Designing and Teaching Dexterous Robot Hands

Kenneth Shaw

CMU-RI-TR-26-53

May 2026

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. Abhinav Gupta, *Carnegie Mellon University*
Prof. Jitendra Malik, *University of California, Berkeley*
Dr. Ankur Handa, *Nvidia*

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

Abstract

Consider the simple acts of typing on a keyboard, hammering a nail, or using chopsticks. Our hands combine strength, dexterity, and sensitivity, enabling precise fingertip forces and more than 70 distinct grasping strategies. These capabilities shape how we interact with the world and are deeply tied to the human brain. In fact, some of the development of human intelligence has often been linked to the demands of manipulating the physical world [157, 392].

In robotics, a long-standing goal is to build humanoid robots that can perform everyday household tasks as naturally as humans. Yet robotic manipulation today remains largely limited to simple grippers or suction cups in structured pick-and-place tasks. Achieving general dexterous manipulation remains difficult. Robotic hands still lack the strength, sensing, and adaptability of human hands, and building the "robot brain" needed to control them across diverse tasks and environments remains an open challenge.

On the hardware side, existing robot hands are often costly, difficult to obtain, and hard to use reliably. To address this, I introduce a family of LEAP hands designed for machine learning research. These hands are dexterous, affordable, easy to use, and simple to build and customize. My joint-driven design is straightforward to manufacture, modify, and simulate, and has already influenced many robot hand designs in the field. My tendon-driven design, inspired by the placement of human muscles in the forearm, uses 3D-printed compliant rigid-soft fingers that make the hands more human-like in their interactions with the world. This compliance improves robustness under uncertain contact while preserving useful strength and remaining easy to fabricate. Together, these robot hands provide practical, accessible, and open-source hardware for advancing dexterity.

Robot hands must also learn to use their dexterity in a human-like way. For example, a robot should grasp a knife by the handle rather than the blade. This is challenging because a humanoid with two hands has over 50 degrees of freedom, many possible contact points and must behave like humans do. To address this, the key approach is to teach robot hands similar to how babies learn by watching others and interacting with the environment. First, since robot hands share a similar structure with human hands, they can learn by watching human videos. By retargeting human motion into robot hand motion, this data can teach robot hands. They can even continue improving through real-world experience. I also build motion-capture-based teleoperation systems, including bimanual systems such as BiDex, to collect higher-quality demonstrations for more difficult tasks. Finally, massively parallel simulation is used to generate geometrically aware dexterous grasps. Together, this thesis takes a full stack approach to study how hardware and human data can make dexterous robot hands more practical and generalizable.

Acknowledgments

This thesis would not have been possible without the incredible community of students in Prof. Deepak Pathak's and Prof. Abhinav Gupta's labs. Deepak's lab has been a truly bustling and inspiring environment, where students are constantly thinking deeply, sharing ideas, challenging one another, and collaborating on new directions. Many of the ideas in this thesis grew out of that community, and I am deeply grateful to everyone who shaped my thinking and contributed to this work. Beyond the research itself, I also formed many lasting friendships and collaborations here, for which I am especially thankful.

I would also like to thank my 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 but the 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 work. Shikhar 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 also always appreciated the late night academic discussions with Russell Mendonca. He has this intense curiosity, enthusiasm, and belief in AI that really taught me so much.

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 many key ideas and problems about robot hands. Also, because he came from a stint in industry, he made great visualizations, figures and code comments that I learned a lot from.

In my later years at CMU, I spent a lot of time working with and advising a whole new cohort of students. I especially want to thank Jason Liu for being a close friend and a great collaborator. Jason Liu has this intense curiosity and pays incredible attention to the details. He knows details about simulation, geometric fabrics and controls that maybe almost nobody in the world also knows. He doesn't just want to build demos, he builds real robots that work. I barely needed to give him advice and I learned so much working with him.

I also want to thank Tony Tao, Yulong Li, Aditya Kannan, and Jim Young for being fantastic MS student collaborators across a wide range of projects. I am also grateful to Ray Liu, Mingxuan Li, Sri Anumakonda, Haoyu Xiong, and Shagun Uppal for being such motivated and hardworking students. Working with all of them taught me a great deal about mentorship, and I truly enjoyed our time working together.

Prof. Nancy Pollard's lab has been a wonderful collaboration. Through working with them, I learned how to think about mechanism design in a much more principled way. I never formally studied mechanical engineering as part of my education, but I've really enjoyed learning about it and getting to work on it. This collaboration also let us take advantage of one of CMU's greatest strengths: the breadth of its robotics research, spanning everything from mechanical engineering to robot learning.

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.

My internship at NVIDIA was my first experience working in industry, and it taught me a great deal about simulation technology and robotics in a company setting. I am especially grateful to Ankur Handa for opening that world to me and for remaining a great friend. Although I may have taken on more than was realistic for a short internship, the experience deeply enriched my PhD journey. I also met Arthur Allshire and Ritvik Singh, incredible collaborators whose expertise in simulation and RL training is truly exceptional, and with whom I am grateful to still stay in touch.

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. They

helped me immensely as an undergrad and instilled that research bug in me. I learned so much from their kind mentorship and I 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 research practices. A lot of my intuitions in hardware are owed to this robotics team.

Of course, I would also like to thank my family. 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, Air Force Office of Scientific Research (AFOSR) FA9550-23-1-0747, AIST CMU grant and Google Research Award.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Take a moment to think about your hands. Just this morning, you probably brushed your teeth, cooked breakfast, poured a cup of coffee, typed on a keyboard, and maybe even did some pullups. These actions feel easy, but they reveal an extraordinary range of ability. Our hands can move smoothly from delicate, fine-grained motions to lifting heavy weights. Our fingers can pinch tightly around a tiny object, stretch around a large bottle, gently hold an egg, and firmly grip a hammer. They can repeat these actions for years while remaining strong, flexible, and remarkably durable. Just as importantly, our hands can feel the world through our fingers and palms. This sense of touch helps us use tools, keep objects from slipping, adjust our grasp, and even identify objects by feel alone.

Human hands are incredible biological ”hardware,” shaped by millions of years of evolution [125]. Each hand contains a carefully arranged system of bones, joints, muscles, tendons, ligaments, nerves, and skin. Our fingers alone contain 14 phalanges, while the wrist and palm include 13 additional bones that provide structure, support, and mobility. These bones are connected by joints such as the MCP, PIP, and DIP joints, which allow the fingers to bend, curl, and adapt to objects of many shapes. Muscles in the hand and forearm pull on tendons attached to the bones to generate motion, while ligaments stabilize the structure and synovial fluid helps the joints move smoothly. At the same time, the radial, median, and ulnar nerves carry signals related to touch, temperature, pain, and pressure through the fingers and palm. Together, this intricate combination of structure, actuation,

sensing, and durability gives the human hand its remarkable capabilities [85, 191, 328].

Of course, this biological hardware is only useful because it is controlled by the human brain, another marvel of evolution. The brain processes visual and tactile information, coordinates many muscles at once, and turns raw sensation into purposeful action. In fact, the development of the hand and the development of human intelligence are deeply connected: our ability to grasp, manipulate, and explore the world helped shape how we learn and think [392]. This process begins early in life. As infants, we explore the world by reaching, touching, and grasping. Within only a few months, babies begin learning how to coordinate their fingers and hold onto objects, and they can also learn quickly by watching demonstrations from parents [88, 156, 229]. Over time, these experiences help us generalize to new objects, tools, and situations, which is a central part of intelligence [46, 157, 392]. Through this combination of exploration and demonstration, humans learn a rich repertoire of everyday hand behaviors, including more than 80 different types of grasps [239].

Given how central hands are to human action and intelligence, it is natural to imagine robots with similar dexterity. Science fiction has imagined this future for decades through humanoid robots that live and work alongside people. C-3PO travels across the galaxy interacting with humans and their environments, while Rosey from The Jetsons was imagined in the 1960s as a household robot capable of helping with everyday chores. These robots are compelling because they suggest that a machine with a human-like body could naturally perform human tasks.

Real robots, however, have developed very differently. Instead of human-like hands, most deployed robot arms use simple two-finger grippers, claws, or suction cups. This has been a practical choice in factories and warehouses, where many tasks involve repeatedly picking up, placing, or moving known objects in structured and closed environments. In these settings, simple end-effectors are often enough: they are easier to build, easier to control, and more reliable than complex hands. As a result, robotics has made progress with tools that are far less dexterous and adaptable than human hands, but well matched to the constrained tasks in limited settings.

Recently, interest has returned to humanoid robots and prosthetic hands that can operate in everyday human environments. In these settings, simple grippers are no longer enough. Homes, workplaces, tools, and everyday objects are designed around human hands, so robots that assist people must be able to grasp, carry, open, pour, twist, press, and manipulate

objects in human-like ways. Early humanoids such as Honda ASIMO offered a glimpse of this future, demonstrating behaviors such as walking, climbing stairs, playing soccer, and pouring a drink with its hands [8]. More recent efforts, including Tesla Optimus and other humanoid platforms, show how much progress has been made in locomotion, balance, and whole-body coordination [29]. Yet the gap remains clear: robots have made striking progress in moving through the world, but far less progress in dexterously interacting with it. For robots to become useful manipulators in everyday life, prosthetics, and other human-centered settings, both the hands themselves and the machine learning systems that control them must become much more capable.

This thesis is about making robot hands more capable. I approach this problem from two directions. First, we need better robot hand hardware. How can we emulate the most important capabilities of the human hand while still building hands that are practical, robust, affordable, and easy for researchers to use? This requires designing hands that are anthropomorphic enough to perform human-like manipulation, strong enough for real tasks, and inexpensive enough to support widespread adoption. Second, we need better methods for controlling these hands. Just as the human brain learns from vision, touch, exploration, and demonstration, robot hands must learn from rich sources of data, including human videos, teleoperation, and real-world experience. Finally, this thesis brings these two directions together. By jointly improving the hands and the learning systems that control them, we push robot manipulation toward more general, fine-grained, and human-like dexterity.

1.2 Designing Dexterous Robot Hands for Machine Learning

Designing a useful robot hand is difficult because the human hand is not just a simple gripper. It has many joints, many possible motions, and a compact structure that allows it to wrap around, pinch, press, push, and hold objects in many different ways. It is also surprisingly strong. Human hands can carry heavy objects, firmly grasp tools, apply large forces, and withstand repeated contact with the world, while still remaining precise enough to pick up something small or fragile. Replicating this combination of dexterity, compactness, strength, and durability in robot hardware has long been considered one of the hardest challenges in

dexterous robotics. Adding more joints can make a hand more expressive, but also makes it harder to build, control, and repair. Making a hand stronger can make it bulkier or more expensive. Making it cheaper can make it weaker or less reliable. Yet if robot hands are to be broadly useful, they must balance all of these requirements at once. How can we design anthropomorphic robot hands that are capable enough for rich manipulation, but still practical enough to build, maintain, and use in real-world research?

Despite decades of work, conventional robot hands have not yet become widely adopted. For many years, the main commercially available dexterous hands have been the Shadow Hand [5] and the Allegro Hand [7]. These systems are powerful research platforms, but they are expensive, costing roughly $150k and $15k respectively, and they remain difficult to use without significant expertise. At the same time, many custom robot hands have shown impressive capabilities within individual labs, but are hard for others to reproduce because they require complex fabrication, specialized assembly, or are not fully open-source [91, 204, 397]. Open-source hands such as InMoov [22] and DexHand [13] lower the barrier to entry, but often make other tradeoffs: they may have too few joints/degrees of freedom, limited strength, or insufficient durability for demanding dexterous manipulation. As a result, researchers have often had to choose between hands that are capable but expensive and difficult to use, or hands that are accessible but not capable enough.

