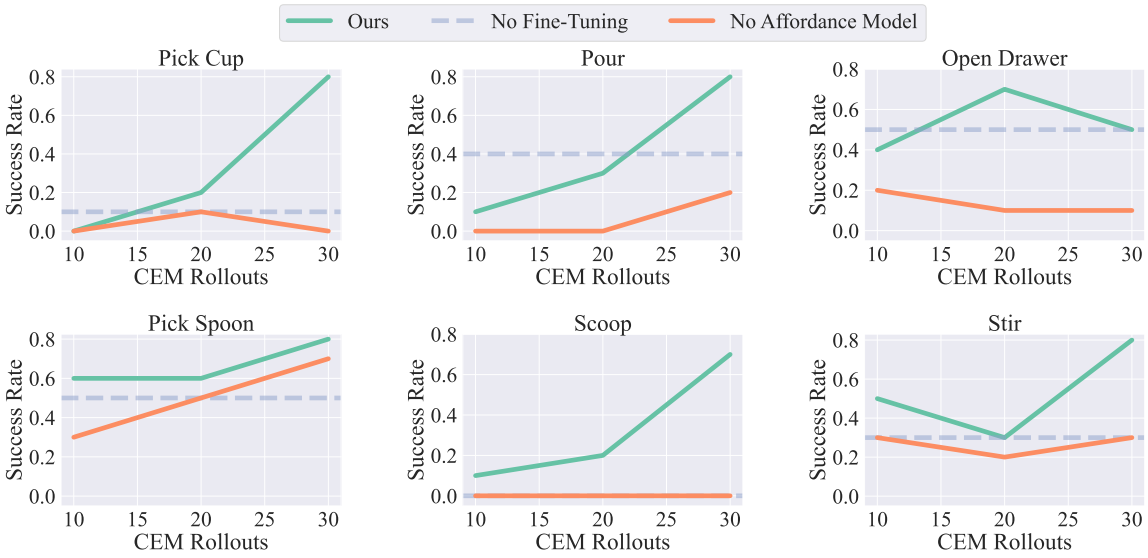

Figure 6.6: Improvement results for 6 tasks: pick cup, pour, open drawer, pick spoon, scoop, and stir. We see a steady improvement in our method as more CEM episodes are collected.

ized by the prior. While our model is decent on simpler tasks, the model struggles on tasks like stir and scoop that require strong power grasps (shown in Table 6.2). In these tasks, the spoon collides with other objects, so fine-tuning the prior to hold the back of the spoon is important in maintaining a reliable grip throughout the post-grasp motion. Because DEFT incorporates real-world experience with the prior, it is able to sample contact locations and grasp rotations that can better execute the task.

Human and automated rewards We ablate the reward function used to evaluate episodes. Our method queries the operator during the task reset process to assign a continuous score from 0 to 1 for the grasp. Because the reset process requires a human-in-the-loop regardless, this adds little marginal cost for the operator. But what if we would like these rewards to be calculated autonomously? We use the final image collected in the single post-grasp human demonstration from Section 6.4 as the goal image. We define the reward to be the negative embedding distance between the final image of the episode and the goal image with either an R3M [178] or a ResNet [112] encoder. The model learned from ranking trajectories with R3M reward is competitive with DEFT in all but one task, indicating that using a visual reward model can provide reasonable results compared to human rewards.

Method	Pour Cup		Open Drawer		Pick Spoon	
	train	test	train	test	train	test
Reward Function:						
R3M Reward	0.0	0.0	0.4	**0.5**	0.5	0.4
Resnet18 Imagenet Reward	0.1	0.2	0.3	0.1	0.4	0.2
Policy Ablation:						
DEFT w/ MLP	0.0	0.0	0.5	0.0	0.6	0.5
DEFT w/ Transformer	0.4	0.5	**0.6**	0.1	0.4	0.5
DEFT w/ Direct Parameter est.	0.1	0.1	0.1	0.0	0.3	0.0
DEFT	**0.8**	**0.9**	0.5	0.4	**0.8**	**0.6**

Table 6.3: Ablations for (1) reward function type, (2) model architecture, and (3) parameter estimation.

Model Architecture We investigate different models and training architectures for the policy trained on the rollouts (Table 6.3). When we replace the conditional VAE with an MLP that predicts residuals, the model has difficulty learning the grasp rotation to effectively pour a cup. We find that the MLP cannot learn the multi-modality of the successful data properly. Our transformer ablation is an offline method similar to [70] where in addition to the image and affordance model outputs, we condition on the reward outputs and train a transformer to predict the residual. At test time the maximum reward is queried and the output is used in the rollout. While this method performs well, we hypothesize that the transformer needs more data to match DEFT. Finally, we train a VAE to directly estimate ξ instead of the residual. This does not effectively distill the information from the affordance prior without the training time allotted. As a result, it often makes predictions that are far from the correct grasp pose.

Performance on complex tasks and soft object manipulation We investigate the performance of DEFT on more challenging tasks. Tasks involving soft objects cannot be simulated accurately, while our method is able to perform reasonably on food manipulation tasks as shown in Figure 6.7.

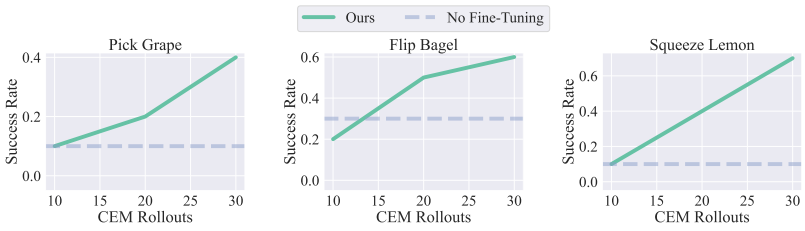

Figure 6.7: We evaluate DEFT on three difficult manipulation tasks.

Of the three tasks, our method has the most difficulty with the Pick Grape task. Because

grapes are small, the fingers must curl fully to maintain a stable grasp. A limitation of our hand is that the range of its joints does not allow it to close the grasp fully and as a result, it has difficulty in consistently picking small objects. This also makes it challenging to hold heavy objects like the spatula in Flip Bagel, but with practice DEFT learns to maintain a stable grasp of the spatula. For Squeeze Lemon, DEFT develops a grasp that allows it to apply sufficient pressure above the juicer. Specifically, our method takes advantage of the additional fingers available for support in hands.

6.7 Discussion and Limitations

In this paper, we investigate how to learn dexterous manipulation in complex setups. DEFT aims to learn directly in the real world. In order to accelerate real-world fine-tuning, we build an *affordance* prior learned from human videos. We are able to efficiently practice and improve in the real world via our online fine-tuning approach with a soft anthropomorphic hand, performing a variety of tasks (involving both rigid and soft objects). While our method shows some success on these tasks, there are some limitations to DEFT that hinder its efficacy. Although we are able to learn policies for the high-dimensional robot hand, the grasps learned are not very multi-modal and do not capture all of the different grasps humans are able to perform. This is mainly due to noisy hand detections in affordance pretraining. As detection models improve, we hope to be able to learn a more diverse set of hand grasps. Second, during finetuning, resets require human input and intervention. This limits the real-world learning we can do, as the human has to be constantly in the loop to reset the objects. Lastly, the hand's fingers cannot curl fully. This physical limitation makes it difficult to hold thin objects tightly. Future iterations of the soft hand can be designed to grip such objects strongly.

Chapter 7

Conclusions

In this thesis, we investigated anthropomorphic robot hands. First we created open-source robot hands that are significantly stronger, more dexterous, cheaper and easier to use than the conventional counterparts.

Second, we tried to emulate the human brain. We used human video from the web to teach robots from a huge amount of data. Robots learned how to retarget human video into the robot embodiment. We taught them how to behave autonomously. Finally, we taught them how to fine-tune their policies with real-world exploration.

While this is titled as a conclusion, it is more of a checkpoint. In the future, I will be continuing my PhD on dexterous manipulation with Prof. Deepak Pathak and targeting a few key ideas.

First, how do we built in more dexterity into our robot hands? How can we take inspiration from our human hands to build robot hands that are more capable? A key part of this will be to take more inspiration from anatomy. While many robot hands only focus on MCP joints on the fingers, how can the MCP joints also effect the palm? This should allow robot hands to better stabilize and grasp around objects. Also, another key aspect of anatomy will be the bones and stiffness of the fingers. How can we design new mechanical joints that emulate the human anatomy and enable better dexterity?

Second, how do we use touch? While there has been much work in touch they are often limited to 2-Finger grippers. Our understanding is that touch could be useful with dexterous robot hands to fine-tune behavior. How can we feel the softness of objects to better manipulate them? How can we use chopsticks or tools? Both building touch sensors

and integrating them with tighter control will be key to doing this.

Finally, I would like to enable more useful, dexterous behavior using big data. How can we create dexterous policies using the simulated data generation and transfer them to real-world robot hands? How can we continue to push for better teleoperation systems to collect more useful data?

Appendix A

Experimental Details for Robotic Telekinesis

A.1 Hand/Body Bounding Box Detection

The first step in our retargeting pipeline is to detect two bounding boxes in an image of the human operator; one for the body and one for the right hand. These bounding boxes needn't be perfectly tight bounding boxes; it's more important that they contain the entire body/hand without truncations. We use an implementation of OpenPose [62] from the authors of FrankMocap [200]. First, a 2D body skeleton detector is run over the entire image, and outputs the predicted pixel locations for each of the 18 keypoints on the skeleton. The tight bounding rectangle around the points is then computed, and a fixed padding is applied on all sides to allow a margin of error.