I believe that dexterous manipulation should not be limited to a few labs with expensive hardware and specialized expertise. Instead, robot hands should become as easy to obtain, repair, and use as the simple two-finger grippers that are common in robotics today. To make this possible, a robot hand must be low-cost, robust, repairable, and straightforward to assemble, while still being strong enough to interact with the real world. At the same time, accessibility cannot come at the cost of capability. A useful hand must also be dexterous and anthropomorphic enough to grasp diverse objects, use tools, and support a wide range of manipulation tasks. The challenge is to build hands that are not merely impressive demonstrations, but practical platforms that many researchers can actually use.

Following this belief, I introduce LEAP Hand: a dexterous robot hand and learning platform designed to make capable manipulation more accessible. LEAP Hand costs under $2k, is 3D printable, straightforward to assemble, and is released with simulation, teleoperation tools, and software for robot learning. These tools provide the foundations needed to study learning from demonstrations, human videos, and real-world experience. Rather than treating the hand as a one-off hardware prototype, we present it as a complete

platform for studying dexterous manipulation, with additional details available on the project website at https://leaphand.com. The goal is to lower the barrier to dexterous manipulation research while preserving the strength, dexterity, and human-like structure needed for real-world tasks.

The impact of LEAP Hand comes not only from the hand itself, but from the community that has grown around it. Since its release, LEAP Hand has been adopted by students and researchers around the world for work on teleoperation, learning from human video, and sim2real on tasks such as tool use, human-like grasping, in-hand reorientation. For many researchers, it provides a first practical entry point into dexterous manipulation: a hand that is affordable enough to obtain, simple enough to assemble, and capable enough to support meaningful research. Around this platform, we have built an ecosystem of shared tools, documentation, and experience. LEAP Hand shows that dexterous manipulation can be accessible to all.

However, accessible hardware is only the first step. A robot hand is not dexterous simply because it has many joints or a human-like shape. It also needs a way to be controlled, taught, and improved from data. Once a capable hand can be built by many people, the next question becomes: how can people actually use it to teach dexterous behavior? Teleoperation provides a natural bridge between hardware and learning because it allows humans to directly demonstrate how the hand should move, grasp, and interact with objects. But if teleoperation requires expensive motion-capture systems, specialized gloves, or careful laboratory setup, then the same accessibility problem reappears in a new form.

This motivates Robotic Telekinesis, a teleoperation system that controls a dexterous robot hand from a single RGB webcam. The goal is to make teaching a robot hand as accessible as building one. Instead of relying on specialized sensing hardware, Robotic Telekinesis uses visual models trained from large-scale human data to interpret hand motion and guide the robot toward useful behavior. In this way, LEAP Hand provides the accessible hardware foundation, while Robotic Telekinesis begins to address the learning and data-collection problem that follows.

More broadly, Robotic Telekinesis points to a central theme of this thesis: dexterous robots should learn from the everyday behavior that humans already produce. We should not need to collect robot demonstrations for every task, object, and environment a robot may encounter. Human hand behavior already exists at enormous scale in the world, especially in internet video, where people grasp, carry, use, and manipulate objects across diverse

tasks and settings. If robot hands can learn from this data, then human video can become a scalable source of experience for dexterous manipulation.

Learning from arbitrary human behavior is appealing, but it also introduces a central challenge: humans and robots do not have the same bodies. This embodiment gap makes it difficult to directly convert human motion into robot action. Because our robot hands are designed to be anthropomorphic, this gap is smaller, but it does not disappear. We still need retargeting methods that can translate human hand motion into actions that a robot hand can physically execute. A key insight of this thesis is that this translation can itself be learned from human videos and internet-scale human experience. This allows robot hands to benefit from noisy, diverse, and imperfect human behavior in the wild.

VideoDex builds on this idea by turning internet videos into robot experience. It extracts robot trajectories from human hand motion and treats them as pseudo-robot demonstrations, which can be used to pretrain dexterous policies before real-world learning. In this way, human videos do not replace robot experience, but they provide a powerful prior: they teach the robot what useful hand behavior should look like before the robot must learn the task through its own limited data.

1.3 Compliance Enables Robust Real-World Learning

LEAP Hand was my first step toward accessible, open-source robot hands. It showed that a dexterous hand could be low-cost, easy to build, and useful for robot learning. However, LEAP Hand is still a relatively rigid and bulky mechanism. This is partly because its motors are placed directly in the finger joints, which makes the hand simple, strong, and easy to control, but also makes it difficult to match the compactness, softness, and adaptability of a human hand. Human hands are very different. They are not rigid mechanisms filled with motors. Their skin, tendons, joints, ligaments, and soft tissue allow them to comply with the environment: the hand can conform to objects, distribute contact forces, absorb small errors, and adjust naturally as contact changes. This matters because real-world manipulation is full of uncertainty. Tools have curved handles, cups shift in the hand, soft objects deform, and small errors in contact can change the outcome of a task. A rigid hand must often place each finger precisely and apply the right force at the right moment, while a compliant hand can adapt through contact itself. At the same time, human hands are also remarkably strong and durable. Much of this strength comes from muscles in the forearm, which transmit

force through tendons into the fingers, while their compliant joints and soft tissues help them withstand forces from many directions. The human hand therefore suggests a new direction for robot hand design: moving beyond rigid, joint-driven mechanisms toward hands that combine human-like compliance, tendon-driven strength, and actuation placed outside the finger joints.

This motivates DASH Hand, a smaller tendon-driven soft hand designed for compliant dexterous interaction. The goal of DASH is not only to make another accessible robot hand, but to study how softness and anthropomorphic structure can make a hand more robust in the physical world. Instead of requiring every contact to be exact, a compliant hand can bend, settle, and adapt around objects during interaction. DASH remains relatively easy to fabricate and assemble, but its tendon-driven, compliant fingers allow the hand to better adjust to uncertain contact. Rather than designing the hand only through static analysis, we develop DASH through interaction: building new hand iterations, teleoperating them on real manipulation tasks, observing where they succeed and fail, and using those observations to improve the design. In this way, DASH explores how hand morphology, compliance, tendon actuation, and real-world contact together shape dexterous capability.

Once the hand can physically adapt to objects and comply with the environment, the next question is how to use that robustness for learning. Real-world fine-tuning requires repeated physical interaction: the robot must try grasps, make contact, fail, recover, and try again. With a rigid or fragile hand, these mistakes can quickly become damaging. The hand may collide with the table, push too hard against an object, or apply forces in unexpected directions. A compliant hand makes this process more practical because it can absorb contact, protect itself and the environment, and continue collecting useful experience.

DEFT builds on this DASH Hand by asking how a compliant dexterous hand can improve through real-world experience. It starts with an affordance prior learned from freely available human data on the internet, which gives the robot an initial guess about useful contact: where on the object to grasp, how the hand should approach, and what kind of grasp is likely to support the task. This human-like prior is especially important for tools, where success depends not only on reaching the object, but on holding the right part in the right way. DEFT then refines this initial human behavior through autonomous rollouts in the real world, continually improving the policy through physical interaction. DASH provides the compliant and durable hand needed to execute these repeated interactions safely and robustly, while DEFT provides the learning procedure that turns experience into

better behavior. In this way, compliance is not only useful for manipulation itself; it makes sustained real-world learning with dexterous hands practical. This process emulates how babies learn, by learning by watching and then repeatedly interacting with the world.

Together, DASH and DEFT form a single progression. DASH asks how to build a soft anthropomorphic hand that can adjust to the physical world through compliance. DEFT then asks how that hand can use human data and repeated real-world interaction to improve its behavior. The broader lesson is that compliance is not just a hardware detail, it changes what kind of learning can be done in the real world. A hand that can conform to objects, tolerate uncertain contact, and avoid brittle failures provides a better substrate for fine-tuning policies through physical experience.

1.4 Unlocking Fine-Grained Dexterity

However, even with better hardware, compliance, and human data, something is still missing. Human dexterity is not just the ability to close a hand around an object. People fluidly switch between power grasps, precision grasps, hook grasps, lateral pinches, tripod grasps, and many other forms of contact depending on the object, the task, and the stage of interaction. We grasp a hammer differently from a cup, a cup differently from a credit card, and a credit card differently from chopsticks. Even within a single task, the grasp may change as the hand reaches for an object, secures it, uses it, adjusts it, and releases it. On the hardware side, the hand must be both strong, compliant, and human-like enough to realize the required contacts, while the policy must learn advanced behavior on where to place the fingers, how to maintain contact, and how to adapt as the object and task evolve. This motivates the final direction of the thesis: combining the principles of hand design, compliance, teleoperation, and learning from human data to pursue fine-grained dexterity.

The chapters in this final part follow this combined approach. One lesson from the human hand is that compliance and rigidity are not opposites; they work together. The hand is soft enough to conform to objects and absorb contact, but it is built around rigid bones that provide strength, structure, and force transmission. This hybrid rigid-soft organization is what allows human hands to be both adaptable and powerful. How can robot hands emulate this principle? In LEAP Hand v2 Advanced and LEAP Hand v2, we revisit the hand itself and push the hardware closer to the capabilities needed for fine-grained manipulation. LEAP Hand v2 Advanced introduces a high-DOF hybrid rigid-soft hand with improved strength,

compliance, palm motion, and a more human-like actuation structure. It is designed to support a rich diversity of grasps. LEAP Hand v2 follows similar hybrid rigid-soft design principles, but emphasizes accessibility: it is a small, extremely low-cost, and extremely easy-to-assemble robot hand designed to make dexterous hand research practical for a wider community. Together, these hands extend the LEAP Hand design philosophy from accessible dexterous manipulation toward stronger, softer, and more human-like interaction.

However, hands with this many degrees of freedom are difficult to control from video alone. To collect high-quality data for complex manipulation, we need data that can capture the richness of fine-grained human hand and arm motion. In BiDex, we scale dexterous data collection through low-cost motion-capture-based bimanual teleoperation, allowing a human to control two robot hands and arms for complex behaviors such as pouring, scooping, hammering, drilling, and chopstick manipulation. In DexWild, we move data collection beyond the lab by collecting 1000s of demonstrations with human hands in everyday environments and aligning them with robot observations, allowing policies to generalize across environments and embodiments. Finally, in IFG, we address functional grasping directly, using internet-scale visual guidance to generate task-aware grasps that know not only how to hold an object, but where it should be held for a task. Together, these works bring the thesis full circle: they show that fine-grained dexterity requires robot hands that are not only built more like human hands, with strength, compliance, and rich motion, but also taught more like human hands, through demonstration, interaction and experience from the human world.

Part I

Designing Dexterous Robot Hands for Machine Learning

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://leaphand.com/

Figure 2.1: **Relative size of popular robot hands to scale.** *Left to right,* adult human hand, Allegro Hand [7], LEAP-C Hand, LEAP Hand, Inmoov [22], D'Manus [75]. LEAP Hand is similar in size to Allegro and ~ 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 [88, 156, 229]. 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 [47, 265] across truly diverse terrains, robotic manipulation results in the real world have mostly been limited to one degree-of-freedom parallel jaw grippers [72, 129, 364]. In contrast, dexterous manipulation has largely been limited to simulation [107, 181] with comparatively fewer real-world results [56, 108, 170, 268, 348].