The right hand bounding box is heuristically extracted based on the 2D body skeleton estimate. The bounding box is centered at the pixel corresponding to the right hand wrist, and the side length of the bounding box is conservatively chosen to ensure that the bounding box contains the entire hand. For an image of size 480x640, we use a side length of 150 pixels.

A.2 Hand Pose Estimation

The next step is to estimate the pose of the operator's right hand from a crop of the hand. The crop of the right hand is computed as described in the previous section, and is resized to a shape of 224x224. The crop is passed to a Convolutional Neural Network (CNN), which outputs a low-dimensional representation of the hand configuration. We use an implementation of the hand pose estimation network from [200]. This network uses a ResNet50 trunk [112], followed by a Multi-Layer Perceptron (MLP) regression head, which outputs three relevant parameters of the SMPL-X model [182]. (1) $\beta_h \in \mathbb{R}^{10}$ describes the *shape* of the hand (the dimensions of each finger and the palm), (2) $\theta_h \in \mathbb{R}^{45}$ describes the *pose* of the hand (how the fingers are arranged) and (3) $\phi_h \in \mathbb{R}^3$ describes the global orientation of the hand (how the hand's root coordinate frame is rotated in the image coordinate frame). The SMPL-X model maps the shape and pose parameters (β_h and θ_h) into a full 3D mesh of the hand, and the global orientation parameter ϕ_h transforms the coordinate frame of the mesh so the axes align with the axes of the image coordinate frame.

A.3 Human-to-Robot Hand Retargeting

Here, we describe two implementations of the human-to-robot hand retargeting module, one which uses online optimization (via inference-time gradient descent), and one which uses offline optimization (via a neural network). Both implementations take a human hand pose as input, and outputs joint angles for each of the 16 Allegro hand joints. The human hand pose is parameterized by the tuple (β_h, θ_h) as described in the previous section. The global orientation ϕ_h is not used in hand retargeting, because the Allegro hand has no wrist or palm joints, and therefore, matching the global orientation of the human hand is accomplished by the robot arm and not the robot hand. We use q_a to denote the vector of the 16 Allegro hand joint angles.

A.3.1 Human-to-Robot Hand Energy Function.

Both implementations of the hand retargeting module minimize the same *energy function*, so we describe this first. Inspired by [108], the energy function aims to capture the functional similarity between a human hand pose and a robot hand pose. Five *keypoints* are defined on

each hand: the index fingertip, the middle fingertip, the ring fingertip, the thumb fingertip, and the palm center. Each of these keypoints is associated with a coordinate frame, and the keypoint is the origin of the coordinate frame. Enumerating all pairs of keypoints yields ten *keyvectors*. Four of them are finger-to-palm keyvectors (index-to-palm, middle-to-palm, ring-to-palm and thumb-to-palm), three are inter-finger keyvectors (index-to-middle, index-to-ring and middle-to-ring), and three are finger-to-thumb keyvectors (index-to-thumb, middle-to-thumb and ring-to-thumb). Notably, each keyvector has one endpoint designated as the origin, and the other as the destination. The keyvectors are expressed in the coordinate basis of the origin keypoint's coordinate frame. We refer the reader to Figure 8 of [108], which elegantly depicts the keypoint coordinate frames and the keyvectors on both the human and Allegro hand.

The energy function between a human hand pose (parameterized by the tuple (β_h, θ_h)) and an Allegro hand pose (parameterized by the joint angles q_a) is computed as follows. First, for each $i \in \{1, \ldots, 10\}$, the i-th keyvector is computed on the human hand (call it $\mathbf{v_i^h}$) and the Allegro hand (call it $\mathbf{v_i^a}$). Then, each Allegro hand keyvector $\mathbf{v_i^a}$ is scaled by a constant c_i. The i-th term in the energy function is the Euclidean difference between $\mathbf{v_i^h}$ and $\mathbf{v_i^a}$:

$$E(\ (\beta_h, \theta_h),\ q_a\ ) = \sum_{i=1}^{10} ||\mathbf{v_i^h} - (c_i \cdot \mathbf{v_i^a})||_2^2 \tag{A.1}$$

These scaling constants $\{c_i\}$ are hyperparameters that require some tuning. If the goal is to produce aesthetically appealing retargeted Allegro hand poses on generic hand gestures, one should set each c_i to around 0.625, in order to account for the ratio in sizes between the average human hand and the Allegro hand. If the goal is to maximize functional similarity, in theory, one should set each c_i to 1, to encourage perfect matching of each keyvector. In practice, we find that setting the constants c_i to a value smaller than 1 is optimal for dexterous manipulation teleoperation. This is because in order to stably grasp an object, the fingers Allegro hand must exert forces pushing into the object. This is achieved by commanding the Allegro finger joints to positions that penetrate the object. These joint angles are never actually reached because the fingers end up colliding with the object, which is precisely the goal. For our experiments, we use a scaling constant of 0.8 for each of the finger-to-thumb and finger-to-finger keyvectors, and a scaling constant of 0.5 for each of the finger-to-palm keyvectors. This means that in order to ensure a stable grasp, operators

must squeeze their fingers closer together than they normally would when grasping, but through our human subject study, we find that novice operators quickly realize this and naturally adjust.

A.3.2 Computing the keyvectors on the human hand.

Having described what the keyvectors are and how they are used to define the energy function, we now describe how to compute the keyvectors on the human hand, given the SMPL-X model parameters (β_h, θ_h). The first step is to use the SMPL-X model to generate a full posed 3D mesh of the human hand. Given β_h and θ_h (and a template hand mesh), the SMPL-X model generates a 3D mesh that correctly captures the shape and pose of the human hand. The next step is to transform these vertices into a canonical coordinate frame, centered at the palm center, with the positive x axis pointing out of the hand, the positive y axis pointing toward the thumb, and the positive z axis pointing toward the middle fingertip. This is done by applying a hand-coded transformation between the SMPL-X coordinate frame and the canonical coordinate frame. The next step is to compute the transformation between each of the keypoint coordinate frames and the canonical coordinate frame. This is done using the Kabsch-Umeyama Algorithm [227] for estimating the transformation that best aligns corresponding pairs of points. Concretely, for each keypoint, we manually determine four vertices on the template hand mesh: (1) the keypoint itself, (2) a vertex located along 0.05m along the positive x axis from the keypoint, (3) a vertex located 0.05m along the positive y axis from the keypoint, and (4) a vertex located 0.05m along the positive z axis from the keypoint. This is pre-computed once, up front. At runtime, for a given posed human hand mesh, we gather the 3D coordinates for each of these three points in the canonical coordinate frame. We define a corresponding set of four points: $\{[0, 0, 0], [0.05, 0, 0], [0, 0.5, 0], [0, 0, 0.5]\}$, which denote the coordinates of these points in the coordinate frame of the keypoint. Given these four correspondences, the Kabsch-Umeyama computes the transformation between the keypoint coordinate frame and the canonical coordinate frame that best aligns these corresponding point pairs.

A.3.3 Computing the keyvectors on the Allegro hand.

Here, we describe how to compute the keyvectors on the Allegro hand, given a vector of Allegro hand joint angles. The key idea here is to exploit forward kinematics. The

URDF of the Allegro hand defines the kinematic skeleton of the Allegro hand. The forward kinematics map takes as input a joint angle vector and outputs the transformation between each link's coordinate frame and the root coordinate frame. Each of our keypoints conveniently corresponds to a particular link on the Allegro hand, so the keypoint coordinate frames can simply be read off from the forward kinematics result.

A.3.4 The energy function is differentiable.

One critical point to note is that the forward kinematics map is a fully differentiable function of the Allegro hand joint angles. This is because forward kinematics is essentially a chain of sines, cosines and matrix multiplications. This is important because it means that the energy function is a fully differentiable function of the Allegro joint angles (by the Chain Rule). This is important because it allows us to compute the gradient of the energy function with respect to the Allegro joint angles, and use gradient descent to find the Allegro joint angles that minimize the energy, with respect to a given human hand pose. We now describe two ways to exploit this differentiability: via online gradient descent and via offline gradient descent.

A.3.5 Retargeting via Online Gradient Descent.

We implement the online optimization retargeter by using Stochastic Gradient Descent (SGD) to minimize the energy function. At each iteration, we seed the Allegro joint angles to an initial value. At the first iteration, that initial value is the all-zero vector (which corresponds to an open palm with outstretched fingers). At all subsequent iterations, the seed vector is the result from the previous iteration. We then run a fixed number of SGD steps with a learning rate of 0.05. The number of gradient steps is a hyperparameter, and presents a tradeoff between accuracy and speed. Running more steps results in better convergence, but takes more time. We find 100 steps to be a good point on this spectrum.

A.3.6 Retargeting via Neural Networks.

We implement the offline optimization retargeter with a neural network that takes as input a human hand pose parameterization $\mathrm{concat}(\beta_h, \theta_h) \in \mathbb{R}^{55}$ and outputs a vector of Allegro joint angles $q_a \in \mathbb{R}^{16}$. We use a Multi Layer Perceptron (MLP) with three hidden layers of

sizes 256, 256, 128, and intermediate tanh activations. We apply a tanh to the final output to squeeze the values from $[-\infty, \infty]^{16}$ to $[-1, 1]^{16}$, and then scale the squeezed values to the appropriate values within each of the joint's ranges. (For example, joint 1 on the Allegro hand has a range of $[-0.196, 1.61]$ (in radians). If the raw output of the network for this joint were 1.23, the tanh operation would squeeze it to 0.84 and the rescale operation would rescale it to 1.47 radians, which is 0.84 of the way between -0.196 and 1.61.)

We train this network using a mixture of human hand poses from the Freihand Dataset [249] and "in-the-wild" human hand poses from the 100 Days of Hands dataset [207]. The Freihand dataset contains ground-truth SMPL-X shape and pose parameters for over 30,000 hand configurations, which we use as inputs to the network. The 100 Days of Hands dataset is simply a list of links to YouTube videos that depict humans using their hands for everyday tasks. In total, these span hundreds of millions of frames. We generate human hand poses by running our Hand Pose Estimation module (built on the CNN from [200]). Our final dataset consists of 30,000 samples from the FreiHand dataset and 30,000 randomly chosen samples from the 100 Days of Hands dataset.

A.4 Body Pose Estimation

The body pose estimation pipeline consists of two steps. The first is to estimate a rough body pose from a crop of the operator's body. The second is to refine the right-hand portion of the rough pose estimate by fusing in the more accurate hand pose estimate from the Hand Pose Estimation module.

A.4.1 Rough Body Pose Estimation via a CNN

This step takes as input a crop of the operator's body, resized to a shape of 224 x 224. The crop is passed to a CNN, which outputs a low-dimensional representation of the body configuration. We use an implementation of the body pose estimation network from [200]. This network uses a ResNet50 trunk [112], followed by a Multi-Layer Perceptron (MLP) regression head, which outputs three relevant parameters of the SMPL-X model. (1) $\beta_b \in \mathbb{R}^{10}$ describes the body *shape* (the dimensions of the various body parts), (2) $\theta_b \in \mathbb{R}^{45}$ describes the *pose* of the body (how the limbs are arranged) and (3) $\phi_b \in \mathbb{R}^3$ describes the global orientation of the body (how the body's root coordinate frame is rotated in the

image coordinate frame). The pose parameter θ_b is of shape $(24, 3, 3)$, and each of these 24 matrices denotes the 3 x 3 rotation matrix of a particular joint in the human body skeleton. The SMPL-X model maps the shape and pose parameters (β_b and θ_b) into a full 3D mesh of the body, and the global orientation parameter ϕ_b transforms the coordinate frame of the mesh so the axes align with the axes of the image coordinate frame.

A.4.2 Body-Pose Refinement via Hand Pose Integration.

The body pose estimate obtained via the CNN can fail to capture the finer details of the hand pose, and crucially, can produce incorrect estimates for the rotation of the right-hand wrist relative to its parent in the human body kinematic chain. The Hand Pose Estimation module, however, operates on a zoomed-in crop of the operator's hand, and often produces much better estimates of the hand's global orientation. To exploit this fact, we use the "Copy-and-Paste" Body-Hand Integration module from [200], which refines the local orientation of the wrist based on ϕ_h, the global orientation of the hand estimated by the Hand Pose Estimation module.

A.5 Human-to-Robot Body Retargeting

We now describe how to map from a body pose estimate (β_b, θ_b) to a target pose for the xArm6's end-effector link, relative to its base link. The first step is to define two pairs of corresponding coordinate frames between the human and robot "bodies". The first pair is between the human torso and the robot "torso". We define the robot torso to be 25cm above the robot base frame. Both torso frames are oriented such that the positive x axis points out of the front of the torso, the positive y axis points towards the left side of the body, and the positive z axis points upwards toward the head. The second pair of coordinate frames is between the human's right hand wrist and the robot's wrist (i.e. end-effector). The wrist coordinate frames are centered at the wrist center, with the positive x axis oriented parallel to the vector originating at the palm center and pointing out of the front of the palm, the positive y axis pointing toward the thumb, and the positive z axis pointing toward the middle fingertip.

The problem of determining the relative transformation between the robot's end-effector and its base coordinate frame reduces to the problem of determining the relative transfor-

mation between the end-effector and the torso (because the torso coordinate frame is fixed relative to the robot's base frame). And this problem reduces to determining the relative transformation between the human's right hand wrist coordinate frame and the human's torso coordinate frame. In order to do this, we start at the torso joint, and traverse the human body kinematic chain (defined by the SMPL-X model) from the torso to the wrist, chaining rotations along the path.

A.6 xArm6 Inverse Kinematics Controller

The human-to-robot body retargeter module outputs target poses for the xArm6's end-effector, relative to it's base coordinate frame. The final step is to build a model that uses this target pose to send a steady and smooth stream of joint angle commands to the xArm6's default controller. In practice, we found that this module is crucial for performance and must be carefully implemented with attention to details; if the robot arm does not move smoothly, dexterous manipulation tasks become impossible.

The first step is to handle outliers caused by erroneous body pose estimates. This is done by computing the difference between the arm's current end-effector pose and the end-effector pose output by the retargeting module. If the difference is greater than a threshold, it is clipped. The next step is to combine the (possibly clipped) end-effector pose target with a running Exponentially Moving Average (EMA) of end-effector poses. This helps ensure smooth motion in the presence of noise in the pose estimation and retargeting modules. The following update rule is used to update running average P_{EMA} to incorporate the new target pose P_{new}:

$$P_{\text{EMA}} = \alpha \cdot P_{\text{new}} + (1 - \alpha) \cdot P_{\text{EMA}} \tag{A.2}$$

We find $\alpha = 0.25$ to work well. We note that a lower value of α can introduce lag, but we find that because our system runs at such a high frequency, this is not an issue in practice.

The next step is to compute the difference between the robot's current end-effector pose, and the (newly updated) pose target, and to apply linear interpolation to divide that difference into equally spaced waypoints. Each waypoint end-effector pose is then passed to a Selectively Damped Least Squares (SDLS) Inverse Kinematics (IK) solver [59], implemented in PyBullet [77], which returns a vector of joint angles for the six joints in the

xArm6. These joint angle commands are sent to the xArm6 servo controller.

A.7 Software Architecture

We now describe how we put together all of the aforementioned modules into a single system that efficiently retargets human motion to robot trajectories. We found it natural to design our system as a dataflow graph, with computation being done at the nodes, and inputs/outputs travelling along the edges. We first summarize the computation nodes we use, and then discuss how we optimized runtime performance by using parallel computation within a publisher-subscriber architecture.

A.7.1 The nodes in the dataflow graph.

Each node corresponds roughly to one of the modules described in previous sections:

- **CameraNode**: captures RGB images of the operator at 30Hz.
- **HandBoundingBoxDetectorNode**: receives an operator image, and computes a bounding box of the right hand.
- **BodyBoundingBoxDetectorNode**: receives an operator image, and computes a bounding box of the body.
- **HandPoseEstimationNode**: receives a crop of the operator's right hand, and estimates the SMPL-X model parameters $(\beta_h, \theta_h, \phi_h)$ that parameterize the hand's shape, pose and global orientation.
- **BodyPoseEstimationNode**: receives a crop of the operator's body, and estimates the SMPL-X model parameters $(\beta_b, \theta_b, \phi_b)$ that parameterize the body's shape, pose and global orientation.
- **BodyHandIntegrationNode**: receives a hand pose estimate $(\beta_h, \theta_h, \phi_h)$ and a body pose estimate $(\beta_b, \theta_b, \phi_b)$, and computes a refined body pose estimate by using the Copy-and-Paste integration method from [200].
- **HandRetargetNode**: receives a hand pose estimate $(\beta_h, \theta_h, \phi_h)$, and computes the Allegro joint angles q_a that maximizes similarity with the operator's hand.
- **BodyRetargetNode**: receives a (refined) body pose estimate $(\beta_b, \theta_b, \phi_b)$, computes

the relative transformation between the right hand wrist and the torso, and converts this to a target pose of the xArm6's end-effector link, relative to its base link.

- **AllegroHandControllerNode**: receives a target Allegro hand joint angle vector from the **HandRetargetNode**, interpolates the difference between robot's current joint angle values and the target into small fixed-size intervals, and sends a stream of interpolated joint angle commands to the robot's controller at a fixed frequency.
- **xArm6ControllerNode**: receives a target end-effector pose from the **BodyRetargetNode**, and commands a smoothly interpolated stream of xArm6 joint angle configurations to the robot's controller at a fixed frequency.

A.7.2 Optimizing performance via parallel computation.

It is crucial that our system run as fast as possible, in order to ensure smooth robot motion and to avoid lagging behind the operator. Therefore, a key design decision was to opt for a parallel computation paradigm. The first implementation of our system sequentially chained together the various modules, and achieved a runtime of approximately 3Hz. In this squential implementation, the retargeting time was the sum of the time taken by each module. Our optimized implementation instead used the ROS publisher-subscriber architecture, with each node running on a separate process. Nodes pass inputs and outputs to each other via inter-process messages. With this approach, the retargeting time was determined only by the slowest node, and this achieved a runtime of approximately 25Hz, which greatly improves usability.

A.8 Hardware Architecture

Our setup consists of a Ufactory xArm6 robot arm mounted to a Vention table, with a Wonik Robotics Allegro Hand mounted as the end-effector. The Allegro Hand was upgraded with four 3D printed fingertips that are skinnier than the default tips. 3M TB641 grip tape is applied to the inner parts of the hand and around the fingertip which allows the Allegro Hand to better grip objects, as 3D printed components and the built in metal/plastic parts are slippery. One Intel Realsense D415 camera tracks the operator; we use only the RGB stream. In our experiments, the operator is standing near the robot, but this is not a requirement. The operator only needs to be able to see the robot, in order for them to adjust their movements

to effectively complete tasks. In the future, we hope to enable this via internet webcams which would allow the operator to be located anywhere in the world. Running the system is a desktop system with an AMD Ryzen 3960x CPU, 128GB of RAM and two NVIDIA GeForce RTX 3080TI GPU's.