A major bottleneck in democratizing dexterous manipulation has been the hardware. Tendon-based hands like Shadow [5], while impressively capable [56], cost over 100K USD and often require significant maintenance [54] due their complicated nature. While Inmoov [22] 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 [7, 215]. 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 7.2) and is expensive at over $16K. Please see Section 8.3 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 [6, 19, 20, 43] 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 8.2, 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 [249]. 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://leaphand.com/`.

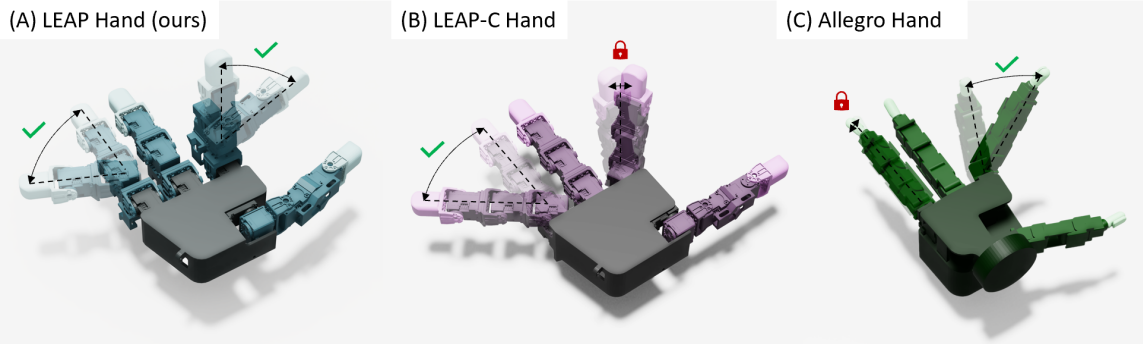

Figure 2.2: Comparison of MCP joints in different robot hands and their dexterity and two different positions. (A) In LEAP-C Hand there is a large range of motion at extended but not flexed position (B) In LEAP Hand, at flexed and extended positions, the fingertip has a large range of motion. (C) In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

2.3 Related Work

Robot Hands Shadow [5] and ADROIT [214] hands paved the way to enable complex, contact-rich dexterous tasks with an anthropomorphic ball joint MCP. [54, 56]. However, they are costly (100k USD) and require constant maintenance. In contrast, the Inmoov hand is 3D printable, tendon-driven, and human-like[22]. 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. [71] present a soft tendon-driven hand that is very flexible for many configurations. Unfortunately, it is difficult to simulate due to its deformable nature [142, 148]. In contrast to tendon-driven hands, which have motors in the wrist, the Allegro Hand [7] has its motors in the finger joints. It is most popular in research labs [58, 169, 353, 361, 371] 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) [75] 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. [406] accomplish within-hand manipulation using rollers attached to the fingers. Humans lack this degree of freedom and manipulate objects in a very different manner. [11, 203, 204] 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 [30]. 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 [34]. This allows us to directly print soft flexible filaments like Ninjaflex [3]. We use this to create many parts for LEAP Hand like the hard palm and soft rubber fingertips.

Learning Dexterity Using a Shadow hand and Sim2real, Andrychowicz et. al. [54, 56, 211] accomplish in-hand rotation for a variety of objects. Policies that scale to thousands of objects can also be trained in simulation [107, 180]. [273] uses the D'Hand to reposition a valve. In-hand rotation of Baoding Balls using the Shadow Hand trained purely in the real-world [271], and pipe insertion using the D'Hand [162] are other notable examples of dexterous manipulation.

Several recent works focus on supervising policies of robot hands [150, 195, 320, 387]. from MANO [317] parameters which parameterize a human hand. Closely related is the teleoperation of robot hands from real-time video [169, 352], which can be used to guide learning and improve sample-efficiency [304, 311]. Hand poses can be extracted from video data available on the web to learn manipulation policies [254, 304]. Large-scale pre-training using internet videos is helpful for efficiently training robot hands for downstream tasks using a few task specific-demos [338] and also on non-dexterous manipulation [67, 281].

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 [67, 304, 338] with machine learning. This can be done using methods such as teleoperation with VR gloves or extracting keypoints from videos of human hands [352]. 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 [239]. 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 [5]. 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 [7, 215] have been more popular.

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 [215]. Prior seminal work has proposed two designs for this (Fig. 8.2). 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 [215]) is less dexterous when closed.

The reason for the lost dexterity in both LEAP-C Hand and Allegro is that the axis of

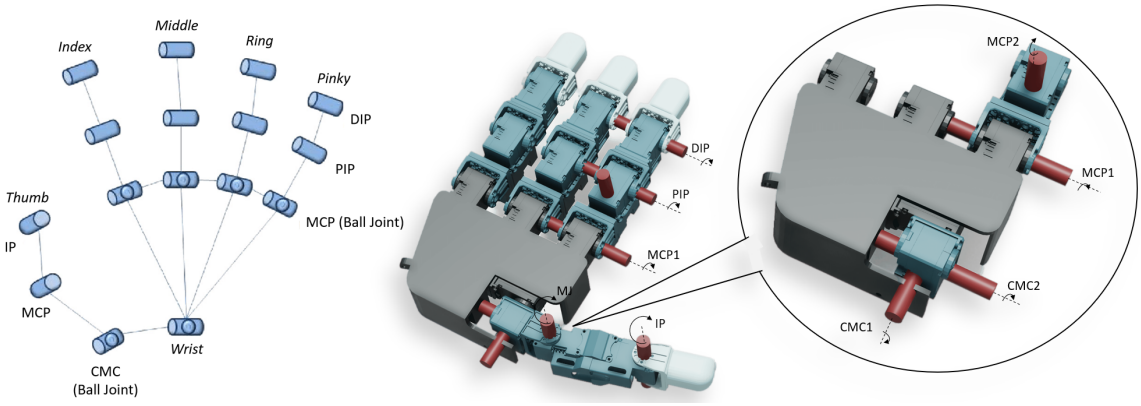

Figure 2.3: The human hand kinematics above has ball joints at the MCP and CMC joints. These are difficult joints for low-cost hands to include. Left Figure from [100]. Comparison of MCP joints in different robot hands. (A) In LEAP-C Hand there is a large range of motion at extended but not flexed position (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.

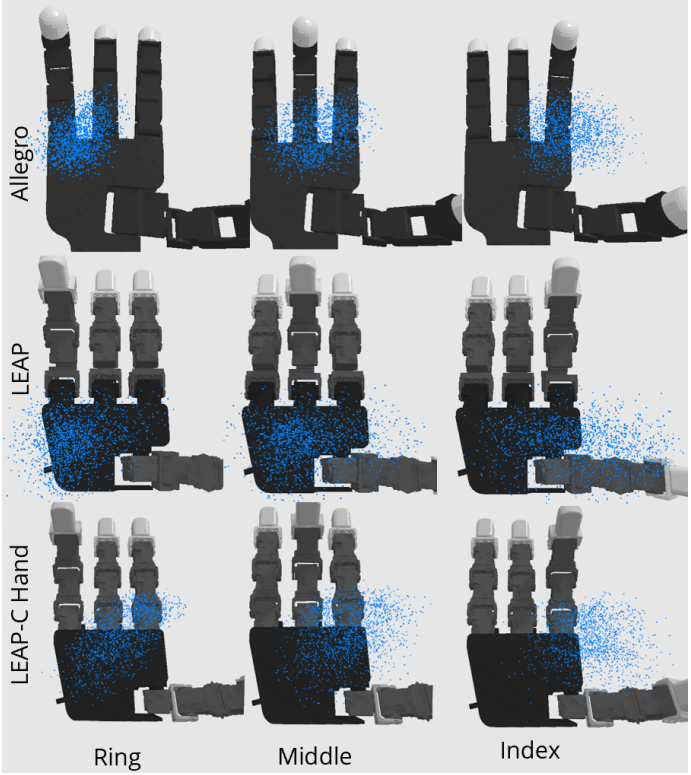

Figure 2.4: We compare the possible positions of opposability of the thumb and each of the other fingers on each of the three hands. We find that LEAP Hand has the best even spread on top of the palm and a very large contact area.

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 8.2 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. 8.2). 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. [101] and Lee et.al. [215], 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 [101]. 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. [403]. 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 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.

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.

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 [5] and AllegroHand [7] cost 100K and 16K USD, respectively, and must be 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 [6, 20] and manipulation [19, 40, 43] 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.

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.

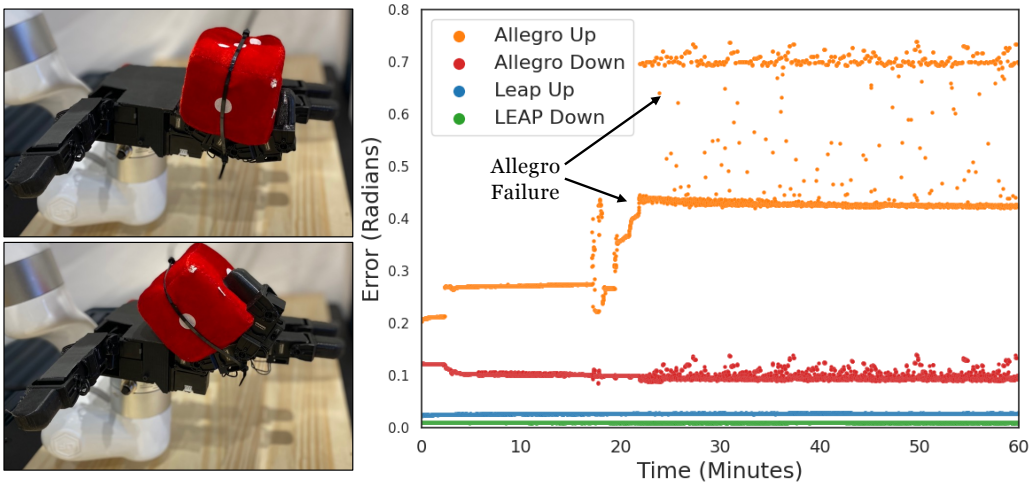

Figure 2.5: **Repeatability test**. *Left:* An illustration of LEAP-C Hand performing repeating grasp-ungrasp on a small plush dice using one joint for an hour. *Right:* Comparison of LEAP-C Hand and Allegro [7]. After just 15 minutes, the Allegro Hand cannot maintain movement. In contrast, LEAP Hand continues to maintain movement with minimal joint error (< 0.05 rad). Videos at https://leaphand.com/

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 [169, 353]. 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 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 [35] 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.