A.9 Human Subject Study Details

The 10 subjects that participated in the study were volunteer colleagues from the author's lab. A few were familiar with this project, but they were not intimately familiar with the details. Critically, they had never used the system before. The human subjects were assured that the data collected was anonymous, the robot never interacted with them in any way, and if they ever felt uncomfortable with the task for any reason they could terminate the experiment early. The act of collecting the data would fall under a Benign Behavioral Intervention: verbal, written responses, (including data entry or audiovisual recording) from adult subjects who prospectively agrees and the following is met: Recorded information cannot readily identify the subject (directly or indirectly/linked). This therefore gives an exemption for IRB approval. Example of this category are solving puzzles under various noise conditions, playing an economic game, being exposed to stimuli such as color, light or sound (at safe levels), performing cognitive tasks.

One author was the conductor of the study. The conductor briefed each subject on how to operate the system. The subjects were asked to stand and stay in frame of the camera during the duration of the experiments. They were asked to not move around too quickly as that would trigger safety limits of the control system, but this was never an issue. No other significant instructions were given. The conductor of the study also kept an emergency stop switch next to them for the safety of the robot system, but it was never used.

For the first few trials, many subjects were confused by the system but quickly adapted to it. The conductor instructed them to continually adapt to the system and try to complete the tasks without giving them additional information. The conductor took down notes on the compliments and complaints of the system while the system was being used.

The conductor only verbally told each subject the goal of the task but did not explain the best way to complete them. The tasks were very simple and intuitive so the subjects were not confused by them. For each task, a failure was recorded when either the time expired, the task became impossible to complete from the object state on the table, or the

subject asked for a reset. Between each trial within each task, the subjects were asked to move the robot arm up away from the table to allow the conductor to reset the object. Between each task, the Telekinesis system was paused and subjects were allowed to rest their arm for a few minutes. The dice pickup task and drawer task was 30 seconds each for 7 trials. The last cup in plate task was 60 seconds long for each of the 7 trials. The total time to complete all three tasks was about 15 minutes.

Appendix B

Experimental Details for VideoDex

B.1 Videos

Task videos, performance videos, data collection and example of internet videos can be found at: https://video-dex.github.io

B.2 Additional Ablations

Comparing Effects of Actions, Visual and Physical Priors: Firstly, we ran an ablation where we pertained a policy on human videos performing the place task and finetune it on the uncover task (using robot data). Similarly, we pretrained a policy on Uncover and finetuned on place. The results are in the below table under `VideoDex-Transfer`. We see that for both tasks the performance degrades slightly, especially in the place task. We also train by adding noise to the demonstration trajectories, by adding two different levels of Gaussian noise with standard deviation being 0.01 and 0.05, shown as `VideoDex-Noise-0.01` and `VideoDex-Noise-0.05`. We find that adding more noise definitely hurts the performance of the method. We also train ResNet18 [112] features initialized from ImageNet [88] training instead of the R3M [178] features, and the results in `VideoDex-ImageNet`. We can see that performance drops off, which indicates that the visual priors are important. Note that all of the reported numbers are on test objects. We present the results in Table B.1.

Method/Task	Place	Uncover
VideoDex-Noise-0.01	0.55	0.87
VideoDex-Noise-0.05	0.50	0.60
VideoDex-ImageNet	0.40	0.62
VideoDex-Transfer	0.60	0.87
VideoDex-Original	0.70	0.90

Table B.1: Ablations that compare effects of different action, visual and physical priors, as well as seeing how pretraining on different data transfers to other tasks.

B.3 Retargeting Details

We first retarget human videos from Epic-Kitchens [78]. Specifically, we use the new data (refresher) from their GoPro Hero 7 Black. We retarget video clips of humans completing tasks that are similar to the robot task. These clips are on average 5-10 seconds each, depending on the task.

Wrist in Camera frame The goal of Perspective-n-Point is to estimate the pose of the calibrated camera given a set of N 3D points in the world and their corresponding 2D point projections in the image. First the camera must be calibrated. To do this, we use COLMAP [203] on a set of videos. It tracks keypoints through frames and estimates the calibration from the internet videos. We find these camera intrinsics for the GoPro:

$$\begin{bmatrix} 2304.002572862 & 0 & 960 \\ 0 & 2304.002572862 & 540 \\ 0 & 0 & 1 \end{bmatrix}$$

Using this calibration, we can now complete the Perspective-n-Point process. We are given two sets of points, 16 points in 3D on the hand model in the model's frame $\begin{bmatrix} X_w, Y_w, Z_w, 1 \end{bmatrix}^t$, and another set of 16 2D points in image frame $\begin{bmatrix} u, v, 1, \end{bmatrix}^t$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Transforms	Description	Method
$M_{C_t}^{Wrist}$	Wrist in each Camera	FrankMocap + PnP
$M_{C_1}^{C_t}$	Track Moving Camera	IMU/ORBSLAM
$M_{World}^{C_1}$	Make Camera parallel to Ground	IMU/Stabilization Sensor
T_{Robot}^{World}	Rescale and Reorient for Robot	Heuristic
M_{Robot}^{Wrist}	$T_{Robot}^{World} \cdot M_{World}^{C_1} \cdot M_{C_1}^{C_t} \cdot M_{C_t}^{wrist}$	

Table B.2: Transformations required to calculate wrist in robot frame from passive videos to use in learning. M denotes a transformation matrix, where T is a general transformation .

We use the OpenCV3 solvePnPRANSAC to complete this calculation. This implementation ensures that the process is resilient to erroneous detections.

Camera in First Camera frame In the SLAM section, the goal is to track the camera through the video. This is required to compensate for the movement of the camera. We use this on a selection of videos where we found the camera to move significantly.

We start the SLAM process two seconds before the action clip begins. We mark the start of the action's frame as the first frame, run it through SLAM and then recover the trajectory of the camera through the entire clip. Specifically we recover the transformation: $M_{C_1}^{C_t}$

The process of monocular SLAM is only valid up to a scale factor. Although Epic Kitchens has noisy accelerometer and gyro information from the camera's sensors, we do not use this data to disambiguate this scale factor throughout the duration of the video clip.

ORBSLAM3 [61] only evaluates real-time video going forward through time and does not recalculate prior poses from future information. While this seems imperfect for this purpose, we find that the results are satisfactory for our purpose. In our setup, we are using these retargeted videos as a prior for learning. These retargeted trajectories are not used directly on the robot so they do not need complete accuracy. The more important characteristic is speed. COLMAP [203] can take hours to process larger video clips, but ORBSLAM3 [61] can complete this process faster than real time. This enables us to use many videos as an action prior for the robot behavior.

Camera Parallel to Ground Now that we have the trajectory in the C_1 frame after SLAM and PnP, we still are missing some key transformations to get into the robot frame. First, the C_1 is not always upright compared to gravity, but the robot always is. If we have a vector normal to the ground either from the synthesized pseudo-depth map from the original Videodex method or privledged information from an accelerometer (accelerometer is not used in the original Videodex method) we can use:

$$\text{pitch} = \tan^{-1}(x_{Acc}/\sqrt{y_{Acc}^2 + z_{Acc}^2}) \tag{B.1}$$

$$\text{roll} = \tan^{-1}(y_{Acc}/\sqrt{x_{Acc}^2 + z_{Acc}^2}) \tag{B.2}$$

$$\text{theta} = \tan^{-1}(\sqrt{x_{Acc}^2 + y_{Acc}^2}/z_{Acc}) \tag{B.3}$$

The pitch and roll would be used to make the trajectory upright. The yaw is not something that is calculable this way because this rotation is around the z axis, or the direction of gravity so it isn't detected by an accelerometer. The theta represents how far the accelerometer z axis is off from upright but is not useful to reorient the frame.

Accelerometer Robot Reorientation There's book-keeping transformations that must be included to rotate everything into the same frame conventions. Accelerometers, like the one in the GoPro have their frame where Z is up, y is into the screen from the lens, and x is to the left if you're looking at the screen. The camera frame has the x-axis pointing to the right from the screen point of view, the y-axis facing down, and the z-axis facing out of the lens. The robot frame has its x-axis facing out towards the table, the y-axis faces to the left from the robot point of view, and the z-axis points up. This then leads to the following results. The camera in world frame in roll, pitch, yaw using fixed axis is: $[pitch, 0, -roll]$. The world frame to robot frame rotation in roll, pitch, yaw using fixed axis is $[-3.14/2, 0, -3.14/2]$ This is used to rotate the trajectories to the robot frame and is the rotation component of T_{Robot}^{World}.

Rescaling for Robot We must fit the trajectories from the human videos into the robot frame. The robot frame has significant workspace limits that the human does not have. Even if the human arm is smaller than the robot's, the human can walk around whereas the robot arm cannot move from the middle of the table. We therefore center the trajectory and

ensure it fits in the robot frame. This is the scaling portion of T_{Robot}^{World}.

We rescale each dimension of the arm trajectory as:

$$M_{World}^{Wrist_N} = M_{World}^{Wrist_N} - (\max(M_{World}^{Wrist_{1..N}}) + \min(M_{World}^{Wrist_{1..N}}))/2 + \text{robotWorkspaceCenter}$$

We would like to generate more similar trajectories to use in possible data augmentation. The naive method is to add gaussian noise to the trajectory. While this can be valid, it adds noise to an already noisy system. Instead we leverage the coordinate frames to create more accurate trajectories. We randomize the workspace scaling that is used by 10 percent. Additionally, we create a rotation M_{World}^{World} that rotates the initial world frame by up to 10 degrees in each fixed axis in roll pitch yaw convention. This perturbs the direction that the robot moves in its frame.

While this augmentation can be helpful with lower amounts of internet data, in our results it was not used as it led to similar results to not using data augmentation.