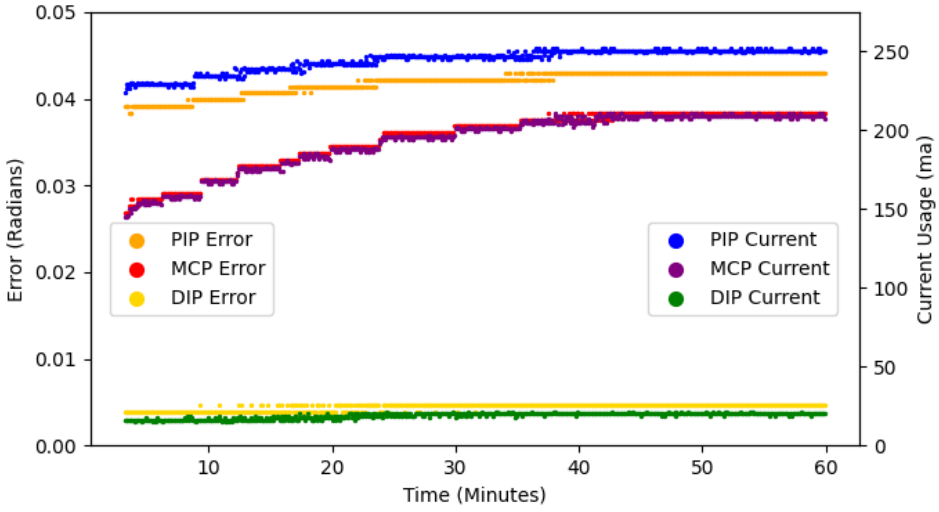

Figure 2.6: **Endurance Test**. We balance a heavy 2kg weight on only one fingertip of LEAP Hand for one hour. On the left axis, we show that the angle error of commanded vs actual remains small. The right axis shows that the current use does initially increase with the temperature of the motor. However,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

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 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

Hand	**Strength** (N)	**Power Density** $N \times DOF/(cm^2)$
Bauer et. al [71]	37.4	0.677
Allegro Hand [7]	8.5	0.35
D'Manus [75]	27.8	0.313
Inmoov Hand [22]	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.

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 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 [75] 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. 10.1). A $200 Ender 3 3D printer [17] 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 [15], 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

Object	Grasp Type [239]	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.

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 [7] 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 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 [24] 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 7.2 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

#	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.

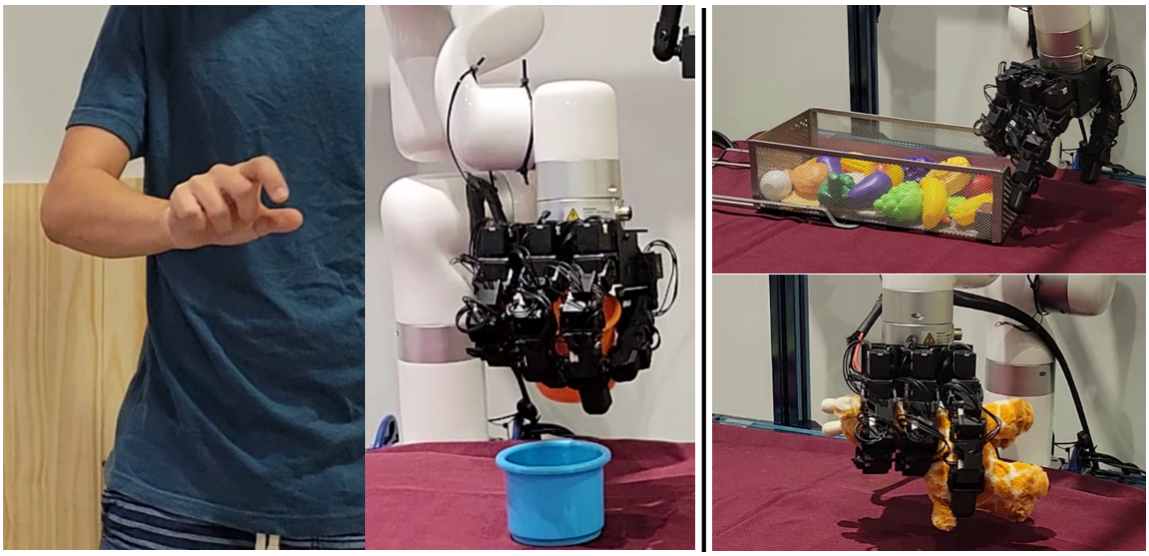

Figure 2.8: **Teleoperation and Behavior Cloning.** *Left*: We perform dexterous teloperation using Telekinesis [352] with a single view color camera. *Right*: We perform behavior cloning from internet video and teeloperated demonstrations using Videodex [338].

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 method from Robotic Telekinesis [352] (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:

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

[352] trains an MLP $H_R(.)$ to implicitly minimize the energy function described in Equation 4.2.

	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 [338].** Hand policies are pretrained on internet videos of humans and finetuned using minimal ($\approx$ 100) teleoperated demos. On this practical use case, LEAP Hand performs better on 12 of 14 {task}$\times${train, test} pairs.

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 [352] 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 [170], LEAP Hand kept running without a degredation 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 [122] as pre-training for our policy using VideoDex [338] along with NDP [64], see Fig. 2.8 (right). We also only use demonstrations collected from prior work [338] 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

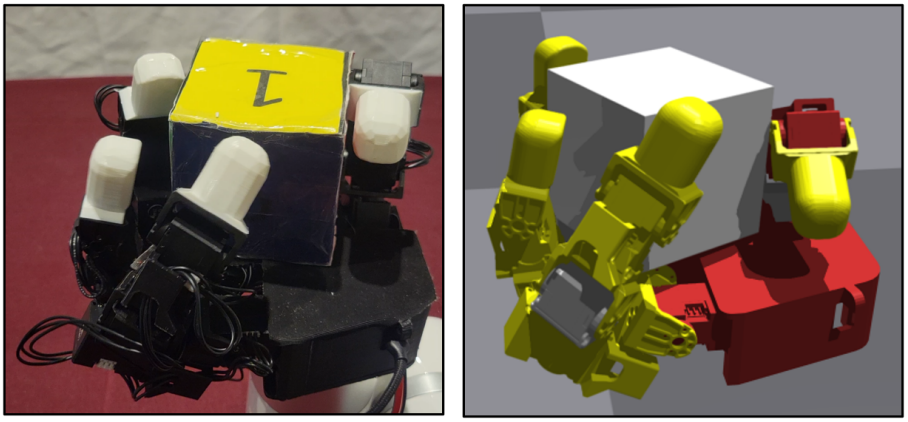

Figure 2.9: **Sim2Real transfer.** *Left*: Simulated LEAP Hand in Isaac Gym [253] completing an in-Hand manipulation task. *Right:* LEAP Hand completing the same task in the real world. Please see our website https://leaphand.com/ for our open source pipeline.

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 [116] architecture for our policy. We first generate a cache of stable grasps similar to [303]. 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 the penalties are -0.1, -1, -0.1, -0.3 respectively. We train PPO [327] with BPPT [391] in IsaacGym [253].

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

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

3.1 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 3.1: 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.

3.2 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 [381] or deactivating explosives [134]. 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 [309, 311].

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 [78], Virtual Reality devices [26, 42], haptic gloves [2] and MoCap [415]), each of them 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 complex-

ity 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 [168], 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.

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 10.1. We demonstrate the

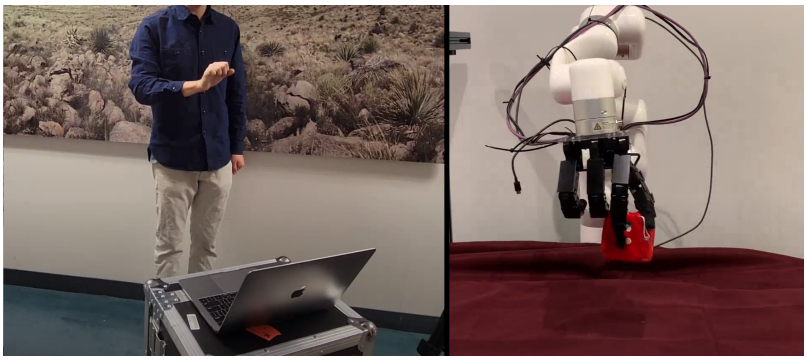

Figure 3.2: An operator completing a dice pickup task while watching the robot through a video conference. Video demos are at https://robotic-telekinesis.github.io/.

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.

3.3 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 [317] is a low-dimensional parametric model of a human hand, and SMPL [246] is an analogous model for the human body. SMPL-X [287] 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 [331] or skeletons [94], or perform a full 3D reconstruction [150, 195, 387]. Rong *et al.* [320] 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 [420], Human3.6M [186] and the CMU Mocap Database [12] are examples of densely-annotated datasets in clean indoor settings. On the other end of the spectrum, the 100 Days of Hands [331] and Epic Kitchens [122] 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.* [82] use thermal

imaging to capture impact heatmaps that reveal patterns in the ways human hands interact with everyday objects. Taheri *et al.* [365] 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.* [172] and Cao *et al.* [95] propose methods for joint hand and object reconstruction, and Hampali et al. [165] present a dataset of hand-object interactions with 3D annotations. Liu *et al.* [238] builds a taxonomy of human grasps to understand the cognitive patterns of human hand behavior [230].

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* [224] train a deep network to map human hand depth images to joint angles in the robotic Shadow Hand, and Antotsiou *et al* [57] 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 [168], 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.* [383] propose a cycle consistency objective to transform motion between animated humanoid characters of different body shapes. Peng *et al* [290] use an approach based on keypoint matching to learn robotic locomotion behaviors from demonstrations of walking dogs. Zakka *et al.* [408] learn a visual reward function that allows reinforcement learning agents to learn from demonstrators with different embodiments.

3.4 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, 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 3.2 illustrates an operator solving a grasping task while monitoring the robot through a video conference feed.

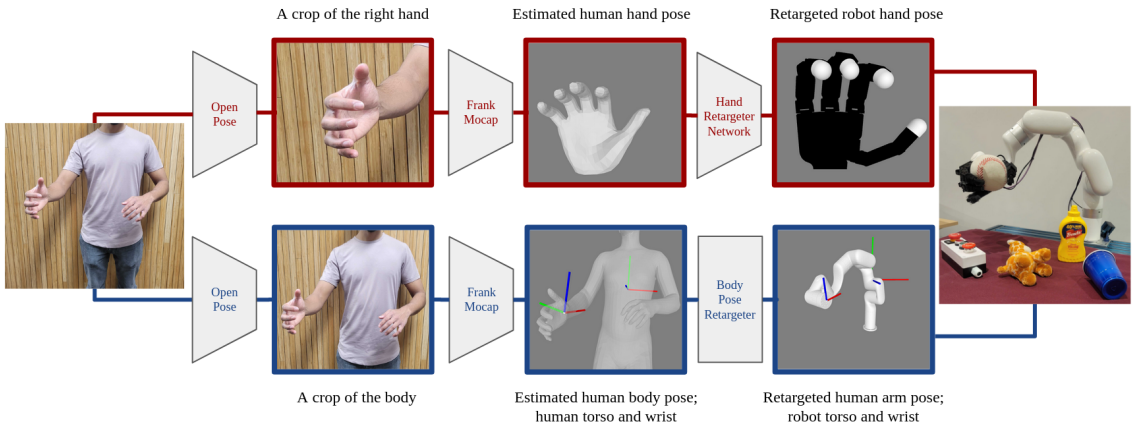

Figure 3.3: A graphical description of our visual teleoperation pipeline. First, a color camera captures an image of the operator. Top: to command the robot hand, a crop of the operator's hand is passed to a hand pose estimator, and the hand retargeting network maps the estimated human hand pose to a 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.

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 4.1, we group the pipeline into two branches: one branch for hand retargeting and the other one for arm retargeting.

3.4.1 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 [94]. The resulting image crop goes to a pose estimator from FrankMocap [320] to obtain hand shape and pose parameters of a 3D MANO model [317] of the operator's right hand. See the "OpenPose" and "FrankMocap" modules in the top row of Figure 4.1 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 to place it in an analogous hand pose (see the third panel on the top branch of Figure 4.1). 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*.

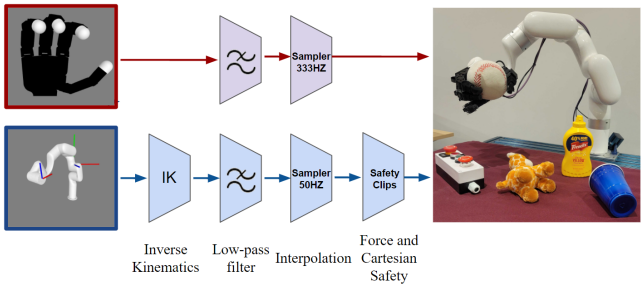

Figure 3.4: **Control Pipeline**. Description of our control stack. Raw target poses are received from the visual retargeting modules, then Inverse kinematics, low-pass filtering, sampling, and safety clipping are performed. The smoothed commands are sent to the robot.

- 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 [122] dataset, which captures ego-centric videos of humans performing daily household tasks, and the 100 Days of Hands [331] dataset, which is a collection of YouTube videos. We run the hand pose estimator from [320] (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 [420], 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

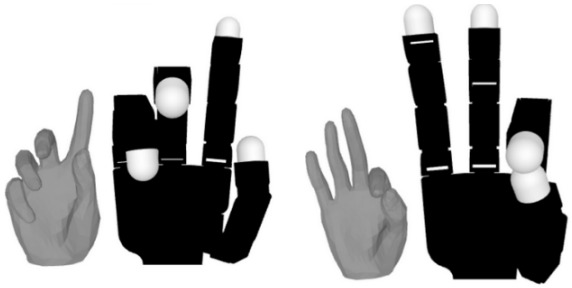

Figure 3.5: **Human-to-robot Translations**. The inputs and outputs of our hand retargeting network. Each of the pairs depicts a human hand pose, and the retargeted Allegro hand pose.

those of the human's. Following [168], 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 $c_i \cdot \mathbf{v_i^a}$:

$$E(\ (\beta_h, \theta_h),\ q_a\ ) = \sum_{i=1}^{10} ||\mathbf{v_i^h} - (c_i \cdot \mathbf{v_i^a})||_2^2 \tag{3.1}$$

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 [96].

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 4.2 that captures 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:

$$\underset{f}{\operatorname{argmin}}\ \mathbb{E}_{x\in\mathcal{X}}\left[E(x, f(x))\right], \tag{3.2}$$

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 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*

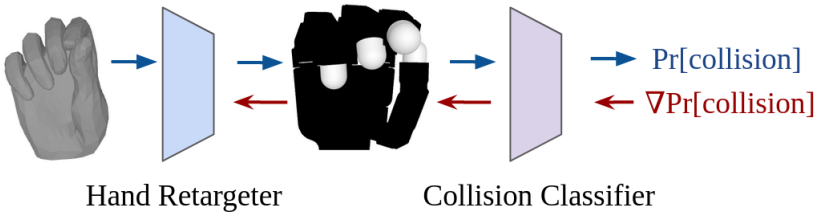

Figure 3.6: A trained self-collision classifier is used as an adversary that penalizes self-colliding joint configurations. The blue arrows denote the forward pass, and the red arrows denote the flow of gradients during the backward pass.

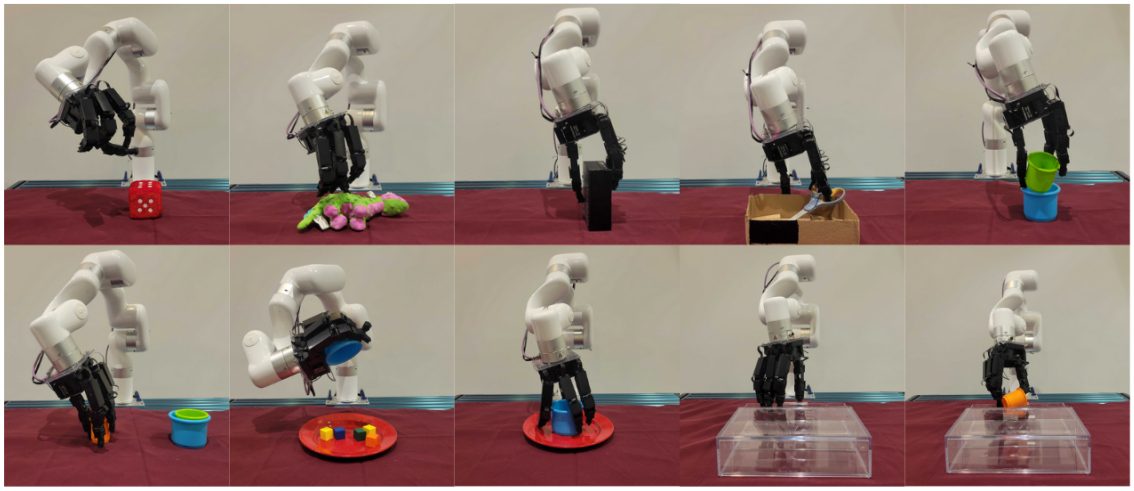

Figure 3.7: Ten different teleoperation tasks. Top row, left to right: Pickup Dice Toy, Pickup Dinosaur Doll, Box Rotation, Scissor Pickup, Cup Stack. Bottom row, left to right: two cup stacking, pouring cubes onto plate, cup into plate, open drawer and open drawer and pickup cup. Please see videos at https://robotic-telekinesis.github.io/.

the retargeter network, as shown in Figure 3.6. 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.

	Success (rate)		Completion Time (sec)		
Task	Ours	DexPilot-Mono*	Ours	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 3.1: 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 [168]). Our system, which uses a neural network retargeter, outperforms the baseline in 7 out of 10 tasks.

3.4.2 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 [94], then pass the crop to the body pose estimator from FrankMocap [320]. We model the human body using the parametric SMPL-X model, and 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 4.1). The bottom row in Figure 4.1 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 [89] 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 3.4 for a depiction of our control stack, and see the appendix for further details about the arm retargeting modules.

3.5 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 [168], but the overall system (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.

$$y^* = \underset{y}{\operatorname{argmin}}\, \mathcal{L}(x, y). \tag{3.3}$$

The code for DexPilot's [168] kinematic retargeting module is not available, so we implement their online optimization solver using the Jax GPU-accelerated auto-differentiation engine [81].

We do not compare our system to the full DexPilot system. DexPilot is designed

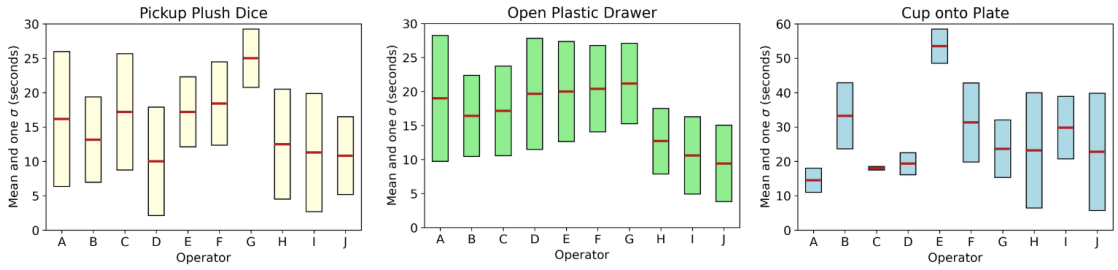

Figure 3.8: Ten novice operators were asked to complete tasks: (1) picking up a plush dice, (2) opening a plastic drawer, and (3) placing a cup onto a plate. For each task, the mean and standard deviation completion times were computed over seven trials.

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 [168], but is otherwise identical to our system.

3.5.1 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 10.4. 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 are described in Table 3.1. 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

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 3.2: 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.

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 3.2. 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.

3.5.2 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 3.8 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.

3.6 Analysis

3.6.1 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 [102], 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 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).

3.6.2 Self-Collision Avoidance

We perform an ablation on the weight of the self-collision classifier (Section 3.4.1) 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 [420] 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

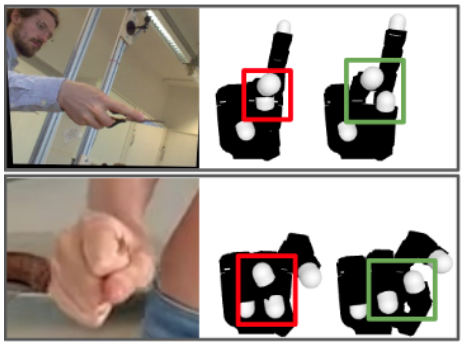

Figure 3.9: **The contribution of an adversarial self-collision loss**. The red boxes highlight instances where the vanilla retargeting network outputs Allegro hand poses that result in self-collision. The green boxes depict the predictions of the network trained with self-collision loss. These robot hand poses maintain functional similarity to the human's hand pose, but avoid self-collision.

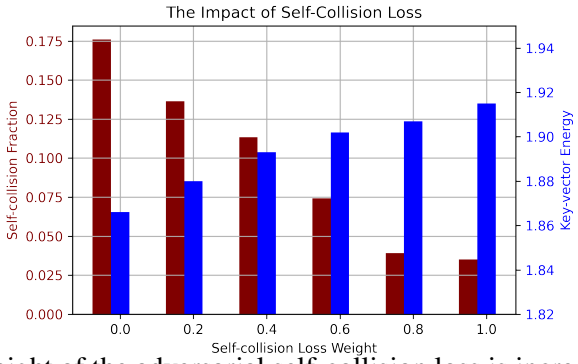

Figure 3.10: As the weight of the adversarial self-collision loss is increased, the hand retargeter network produces fewer self-colliding joint configurations (maroon), but incurs a higher energy with less similar poses (blue). A higher energy means that the predicted robot hand pose is dissimilar to the operator's hand pose.

sum of all the other key-vector matching loss terms in the energy function.) Each 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 3.10. 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 3.9. 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.

3.7 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 4

VideoDex: Learning Dexterity from Internet Videos

4.1 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

4.2 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 [122, 159, 161]. However, most of the prior work tends to use human data as a mechanism for pretraining just the visual representation [274, 277, 293, 329, 394], much like how deep learning has been used as a pretraining tool in related areas of computer vision [110, 175] and natural language processing [87, 139]. 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 4.1). 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 [122, 161]. 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 [65, 66] (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 in human activity datasets from the internet. VideoDex then only needs to adapt to real

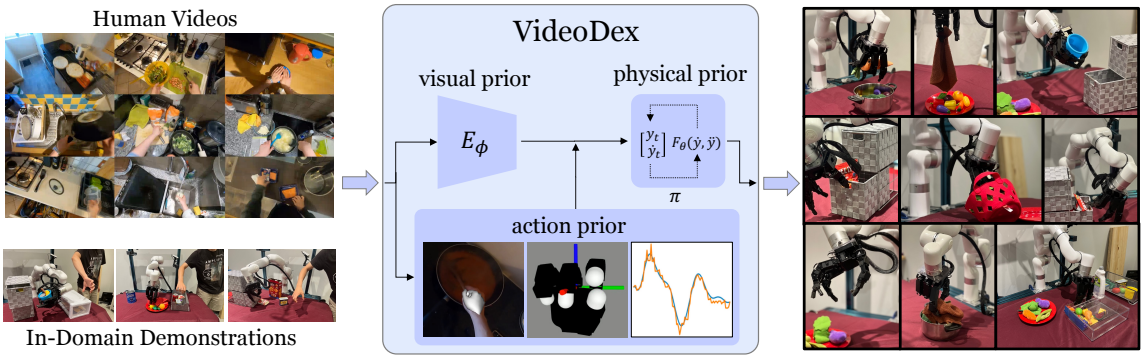

Figure 4.1: We re-target human videos as an action prior, use pretrainined embeddings as a visual prior, and use Neural Dynamical Policies (NDPs) [65] as a physical prior to complete many different tasks on a robotic hand.