We interpolate the length of the trajectories using RBF basis functions. All trajectories from the internet data are rescaled to 200 datapoints. This uniformity enables efficient batch training and was used for all of the results.

Hand Re-targeting We use a similar approach to retargerting as Sivakumar et al. [216] and Handa et al. [109]. Specifically, we use the the detected human hand poses using MANO Romero et al. [199] (and FrankMocap Rong et al. [200]) to match 3D keypoints between human hands and the allegro hand. Given human hand parameters (β, θ), the goal is to minimize the difference between human and robot keypoints: Human v_i^h and robot v_i^r. The robot keypoints are a function of robot joint pose: q. This is done by the implicit energy function (c_i are scale hyperparameters):

$$E_\pi(\ (\beta_h, \theta_h),\ q\ ) = \sum_i ||v_i^h - (c_i \cdot v_i^r)||_2^2 \tag{B.4}$$

This is inefficient to compute in real time, thus similarly to Sivakumar et al. [216], we distill this into a single neural network

$$f_{\text{hand}}((\beta_h, \theta_h)) = \hat{q}$$

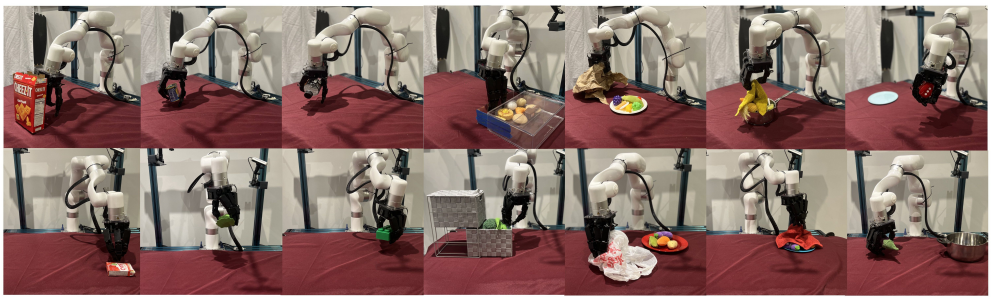

Figure B.1: **Task Images**. A more detailed look at the tasks completed by VideoDex: push, pick, rotate, open, cover, uncover and place. and our website at https://videodex.github.io for further details.

	Pick		Rotate		Open		Cover		Uncover		Place		Push	
	train	test	train	test	train	test	train	test	train	test	train	test	train	test
BC-NDP [44]	0.64 ± 0.11	0.38 ± 0.13	0.94 ± 0.06	0.56 ± 0.13	0.90 ± 0.10	0.60 ± 0.16	0.78 ± 0.15	0.58 ± 0.15	0.88 ± 0.13	0.82 ± 0.12	0.70 ± 0.15	0.35 ± 0.11	1.00 ± 0.00	0.71 ± 0.13
BC-Open[83]	0.50 ± 0.12	0.44 ± 0.13	0.72 ± 0.11	0.38 ± 0.13	0.80 ± 0.13	0.40 ± 0.16	0.44 ± 0.18	0.58 ± 0.15	1.00 ± 0.00	0.91 ± 0.09	0.40 ± 0.16	0.25 ± 0.10	1.00 ± 0.00	0.93 ± 0.07
BC-RNN [83]	0.56 ± 0.12	0.31 ± 0.12	0.78 ± 0.10	0.50 ± 0.13	0.90 ± 0.10	0.50 ± 0.17	0.56 ± 0.18	0.42 ± 0.15	0.88 ± 0.13	0.75 ± 0.13	0.70 ± 0.15	0.50 ± 0.11	1.00 ± 0.00	1.00 ± 0.00
VideoDex	0.81 ± 0.09	0.75 ± 0.11	0.89 ± 0.08	0.69 ± 0.12	0.90 ± 0.10	0.80 ± 0.13	0.78 ± 0.15	0.67 ± 0.14	1.00 ± 0.00	0.90 ± 0.10	0.90 ± 0.10	0.70 ± 0.11	1.00 ± 0.00	1.00 ± 0.00

Table B.3: We present the variance of train objects and test objects for Videodex and baselines described above. See the main paper for the mean results.

This network learns to minimize the energy function E_π described above and is trained by observing internet videos. The hand retargeting setup can be seen in Figures 3 and 4 of the main paper.

Task

The tasks that were completed are Pick, Rotate, Open, Cover, Uncover, Place, Push. In pick, the task is to pickup objects off the table, or a plate/pan. Rotate involves turning an object in place. Open involves opening a drawer. Cover and uncover involve putting on or removing some form of cloth (dish, rubber, paper or plastic) on or from a plate. For place, the robot has to pickup an object and drop it in a plate or pot/pan. For push, the robot has to poke the object with its fingers. These tasks can be seen in Figure B.1. We used videos from Epic Kitchens [78] that were as close as possible to these tasks, and doing similar types of actions. More details can be found in Table B.4.

B.4 Learning Pipeline Details

Learning Setup For our approach, we use the ResNet18 from R3M [178] weights as the visual backbone. This produces an intermediate feature vector of size 512. This is processed

Parameter	Value
Learning Rate	1×10^{-3}
Batch Size	32
Training Demonstrations Per Task	120-175
Human Videos Per Task	350 (Cover/Unicver, Rotate, Push) - 2500 (Open, Pick, Place)
Trajectory Length Human Videos	200 (rescaled)
NDP [43] Basis Functions N	300
NDP [43] Global Parameter α	15

Table B.4: Parameter List

with a 2 layer MLP with a hidden dimension of 512. The visual features are concatenated with the starting hand and wrist pose. We employ two such MLPs, one for the hand and wrist trajectories. These are then processed with an NDP [43]. The NDP processes the input with a single hidden layer to project it into the desired size (parameters W and g). For more information we point the readers to Bahl et al. [43]. We use the implementation from Dasari et al. [83]. We use standard data augmentations from Pytorch. Specifically, we use RandomResizeCrop from a scale of 0.8 to 1.0. We use RandomGrayscale with a probability of 0.05 We use ColorJitter with a brightness of 0.4, contrast of 0.3, saturation of 0.3 and hue of 0.3. Finally, we normalize the RGB values around the typical mean and standard deviation for color images: $\mu = (0.485, 0.456, 0.406)$ $\sigma = (0.229, 0.224, 0.225)$. For different baselines, we used the same backbone (R3M [178]) as our method. We use the same architecture style as well, with the visual features being processed by both a wrist and hand stream. Finally, all network sizes are the same or very similar. We describe our hyperparameters in Table B.4.

B.5 Experimental Setup

We collect data using a dexterous hand robotic teloperation setup [109, 216]. A trained operator stands in front of the camera within view of the robot and operates the system in real-time to collect demonstrations with a trained, uniform style. Another manager stands by. The goal of this manager is to place items on the table for manipulation, randomize locations and types of objects, to start and stop demonstrations for the operator and manage the robot system. We collect about 120-175 demonstrations per task. See table B.2 for details.

Task	Robot Demos	Objects
Pick	125	8
Rotate	140	8
Open	120	4
Cover	124	12
Uncover	145	12
Place	175	10
Push	136	14

Table B.5: Left: Number of trajectories we used for each task. Robot data is collected locally using teleportation. Most of these trajectories are 5-15 seconds in length and capture the motion trajectory of the task and visual data. Right: The number of different objects we used for each task's data collection. In our testing, we show generalization outside of this set of objects.

B.6 Hardware Details

Our hardware setup consists of an LEAP 16 DOF Hand and an XArm 6 manipulator (from Ufactory). The hand is mounted on the wrist of the XArm. To collect data we use a similar teleoperation system as provided by Sivakumar et al. [216] and Handa et al. [109]. We use Intel Realsense D415 cameras to collect human teleoperation and robot videos. We use four NVIDIA RTX 2080TI's for training the policy and running the teleoperated system. We experiment with the Allegro Hand and use it to collect some teleoperated demonstration data, but we find it to be very unreliable and break many times. The motors also quickly overheat and are weak in practice. Therefore, to alleviate these issues we use the LEAP Hand for collecting the final results.

External Codebases

We use the following different external codebases for our pipeline:

- Human body and hand detection: FrankMocap [200], https://github.com/facebookresearch/frankmocap
- NDP and Behavior Cloning [83] code from https://github.com/AGI-Labs/

robot_baselines

- Code for R3M [178] https://github.com/facebookresearch/r3m
- CQL baseline from Takuma Seno [224] (https://github.com/takuseno/d3rlpy)
- COLMAP from Schönberger et al. [203] (https://github.com/colmap/colmap)
- ORBSLAM3 from Campos et al. [61] (https://github.com/UZ-SLAMLab/ORB_SLAM3)
- GoPro Metadata Extractor from (https://github.com/JuanIrache/gpmf-extract)
- Rigid Transform class from (https://github.com/BerkeleyAutomation/autolab_core/blob/master/autolab_core/rigid_transformations.py)
- PnP from (https://opencv.org/)

Appendix C

Experimental Details for DEFT

C.1 Video Demo

We provide video demos of our system at https://dexterous-finetuning.github.io.

C.2 DASH: Dexterous Anthropomorphic Soft Hand

Recently introduced, DASH (Dexterous Anthropomorphic Soft Hand) [166] is a four-fingered anthropomorphic soft robotic hand well-suited for machine learning research use. Its human-like size and form factor allow us to retarget human hand grasps to robot hand grasps easily as well as perform human-like grasps. Each finger is actuated by 3 motors connected to string-like tendons, which deform the joints closest to the fingertip (DIP joint), the middle joint (PIP joint), and the joint at the base of the finger (MCP joint). There is one motor for the finger to move side-to-side at the MCP joint, one for the finger to move forward at the MCP joint, and one for PIP and DIP joints. The PIP and DIP joints are coupled to one motor and move dependently. While the motors do not know the end-effector positions of the fingers, we learn a mapping function from pairs of motor angles and visually observed open-loop finger joint angles. These models are used to command the finger joint positions learned from human grasps.

C.3 MANO Retargeting

For MANO parameters, the axis of each of the joints is rotation aligned with the wrist joint and translated across the hand. However, our robot hand operates on forward and side-to-side joint angles. To translate the MANO parameters to the robot fingers we extract the anatomical consistent axes of MANO using MANOTorch. Once these axes are extracted, each axis rotation represents twisting (not possible for human hands), bending, and spreading. We then match these axes to the robot hand. The spreading of the human hand's fingers (side-to-side motion at the MCP joint) maps to the side-to-side motion at the robot hand's base joint. The forward folding at the base of the human hand (forward motion at the MCP joint) maps to the forward motion at the base of the robot hand's finger. Finally, the bending of the other two finger joints on the human hand, PIP and DIP, map to the robot hand's PIP and DIP joints. While the thumb does not have anatomically the same structure, we map the axes in the same way. Other approaches rely on creating an energy function to map the human hand to the robot hand. However, because the soft hand is similar in anatomy and size to a human hand, it does not require energy functions for accurate retargeting.

C.4 Affordance Model Training

We use data from Ego4D [102], EpicKitchens-100 [78], and HOI4D [157]. After filtering for clips of sufficient length, clips that involve grasping objects with the right hand, and clips that have language annotations, we used 64666 clips from Ego4D, 9144 clips from EpicKitchens, and 2707 clips from HOI4D. In total, we use a dataset of 76517 samples for training our model.

For our contact location model, we use the visual encoder from [178] to encode the image as a 512-dimensional vector. We use the spatial features of the encoder to upsample the latent before applying a spatial softmax to return the contact heatmap. This consists of three deconvolutional layers with 512, 256, and 64 channels in that order.

To predict wrist rotation and grasp pose, we use the language encoder from [194] to compress the language instruction to a 512-dimensional vector. We concatenate the visual and language latents and pass them through a transformer with eight heads and six self-attention layers. We pass the result of the transformer through an MLP with hidden size 576,

and predict a vector of size 48: the first 3 dimensions are the axis-angle rotations; the last 45 dimensions are the joint angles of the hand. These correspond to the parameters output by Frankmocap [200], which we used to get ground truth hand pose in all the datasets. The images used from the training datasets as well as the ground truth labels are released here.

We jointly optimize the L2 loss of the contact location μ, the wrist rotation θ_{wrist} and grasp pose P. The weights we used for the losses are $\lambda_\mu = 1.0, \lambda_\theta = 0.1, \lambda_P = 0.1$. We train for 70 epochs with an initial learning rate of 0.0002, and a batch size of 224. We used the Adam optimizer [129] with cosine learning rate scheduler. We trained on a single NVIDIA RTX A6000 with 48GB RAM.

C.5 Fine-Tuning Parameters

Below are the values of the parameters used for the CEM phase of VideoDex.

Parameter	Value	Description
E	10	Number of elites for CEM
M	10	Number of warm-up episodes
N	30	Total number of CEM episodes
σ_μ	0.02	Initial contact location Standard Deviation (meters)
$\sigma_{\theta_{\text{wrist}}}$	0.2	Initial wrist rotation Standard Deviation (euler angle radians)
σ_P	0.05	Initial soft hand joints Standard Deviation

Table C.1: Values for fixed parameters in fine-tuning Algorithm 2.

C.6 Success Criteria for Tasks

We define the criteria for success in each of our 9 tasks as follows:

- Pick Cup: Cup must leave table surface and stay grasped throughout trial.
- Pour Cup: Cup must be grasped throughout trial and also rotate so that the top of the cup is at a lower height than the base.
- Open Drawer: Drawer is initially slightly open so that it can be grasped. By the end of the episodes, the drawer should be at least 1 centimeter more open than it was at the beginning.
- Pick Spoon: The spoon must not be in contact with the table at the end of the trial.

- Stir Spoon: The spoon base must rotate around the jar/pot at least 180 degrees while grasped.
- Scoop Grape: The spoon must hold a grape at the end of the trial while being held by the soft hand.
- Pick Grape: All grapes must be held by the hand above the table surface. In particular, if any single grape falls due to a weak stem, this is considered a failure.
- Flip Bagel: The side of the bagel that is facing up at the end of the trial should be opposite the side facing up at the beginning.
- Squeeze Lemon: The lemon should be grasped securely on top of the juicer.

Bibliography

[1] Haptx. https://haptx.com/.

[2] Ninjatek ninjaflex edge. https://ninjatek.com/shop/edge/.

[3] Shadowhand. https://www.shadowrobot.com/.

[4] Unitree a1. https://www.unitree.com/en/a1.

[5] Allegro hand. https://www.wonikrobotics.com/research-robot-hand.

[6] Asimo by honda. https://asimo.honda.com/.

[7] Boston dyanmics atlas. https://bostondynamics.com/atlas/.

[8] Clone robotics. https://www.clonerobotics.com/.

[9] Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/.

[10] Dex hand. https://www.dexhand.org/.

[11] Robotis dynamixel. https://www.robotis.us/dynamixel-xc330-m288-t/. Accessed on 2022-11-26.

[12] Creality ender 5 3d printer. https://www.creality.com/.

[13] Franka panda. https://www.franka.de/.

[14] Unitree go1. https://www.unitree.com/en/go1.

[15] Healthline. https://www.healthline.com/health/average-hand-size#adults. Accessed on 2022-11-29.

[16] Inmoov hand. https://inmoov.fr/.

[17] Manus. https://www.manus-meta.com, note=Accessed on 2022-11-28.

[18] Ninjaflex edge. https://ninjatek.com/shop/edge/. Accessed on 2022-11-26.

[19] Oculus rift. https://www.oculus.com/rift/.

[20] Tesla optimus block sorting. https://youtu.be/D2vj0WcvH5c?si=KVGRy4Hvx0YhcZb3.

[21] Pneuflex tutorial. https://www.robotics.tu-berlin.de/menue/software_tutorials/pneuflex_tutorial/.

[22] Reprap open-source 3d printer. https://www.reprap.org/wiki/RepRap.

[23] Robotis dynamixels. https://www.robotis.us/.

[24] Steam vr. https://www.steamvr.com/en/. Accessed on 2023-02-03.

[25] Ur5. https://www.universal-robots.com/products/ur5-robot/.

[26] Htc vive. https://www.vive.com/.

[27] xarm6 by ufactory. https://www.ufactory.cc/xarm-collaborative-robot, .

[28] Ufactory xarm6. https://www.ufactory.cc/product-page/ufactory-xarm-6, . Accessed on 2023-02-03.

[29] Sylvain Abondance, Clark B Teeple, and Robert J Wood. A dexterous soft robotic hand for delicate in-hand manipulation. *IEEE Robotics and Automation Letters*, 5 (4):5502–5509, 2020.

[30] Karen E. Adolph and Sarah E. Berger. *Motor Development*, chapter 4. John Wiley Sons, Ltd, 2007. ISBN 9780470147658. doi: https://doi.org/10.1002/9780470147658.chpsy0204. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470147658.chpsy0204.

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

[32] 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.

[33] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *NIPS*, 2016.

[34] 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.

[35] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *NeurIPS*, 2017.

[36] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[37] Dafni Antotsiou, Guillermo Garcia-Hernando, and Tae-Kyun Kim. Task-oriented hand motion retargeting for dexterous manipulation imitation. In *Proceedings of the*

European Conference on Computer Vision (ECCV) Workshops, pages 0–0, 2018.

[38] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. *arXiv preprint arXiv:2203.13251*, 2022.

[39] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation, 2022. URL https://arxiv.org/abs/2203.13251.

[40] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation, 2022. URL https://arxiv.org/abs/2203.13251.

[41] Haruhiko Asada and J-JE Slotine. *Robot analysis and control*. John Wiley & Sons, 1991.

[42] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *NeurIPS*, 2020.

[43] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *NeurIPS*, 2020.

[44] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Hierarchical neural dynamic policies. *RSS*, 2021.

[45] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *RSS*, 2022.

[46] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. 2023.

[47] Dominik Bauer, Cornelia Bauer, Arjun Lakshmipathy, Roberto Shu, and Nancy S Pollard. Towards very low-cost iterative prototyping for fully printable dexterous soft robotic hands. In *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, pages 490–497. IEEE, 2022.

[48] Bhardwaj, Mohak and Sundaralingam, Balakumar and Mousavian, Arsalan and Ratliff, Nathan and Fox, Dieter and Ramos, Fabio and Boots, Byron. STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[49] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021.

[50] Raunaq Bhirangi, Tess Hellebrekers, Carmel Majidi, and Abhinav Gupta. Reskin: versatile, replaceable, lasting tactile skins. *CoRL*, 2021.

[51] Raunaq Bhirangi, Abigail DeFranco, Jacob Adkins, Carmel Majidi, Abhinav Gupta, Tess Hellebrekers, and Vikash Kumar. All the feels: A dexterous hand with large

area sensing. *arXiv preprint arXiv:2210.15658*, 2022.

[52] Aude G Billard, Sylvain Calinon, and Florent Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54 (5):370–384, 2006.

[53] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016. URL https://arxiv.org/abs/1604.07316.

[54] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[55] Samarth Brahmbhatt, Cusuh Ham, Charles C. Kemp, and James Hays. ContactDB: Analyzing and predicting grasp contact via thermal imaging. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2019.