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.

4.3 Related Work

Learning for Dexterity Reinforcement learning (RL) with an engineered reward function can show dexterous simulation results [192, 220] but requires lots of data, especially in high DOF dexterous manipulation. This requires simulators [253, 372], which cannot model physics properly, making real-world transfer difficult. Behavior cloning is an approach [80, 294] that can work safely. DIME [59] involves using nearest neighbor matching of image representations with demonstrations to determine actions. Qin et al. [305] teleoperates and learns policies in simulation, followed by Sim2Real transfer. DexMV[304] uses collected human hand videos for robot hand imitation learning. DexVIP [255] 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 [420] for hand poses, 100 Days of Hands [331] for hand-object interactions, Something-Something [159] for semantically similar interactions, Human3.6M [186] and the CMU Mocap Database [12] for Human pose estimation. Epic Kitchens [122], ActivityNet datasets [145], or YouCook [127] are action-driven datasets we focus on for dexterous manipulation.

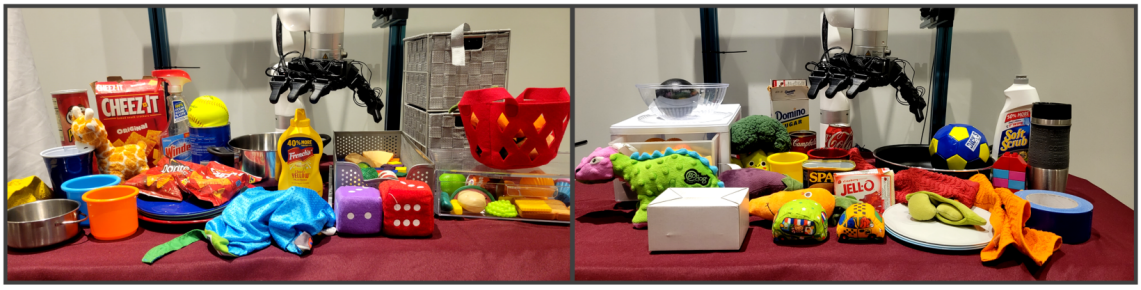

Figure 4.2: The collection of train objects (left) and test objects (right) used for experimentation.

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 [317] model and the human body using SMPL, SMPL-X [246, 287] models. There are many efforts in human pose estimation such as [195, 320, 387]. We focus on FrankMocap [320] for our project as it is robust for online videos. Traditionally, teleoperation approaches have employed hand markers with gloves for motion capture [166] or VR settings [213]. Without gloves, Li et. al. [224] used depth images and a paired human-robot dataset for teleoperation, and Handa et. al. [168] 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 [67, 104, 332], learn action correspondences [325] both in a paired [333] and unpaired manner [355]. This data can also be used to extract explicit actions by leveraging structure in the collection (such as reacher-grabber tools [404]) or prediction of future hand and object locations [217], as well as keypoint detectors [126]. This can also be used to build representations for robot learning [277, 281]. R3M [277] trains on the Ego4D [161] 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.

4.4 Background

4.4.1 Neural Dynamic Policies

Neural Dynamic Policies (NDPs) [65, 66, 130], 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 [184, 283, 297, 324]:

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f_w(x, g), \tag{4.1}$$

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.

4.4.2 Learning from Watching Humans

Recently, Sivakumar et al. [352] introduced Robotic Telekinesis, a pipeline that teleoperates the Allegro Hand [7] using a single RGB camera. Leveraging work in monocular human hand and body pose estimation [320], hand and body modeling [246, 287, 317], 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. [352]'s approach for demonstration collection.

We borrow the human hand to robot hand retargeting method from Robotic Telekinesis [352] 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:

$$E_\pi(\ (\beta_h, \theta_h),\ q\ ) = \sum_{i=1}^{10} ||v_i^h - (c_i \cdot v_i^r)||_2^2 \tag{4.2}$$

Sivakumar et al. [352] train an MLP $H_R(.)$ to implicitly minimize this energy function in 4.2, conditioned on knowing human poses (β, θ). For more details, we refer the readers to Sivakumar et al. [352].

4.5 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.

4.5.1 Visual Priors from Human Activity Data

Many previous works [274, 277, 394] 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. [277] as a useful visual initialization for our policy. Nair et al. [277] 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 [174] encoder, E_ϕ from R3M [277]. The output of this network is our visual representation for learning.

4.5.2 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. [168] and Sivakumar et al. [352] 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 pretrain our policy network with the action prior. See Figure 4.4 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 [122] but an action detection network such as [118] can be used. Now, we must detect the hand. VideoDex first computes a crop c around the operator's hand using OpenPose [94] and the result is passed to FrankMocap [320] to obtain hand shape (β) and

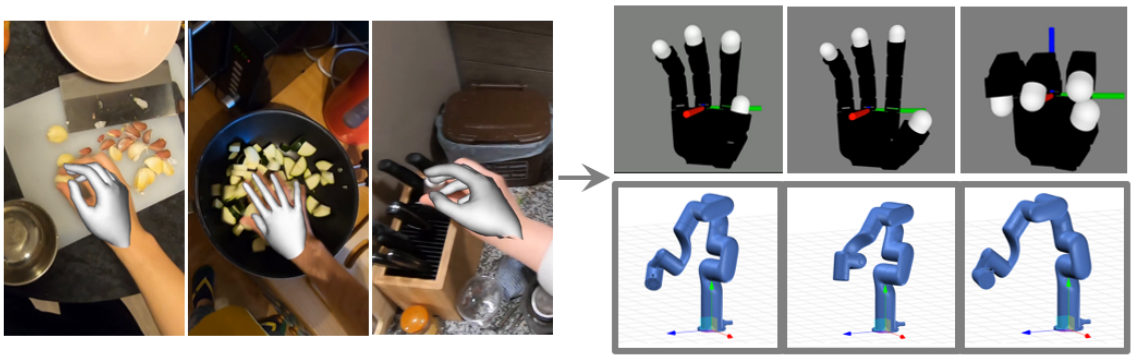

Figure 4.3: To use internet videos as pseudo-robot experience, we re-target human hand detections from the 3D MANO model [317] to 16 DoF robotic hand (LEAP) embodiment and we retarget the wrist from the moving camera to the xArm6 [43] embodiment. Videos at https://video-dex.github.io

pose parameters (θ) of the 3D MANO model [317]. 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 [151]. 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 [326].

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 [93].

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 [416]). Then an estimated depth map from RGB frames only using Adabins [73] 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

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} [320].
  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 4.3, 4.4 and 4.5,
      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

normal vector and the following equations:

$$\text{pitch} = \tan^{-1}(x_{Acc}/\sqrt{y^2_{Acc} + z^2_{Acc}}) \tag{4.3}$$

$$\text{roll} = \tan^{-1}(y_{Acc}/\sqrt{x^2_{Acc} + z^2_{Acc}}) \tag{4.4}$$

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^{World}_{Robot} which rescales and rotates the human trajectory in the world frame τ^{wrist}_W into the robot trajectory τ^{wrist}_R. The final function to

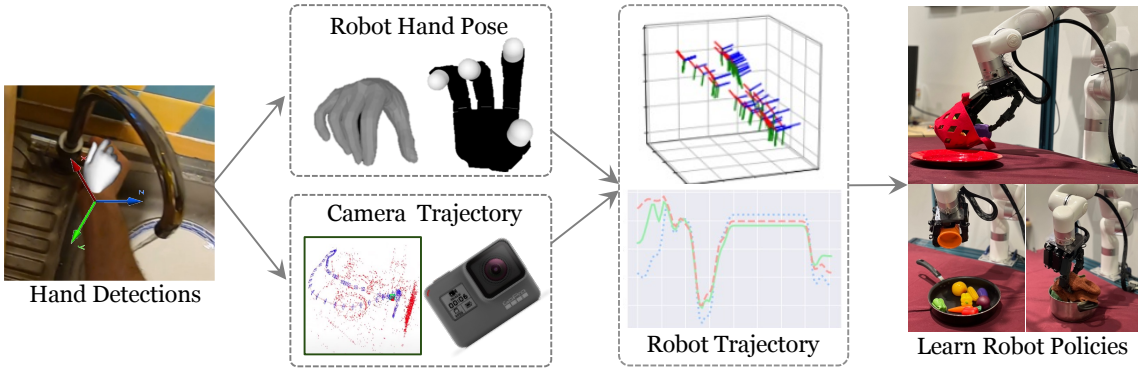

Figure 4.4: To use human videos as an action prior for training policies, we re-target them to the robot embodiment. The detected human fingers are converted to the robot fingers using a learned energy function. The wrist is re-targeted using the detections and camera trajectory and transformed to the robot arm.

obtain M^{Wrist}_{Robot} can be described as:

$$M^{Wrist}_{Robot} = T^{World}_{Robot} \cdot M^{C_1}_{World} \cdot M^{C_t}_{C_1} \cdot M^{wrist}_{C_t} \tag{4.5}$$

Re-targeting Hand Pose Human hands are also in a different *embodiment* compared to that of robot hands, like our 16 DOF LEAP Hand [335] . Similarly, to Sivakumar et al. [352], 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. [352], and get human hand trajectories: τ^{hand}_R in the robot's embodiment. We use τ_R to denote the combined hand and wrist trajectories: τ^{hand}_R, τ^{wrist}_R. See Figure 4.3 for a visualization.

4.5.3 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 [174] encoder provided by Nair et al. [277], E_ϕ. Then, we introduce a *physical prior* to the network, the physically-inspired Neural Dynamic Policies [65, 66].

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:

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

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 [122]. 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 [277] initialized ResNet-18 [174]. We process these features with a 3-layer MLP with a hidden layer size of 512, which are then processed by 2 NDP [65] networks.

4.6 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

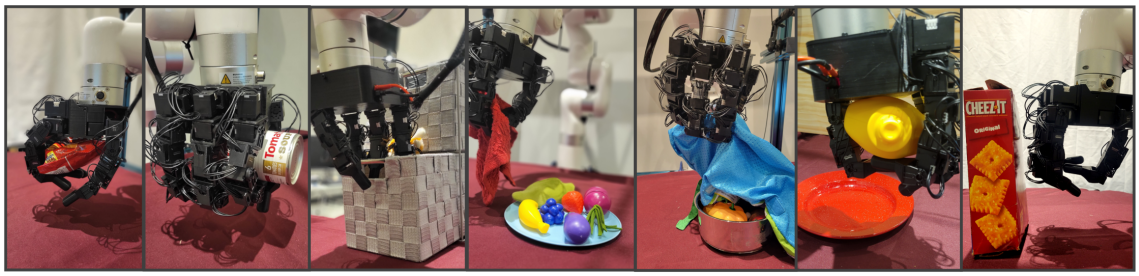

Figure 4.5: Tasks used in experiments. From left to right: pick, rotate, open, cover, uncover, place and push. See `https://video-dex.github.io` for videos of these tasks.

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 [43]. Separate action priors are trained for the 16 DoF LEAP Hand and the 2-finger gripper.