[56] Paul W Brand. Clinical mechanics of the hand. In *Hand Rehabilitation in Occupational Therapy*, pages 183–184. Routledge, 2012.

[57] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[58] Tina Bruce. *Learning through play, for babies, toddlers and young children*. Hachette UK, 2012.

[59] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005. doi: 10.1080/2151237X.2005.10129202.

[60] Jörg Butterfaß, Markus Grebenstein, Hong Liu, and Gerd Hirzinger. Dlr-hand ii: Next generation of a dextrous robot hand. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 109–114. IEEE, 2001.

[61] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[62] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[63] Zhe Cao, Ilija Radosavovic, Angjoo Kanazawa, and Jitendra Malik. Reconstructing hand-object interactions in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12417–12426, 2021.

[64] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019.

[65] Ilaria Cerulo, Fanny Ficuciello, Vincenzo Lippiello, and Bruno Siciliano. Teleoperation of the schunk s5fh under-actuated anthropomorphic hand using human hand motion tracking. *Robotics and Autonomous Systems*, 89:75–84, 2017.

[66] Maxime Chalon, Markus Grebenstein, Thomas Wimböck, and Gerd Hirzinger. The thumb: Guidelines for a robotic design. In *2010 IEEE/RSJ international conference on intelligent robots and systems*, pages 5886–5893. IEEE, 2010.

[67] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. DexYCB: A benchmark for capturing hand grasping of objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[68] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.

[69] Feifei Chen and Michael Yu Wang. Design optimization of soft robots: A review of the state of the art. *IEEE Robotics & Automation Magazine*, 27(4):27–43, 2020.

[70] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[71] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. *Conference on Robot Learning*, 2021.

[72] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. *arXiv preprint arXiv:2211.11744*, 2022.

[73] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–

1607. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

[74] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[75] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark. https://github.com/open-mmlab/mmaction2, 2020.

[76] Marco Controzzi, Christian Cipriani, and Maria Chiara Carrozza. *Design of Artificial Hands: A Review*, pages 219–246. Springer International Publishing, Cham, 2014. ISBN 978-3-319-03017-3. doi: 10.1007/978-3-319-03017-3_11. URL https://doi.org/10.1007/978-3-319-03017-3_11.

[77] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2020.

[78] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[79] Charles Darwin. On the origin of species. 1859.

[80] Neha Das, Sarah Bechtle, Todor Davchev, Dinesh Jayaraman, Akshara Rai, and Franziska Meier. Model-based inverse reinforcement learning from visual demonstrations. *arXiv preprint arXiv:2010.09034*, 2020.

[81] Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2634–2641, 2013.

[82] Sudeep Dasari, Jianren Wang, Joyce Hong, Shikhar Bahl, Yixin Lin, Austin Wang, Abitha Thankaraj, Karanbir Chahal, Berk Calli, Saurabh Gupta, David Held, Lerrel Pinto, Deepak Pathak, Vikash Kumar, and Abhinav Gupta. Rb2: Robotic manipulation benchmarking with a twist. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/3988c7f88ebcb58c6ce932b957b6f332-Paper-round2.pdf.

[83] Sudeep Dasari, Jianren Wang, Joyce Hong, Shikhar Bahl, Yixin Lin, Austin S Wang, Abitha Thankaraj, Karanbir Singh Chahal, Berk Calli, Saurabh Gupta, et al. Rb2: Robotic manipulation benchmarking with a twist. In *NeurIPS Datasets and Benchmarks Track (Round 2)*, 2021.

[84] Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration. *arXiv preprint arXiv:2008.07682*, 2020.

[85] Roger Davies. Technology versus terrorism. *Jane's International Defence Review*, pages 36–43, 2001.

[86] Raphael Deimel, Patrick Irmisch, Vincent Wall, and Oliver Brock. Automated co-design of soft hand morphology and control strategy for grasping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1213–1218. IEEE, 2017.

[87] Cosimo Della Santina, Cristina Piazza, Giorgio Grioli, Manuel G Catalano, and Antonio Bicchi. Toward dexterous manipulation with augmented adaptive synergies: The pisa/iit softhand 2. *IEEE Transactions on Robotics*, 34(5):1141–1156, 2018.

[88] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[89] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[90] Christian Duriez and Thor Bieze. Soft robot modeling, simulation and control in real-time. In *Soft Robotics: Trends, Applications and Challenges: Proceedings of the Soft Robotics Week, April 25-30, 2016, Livorno, Italy*, pages 103–109. Springer, 2017.

[91] Yahya Elsayed, Augusto Vincensi, Constantina Lekakou, Tao Geng, CM Saaj, Tommaso Ranzani, Matteo Cianchetti, and Arianna Menciassi. Finite element analysis and design optimization of a pneumatically actuating silicone module for robotic surgery applications. *Soft Robotics*, 1(4):255–262, 2014.

[92] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, pages 961–970, 2015.

[93] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stéphane Cotin. *SOFA: A Multi-Model Framework for Interactive Physical Simulation*, pages 283–321. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-29014-5. doi: 10.1007/8415_2012_125. URL https://doi.org/10.1007/8415_2012_125.

[94] Naishi Feng, Qiurong Shi, Hong Wang, Jiale Gong, Chong Liu, and Zhiguo Lu. A soft robotic hand: design, analysis, semg control, and experiment. *The International Journal of Advanced Manufacturing Technology*, 97:319–333, 2018.

[95] Yao Feng, Vasileios Choutas, Timo Bolkart, Dimitrios Tzionas, and Michael J Black. Collaborative regression of expressive bodies using moderation. *arXiv preprint arXiv:2105.05301*, 2021.

[96] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL https://doi.org/10.1145/358669.358692.

[97] David F Fouhey, Xiaolong Wang, and Abhinav Gupta. In defense of the direct perception of affordances. *arXiv preprint arXiv:1505.01085*, 2015.

[98] Eleanor J Gibson. Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge. *Annual review of psychology*, 39(1):1–42, 1988.

[99] Sam Goldstein, Dana Princiotta, and Jack A Naglieri. Handbook of intelligence. *Evolutionary theory, historical perspective, and current concepts*, 10:978–1, 2015.

[100] Mohit Goyal, Sahil Modi, Rishabh Goyal, and Saurabh Gupta. Human hands as probes for interactive object understanding. In *CVPR*, 2022.

[101] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[102] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

[103] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021.

[104] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017.

[105] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3196–3206, 2020.

[106] Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D Twigg, and Kenrick Kin. Online optical marker-based hand tracking with deep labels. *ACM Transactions on Graphics (TOG)*, 37(4):1–10, 2018.

[107] 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, 2019. URL https://arxiv.org/abs/1910.03135.

[108] 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, 2020. doi: 10.1109/ICRA40945.2020.9197124.

[109] 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.

[110] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv preprint arXiv:2210.13702*, 2022.

[111] Yana Hasson, Gül Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *CVPR*, 2019.

[112] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[113] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[114] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[115] Wenlong Huang, Igor Mordatch, Pieter Abbeel, and Deepak Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021.

[116] Wenlong Huang, Igor Mordatch, Pieter Abbeel, and Deepak Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021.

[117] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 2013.

[118] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36 (7):1325–1339, 2013.

[119] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *ICRA*, 2019.

[120] Lynette A Jones and Susan J Lederman. *Human hand function*. Oxford university press, 2006.

[121] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[122] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[123] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. *CoRR*, abs/1712.06584, 2017. URL http://arxiv.org/abs/1712.06584.

[124] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *ICRA*, 2011.

[125] Jeffrey Randall Kerr. *An analysis of multi-fingered hands: a dissertation*. PhD thesis, Stanford University, 1985.

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

[127] Uikyum Kim, Dawoon Jung, Heeyoen Jeong, Jongwoo Park, Hyun-Mok Jung, Joono Cheong, Hyouk Ryeol Choi, Hyunmin Do, and Chanhun Park. Integrated linkage-driven dexterous anthropomorphic robotic hand. *Nature communications*, 12(1): 1–13, 2021.

[128] Yong-Jae Kim, Junsuk Yoon, and Young-Woo Sim. Fluid lubricated dexterous finger mechanism for human-like impact absorbing capability. *IEEE Robotics and Automation Letters*, 4(4):3971–3978, 2019.

[129] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[130] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[131] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[132] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Advances in neural information processing systems*, 21, 2008.

[133] 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.

[134] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *RSS*, 2021.

[135] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[136] Vikash Kumar and Emanuel Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 657–663, 2015. doi: 10.1109/HUMANOIDS.2015.7363441.

[137] Vikash Kumar, Yuval Tassa, Tom Erez, and Emanuel Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6808–6815. IEEE, 2014.

[138] Dong-Hyuk Lee, Jae-Han Park, Sung-Woo Park, Moon-Hong Baeg, and Ji-Hun Bae. Kitech-hand: A highly dexterous and modularized robotic hand. *IEEE/ASME Transactions on Mechatronics*, 22(2):876–887, 2016.

[139] Jangwon Lee and Michael S Ryoo. Learning robot activities from first-person human videos using convolutional future regression. In *CVPR Workshops*, pages 1–2, 2017.

[140] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML*, 2013.

[141] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

[142] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *ISER*, 2016.

[143] Rui Li, Hongyu Wang, and Zhenyu Liu. Survey on mapping human hand motion

to robotic hands for teleoperation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2647–2665, 2022. doi: 10.1109/TCSVT.2021.3057992.

[144] Shuang Li, Xiaojian Ma, Hongzhuo Liang, Michael Görner, Philipp Ruppel, Bin Fang, Fuchun Sun, and Jianwei Zhang. Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 416–422. IEEE, 2019.

[145] Zexiang Li and Shankar Sastry. Issues in dextrous robot hands. *Dextrous robot hands*, pages 154–186, 1990.

[146] Zexiang Li, JF Canny, and Shankar S Sastry. On motion planning for dextrous manipulation. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 775–780, 1989.

[147] Minas V Liarokapis, Panagiotis K Artemiadis, and Kostas J Kyriakopoulos. Telemanipulation with the dlr/hit ii robot hand using a dataglove and a low cost force feedback device. In *21st Mediterranean Conference on Control and Automation*, pages 431–436. IEEE, 2013.

[148] Klaus Libertus, Amy S Joh, and Amy Work Needham. Motor training at 3 months affects object exploration 12 months later. *Developmental Science*, 19(6):1058–1066, 2016.

[149] Colin M Light, Paul H Chappell, and Peter J Kyberd. Establishing a standardized clinical assessment tool of pathologic and prosthetic hand function: normative data, reliability, and validity. *Archives of physical medicine and rehabilitation*, 83(6): 776–783, 2002.

[150] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

[151] Changliu Liu and Masayoshi Tomizuka. Designing the robot behavior for safe human robot interactions. In *Trends in Control and Decision-Making for Human-Robot Collaboration Systems*, pages 241–270. Springer, 2017.

[152] Jia Liu, Fangxiaoyu Feng, Yuzuko C. Nakamura, and Nancy S. Pollard. A taxonomy of everyday grasps in action. *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 573–580, 2014.

[153] Jia Liu, Fangxiaoyu Feng, Yuzuko C Nakamura, and Nancy S Pollard. A taxonomy of everyday grasps in action. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 573–580. IEEE, 2014.

[154] Jia Liu, Fangxiaoyu Feng, Yuzuko C. Nakamura, and Nancy S. Pollard. A taxonomy of everyday grasps in action. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 573–580, 2014. doi: 10.1109/HUMANOIDS.2014.7041420.

[155] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *CVPR*, 2022.

[156] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21013–21022, June 2022.

[157] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21013–21022, June 2022.

[158] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[159] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[160] Raymond R Ma and Aaron M Dollar. On dexterity and dexterous manipulation. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 1–7. IEEE, 2011.

[161] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[162] 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.

[163] Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning (CoRL)*, 2021.

[164] Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning*, pages 651–661. PMLR, 2022.

[165] Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning*, pages 651–661. PMLR, 2022.

[166] Pragna Mannam, Kenneth Shaw, Dominik Bauer, Jean Oh, Deepak Pathak, and Nancy Pollard. A framework for designing anthropomorphic soft hands through interaction, 2023.

[167] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

[168] Roberto Martin-Martin, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. *IROS*, 2019.

[169] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Alan: Autonomously exploring robotic agents in the real world. *arXiv preprint arXiv:2302.06604*, 2023.

[170] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[171] Andrew S Morgan, Kaiyu Hang, Bowen Wen, Kostas Bekris, and Aaron M Dollar. Complex in-hand manipulation via compliance-enabled finger gaiting and multimodal planning. *IEEE Robotics and Automation Letters*, 7(2):4821–4828, 2022.

[172] Mustafa Mukadam, Xinyan Yan, and Byron Boots. Gaussian process motion planning. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 9–15. IEEE, 2016.

[173] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[174] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *ICCV*, 2019.

[175] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[176] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *NeurIPS*, pages 9191–9200, 2018.

[177] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[178] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[179] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation, 2021.

[180] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and

generalization of motor skills by learning from demonstration. In *ICRA*, 2009.

[181] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[182] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[183] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

[184] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[185] Jan Peters, Katharina Mülling, and Yasemin Altün. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1607–1612. AAAI Press, 2010.

[186] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016.

[187] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

[188] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

[189] M. Prada, A. Remazeilles, A. Koene, and S. Endo. Dynamic movement primitives for human-robot interaction: Comparison with human behavioral observation. In *International Conference on Intelligent Robots and Systems*, 2013.

[190] Steffen Puhlmann, Jason Harris, and Oliver Brock. RBO hand 3: A platform for soft dexterous manipulation. *IEEE Transactions on Robotics*, 38(6):3434–3449, dec 2022. doi: 10.1109/tro.2022.3156806. URL https://doi.org/10.1109%2Ftro.2022.3156806.

[191] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-Hand Object Rotation via Rapid Motor Adaptation. In *Conference on Robot Learning*

(CoRL), 2022.

[192] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[193] Yuzhe Qin, Hao Su, and Xiaolong Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation, 2022. URL https://arxiv.org/abs/2204.12490.

[194] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

[195] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. *arXiv preprint arXiv:2004.04650*, 2020.

[196] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[197] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[198] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[199] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), November 2017.

[200] 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 (ICCV) Workshops*, pages 1749–1759, October 2021.

[201] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*. Springer, 2006.

[202] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with

interaction. *arXiv preprint arXiv:2011.06507*, 2020.

[203] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[204] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[205] Robert J Schwarz and C Taylor. The anatomy and mechanics of the human hand. *Artificial limbs*, 2(2):22–35, 1955.

[206] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018.

[207] Dandan Shan, Jiaqi Geng, Michelle Shu, and David Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, pages 9869–9878, 2020.

[208] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14), 2021.

[209] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019.

[210] Kenneth Shaw and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *in Submission, ICRA*, 2023.

[211] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *CoRL*, 2022.

[212] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. VideoDex: Learning Dexterity from Internet Videos. In *Conference on Robot Learning (CoRL)*, 2022.

[213] Zilin Si, Tianhong Catherine Yu, Katrene Morozov, James McCann, and Wenzhen Yuan. Robotsweater: Scalable, generalizable, and customizable machine-knitted tactile skins for robots. *arXiv preprint arXiv:2303.02858*, 2023.

[214] Leon Sievers, Johannes Pitz, and Berthold Bäuml. Learning purely tactile in-hand manipulation with a torque-controlled hand. *arXiv preprint arXiv:2204.03698*, 2022.

[215] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[216] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube, 2022.

[217] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: learning a robotic hand imitator by watching humans on youtube. *RSS*, 2022.

[218] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. In *RSS*, 2020.

[219] K. Sohn, X. Yan, and H. Lee. Learning Structured Output Representation using Deep Conditional Generative Models. In *NeurIPS*, 2015.

[220] Balakumar Sundaralingam and Tucker Hermans. Relaxed-rigidity constraints: kinematic trajectory optimization and collision avoidance for in-grasp manipulation. *Autonomous Robots*, 43(2):469–483, 2019.

[221] Subramanian Sundaram, Petr Kellnhofer, Yunzhu Li, Jun-Yan Zhu, Antonio Torralba, and Wojciech Matusik. Learning the signatures of the human grasp using a scalable tactile glove. *Nature*, 569(7758), 2019. doi: 10.1038/s41586-019-1234-z.

[222] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444. IEEE, 2021.

[223] Omid Taheri, Nima Ghorbani, Michael J Black, and Dimitrios Tzionas. Grab: A dataset of whole-body human grasping of objects. In *European Conference on Computer Vision*, pages 581–600. Springer, 2020.

[224] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021.

[225] From One Hand to Multiple Hands: Imitation Learning for Dexterous Manipulation from Single-Camera Teleoperation. Qin, yuzhe and su, hao and wang, xiaolong, 2022.

[226] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012.

[227] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991.

[228] Jean Vertut and Philippe Coiffet. Robot technology. vol. 3a. teleoperation and robotics: evolution and development. 1985.

[229] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. *CoRR*, abs/1804.05653, 2018. URL http://arxiv.org/abs/1804.05653.

[230] Jiayi Wang, Franziska Mueller, Florian Bernard, Suzanne Sorli, Oleksandr Sotnychenko, Neng Qian, Miguel A Otaduy, Dan Casas, and Christian Theobalt. Rgb2hands: real-time tracking of 3d hand interactions from monocular rgb video. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

[231] Xiaolong Wang, Rohit Girdhar, and Abhinav Gupta. Binge watching: Scaling affordance learning from sitcoms. In *CVPR*, 2017.

[232] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[233] Frank R Wilson. *The hand: How its use shapes the brain, language, and human culture*. Vintage, 1999.

[234] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[235] Zhe Xu and Emanuel Todorov. Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3485–3492. IEEE, 2016.

[236] Lixin Yang, Xinyu Zhan, Kailin Li, Wenqiang Xu, Jiefeng Li, and Cewu Lu. CPF: Learning a contact potential field to model the hand-object interaction. In *ICCV*, 2021.

[237] Yufei Ye, Abhinav Gupta, and Shubham Tulsiani. What's in your hands? 3d reconstruction of generic objects in hands. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3895–3905, 2022.

[238] Tsuneo Yoshikawa. Dynamic manipulability of robot manipulators. *Transactions of the Society of Instrument and Control Engineers*, 21(9):970–975, 1985.

[239] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[240] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv preprint arXiv:2008.04899*, 2020.

[241] Shenli Yuan, Austin D Epps, Jerome B Nowak, and J Kenneth Salisbury. Design of a roller-based dexterous hand for object grasping and within-hand manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8870–8876. IEEE, 2020.

[242] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[243] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. *arXiv preprint arXiv:2106.03911*, 2021.

[244] Yu Zhang and Robert K Katzschmann. Creation of a modular soft robotic fish testing platform. *arXiv preprint arXiv:2201.04098*, 2022.

[245] Wenping Zhao, Jinxiang Chai, and Ying-Qing Xu. Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data. In *Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation*, pages 33–42,

2012.

[246] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *ECCV*, 2022.

[247] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Phillip Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. *arXiv preprint arXiv:2201.02605*, 2022.

[248] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019.

[249] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 813–822, 2019.