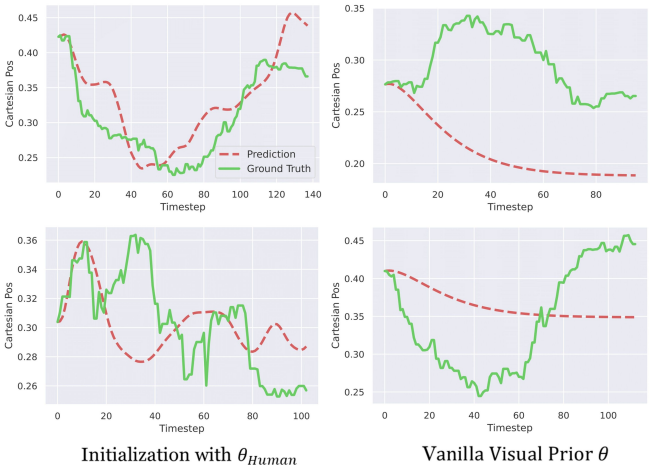

Figure 4.6: Networks initialized using action priors on human data without further training are closer to ground truth robot trajectories than networks only initialized using visual priors.

4.7 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 [130]. `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` [212], 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 [349] (`VideoDex-VGG`) and the MVP network [394] [176] (`VideoDex-MVP`) based representation

	Pick		Rotate		Open		Cover		Uncover		Place		Push	
	train	test	train	test	train	test	train	test	train	test	train	test	train	test
`BC-NDP` [66]	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`[130]	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` [130]	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 4.1: We present the results of train objects and test objects for Videodex and baselines as described above.

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 4.6. 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 4.6. 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 4.6. 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. [277] (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` [212] does not perform as well as the rest of the approaches, even under-performing the Behavior Cloning setup.

	Place	**Open**	**Pick**
`1-DOF BC-Open`[130]	0.62	0.69	0.71
`1-DOF VideoDex`	**0.69**	**0.82**	**0.77**

Table 4.2: We compare how the 1-DOF xArm gripper performs using Videodex. [43] Separate demonstrations were collected using this gripper.

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.

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 4.2, 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 4.3: 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 4.3, 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.

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`) [394] and VGG (`VideoDex-VGG`) [349] and their performance was below VideoDex using Nair et al. [277]. This is likely because both encoders are much larger than the ResNet18 [174] 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. [394] 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 [7]. 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 [335] .

	Train	**Test**
Baselines:		
`BC-NDP` [66]	0.70	0.35
`BC-Open` [130]	0.40	0.25
`BC-RNN` [130]	0.70	0.50
`CQL` [212]	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 4.4: We present the results of the ablations discussed in Section 4.6. These are all performed on the `place` task.

4.8 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 [122], 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.

Part II

Compliance Enables Robust Real-World Learning

Chapter 5

DASH: Designing Anthropomorphic Soft Hands through Interaction

5.1 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.

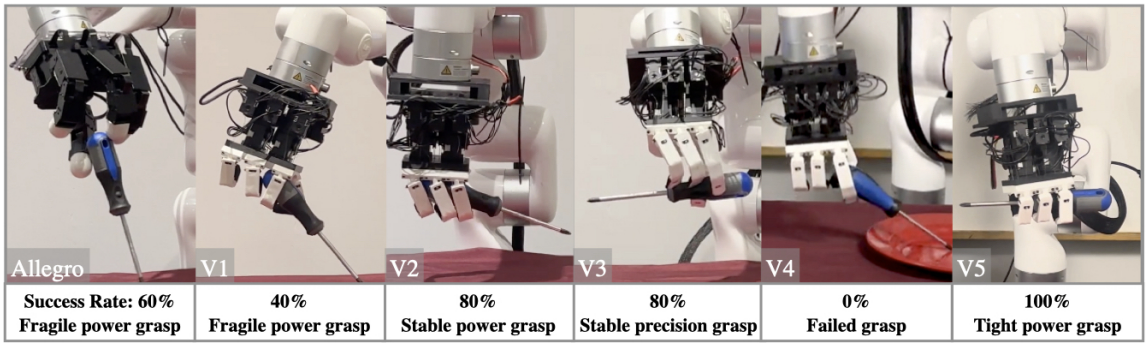

Figure 5.1: Manipulation task performance over five iterations of DASH designed through rapid prototyping and real-world evaluation on tasks alongside task performance of our baseline hand Allegro.

5.2 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 [105]. Hands such as the BRL/Pisa/IIT SoftHand [137] or the RBO hand [300] 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 [71] or to quickly create complex molds [412]. 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 [45] or sequential keyframed open-loop poses [71].

Our key insight is that we can evaluate these systems beyond hand-crafted policies or sequential keyframed open-loop poses using recent advancements in teleoperation systems. Improvements in hand tracking and pose estimation [320] have led to the development of vision-based teleoperation approaches including using a single RGB camera for real-time

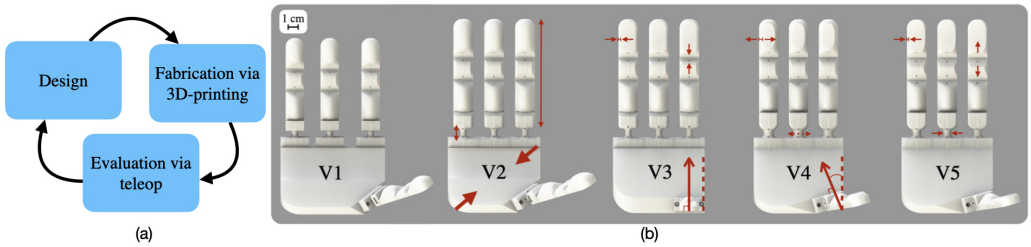

Figure 5.2: (a) Our soft robotic hand design process involving rapid prototyping and real-world evaluation (b) CAD models and differences across DASH iterations v1 through v5, as explained in Section 5.5.

tracking of human hand poses [353]. 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 5.2(a). While manually testing and revising designs to improve systems is not a new concept [412], 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 5.1. 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 [215], 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.

5.3 Related Work

Soft robotic hands such as RBO [300] 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 [119]. 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 [143, 149]. For instance, the SOFA soft robot simulator has been used to co-optimize control and design of soft hands [136]. 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 [412]. 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 [105].

Learning control policies for dexterous manipulation is challenging due to the high DoFs and complex interactions [56, 271]. 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 [60, 338]. 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* [223]. For our soft hand, we adopt a similar joint-to-joint mapping technique as Liarokapis et al. [228].

Our work extends the design and fabrication methodology presented by Bauer et al. [71], 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. [71], 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.

5.4 Experiment Setup

5.4.1 Robot Hand design

Finger Joints

All iterations of DASH consist of four fingers: the thumb, index, middle, and ring fingers (see Figure 5.3(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 fingertip): the metacarpophalangeal (MCP) joint, proximal interphalangeal (PIP) joint, and the distal interphalangeal (DIP) joint. The joints for each finger are shown in Figure 5.3(b).

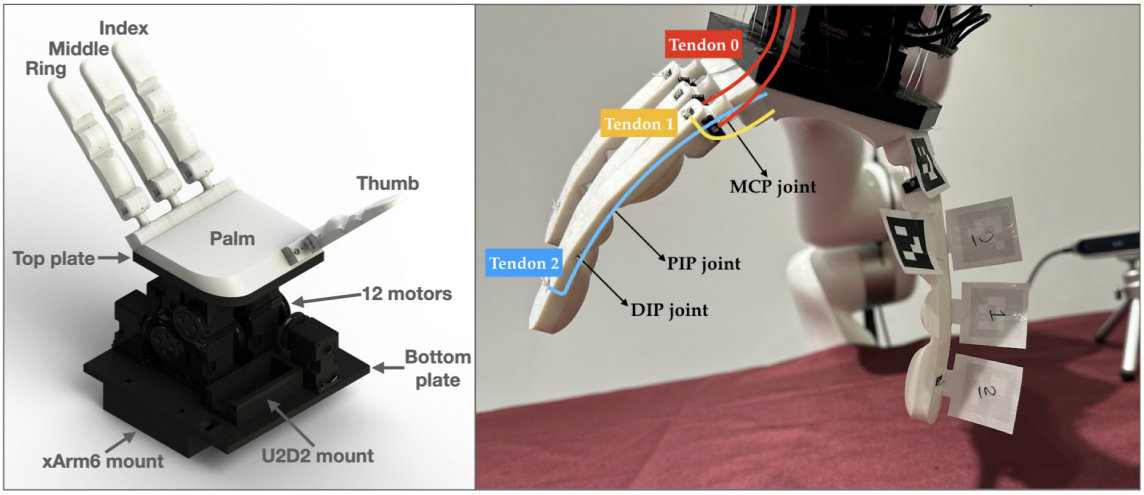

Figure 5.3: (a) Assembly of DASH-v3 (top to bottom) including fingers, palm, top plate, motors, bottom plate, xArm6 mount. (b) Calibration procedure to map motor angles to finger joint positions, where tendon 0 actuates MCP side-to-side, tendon 1 actuates MCP forward folding motion, and tendon 2 curls the finger controlling both PIP and DIP joints.

Tendons

Each finger is controlled by four tendons shown in Figure 5.3(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.

5.4.2 Fabrication using 3D-printing

The hand assembly, shown in Figure 5.3(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 5.3(a)) while the soft hand was printed with Ninjaflex Edge (83A shore hardness) [25] using an Ender 3 S1 Plus. For all the robot experiments, the hand was mounted onto a xArm6 [44] 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 [419].

5.4.3 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 5.3(b), as motors 0, 1, and 2, respectively. The equations for MCP, PIP, and DIP joint angles are shown below. In equation 5.1, we learn the MCP joint angles $\theta_{\text{MCP}_{\text{side}}}, \theta_{\text{MCP}_{\text{fwd}}}$ jointly since the amount of side-to-side angle at the MCP joint can restrict the forward folding motion of the finger. In equation 5.2, the Motor 2 angle θ_{motor_2} is an average measure of the motor angle for the desired PIP and DIP joint angles $\theta_{\text{PIP}}, \theta_{\text{DIP}}$ since the same tendon controls both the PIP and DIP joints. The weights in equations 5.1 and 5.2 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 5.1.

$$\begin{bmatrix} \theta_{\text{motor}_0} \\ \theta_{\text{motor}_1} \end{bmatrix} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix} \cdot \begin{bmatrix} \theta_{\text{MCP}_{\text{side}}} \\ \theta_{\text{MCP}_{\text{fwd}}} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{5.1}$$

$$\theta_{\text{motor}_2} = \frac{\theta_{\text{PIP}} w_5 + b_3}{2} + \frac{\theta_{\text{DIP}} w_6 + b_4}{2} \tag{5.2}$$

Teleoperation System

We use Manus Meta Quantum Metagloves [24] designed for VR tracking and Mocap Use, as shown in Figure 5.4 (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

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

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 5.4, we employ wearable SteamVR trackers [38], 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.

5.4.4 Manipulation Tasks

Each DASH iteration is tested on a suite of tasks, named DASH-30, listed in Table 5.2 which were inspired by the different types of grasps defined in Liu et al. [240] and tasks from previous teleoperation works [167, 353]. 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.

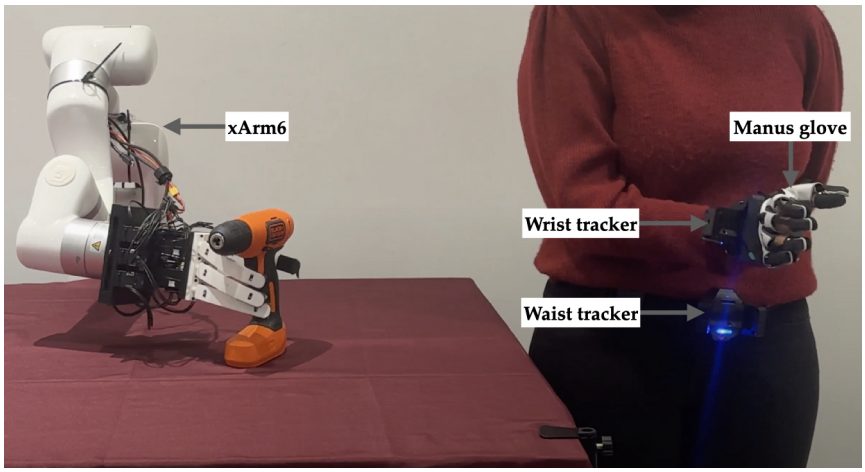

Figure 5.4: Manus Meta Quantum Metagloves used for tracking the hand for teleoperating the robot arm and DASH.

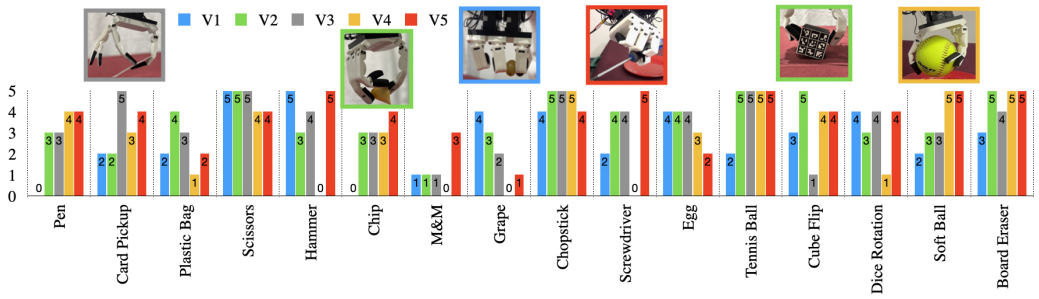

Figure 5.5: Subset of tasks with different performance success across v1 to v5 on specific tasks used to inform design iteration. The top row of inset images shows representative tasks of successful tasks for each hand.

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.

5.5 DASH Iterative Design Studies

5.5.1 Iteration v1

DASH-v1 performs best on pick and place tasks and tasks using adduction-abduction such as the grape task, shown in Figure 5.5 due to the larger hand size and space between fingers as illustrated in Figure 5.2(b).

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 5.2: 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.

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 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 5.3 for results). DASH-v1 hand design is shown on the left in Figure 5.2(b). We designed the full

hand assembly for v1 in 1 month.

Evaluation

We test DASH-v1 on the 30 manipulation tasks from Section 5.4.4, 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 5.6. 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 5.5 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.

5.5.2 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 5.5 inset, due to higher finger strength and fingertip reachability from a smaller palm and longer finger hand design shown in Table 5.3 and Figure 5.2(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 5.2(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) [21] . 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 5.2(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 5.3. 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 5.5, 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 5.6), 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 5.5, the most significant improvements are seen for Pen, Chip, Tennis ball and Cube Flip. The inset in Figure 5.5 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.

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 5.3: 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.

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.

5.5.3 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 5.5. Thinner fingertips, however, led to weaker finger strength which decreased task success for tasks such as Dry-erase Board Eraser and Grape off stem.

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 5.2(b) and in Table 5.3. In addition to the thumb placement, the fingertip shape was changed from a rounded surface to a thinner wedge-like surface (see Figure 5.2). 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 5.3). 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.

Evaluation

As shown in Figure 5.6, 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 5.5). 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.

5.5.4 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 5.5 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

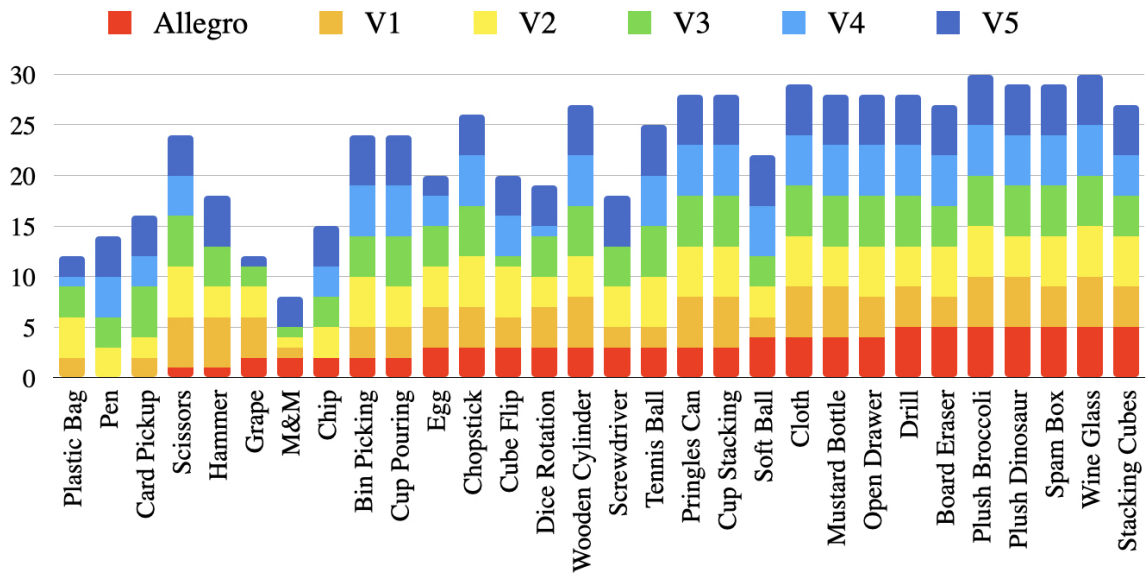

Figure 5.6: Task performance over 5 repetitions of each task across v1, v2, v3, v4, v5, and Allegro as baseline. The tasks are ordered difficult to easy from left to right, according to task performance of Allegro.

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 5.5. 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 5.3. 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.

5.5.5 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 5.1.

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 5.3), 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.

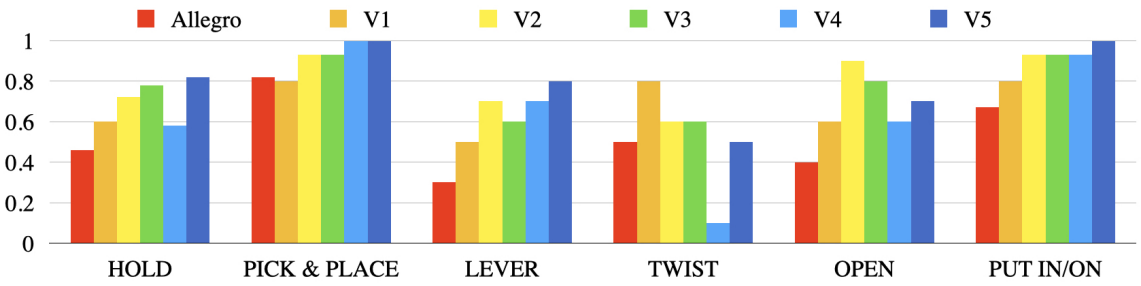

Figure 5.7: Task performance across v1, v2, v3, v4, v5, and Allegro as baseline on each category of tasks from Table 5.2.

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 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 5.5, 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 5.3), 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 5.5. One interesting result involved the v5 screwdriver hold, which aligned perfectly in the groove on the tool handle.

5.6 Baseline Study: Allegro Dexterous Hand

Allegro [215] 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 5.2) 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 5.6. Allegro performed best on pick and place tasks compared to other types of tasks as shown in Figure 5.7. 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 5.6).

5.7 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 5.7, we see that all iterations of DASH outperform the Allegro baseline on most categories of tasks listed in Table 5.2 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 5.5, 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 5.2(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. [71] or RBO [300], using the same setup used for DASH iteration, similar to our Manus [24] 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 5.2(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.

5.8 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.

Chapter 6

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

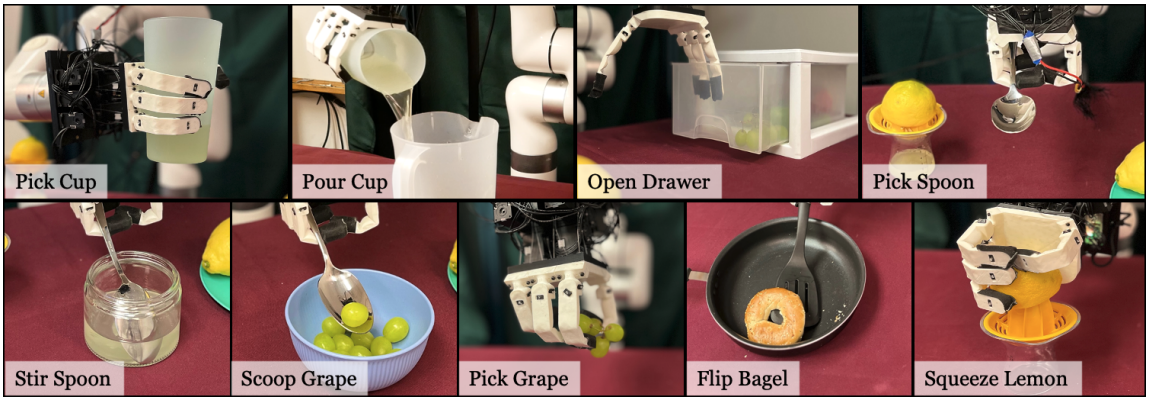

Figure 6.1: We present DEFT, a novel approach that can learn complex, dexterous tasks in the real world in an efficient manner. DEFT manipulates tools and soft objects without any robot demonstrations.

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

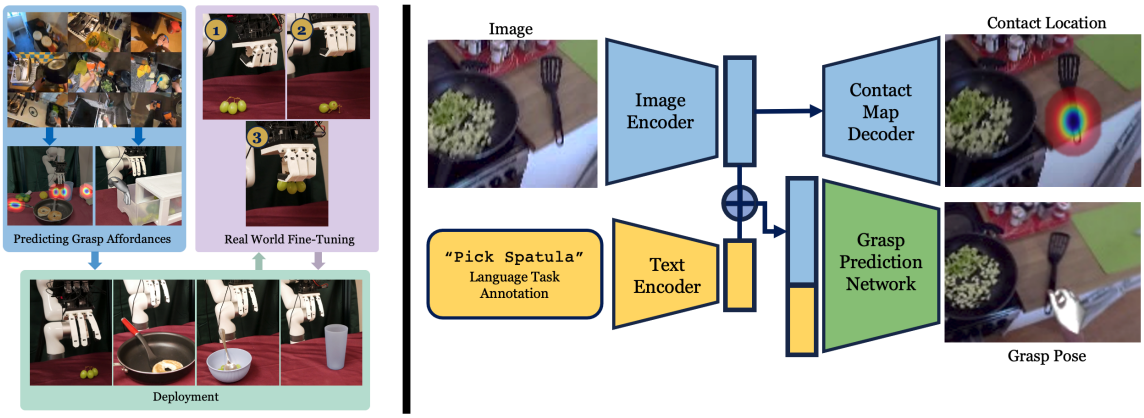

Figure 6.2: **Left:** DEFT consists of two phases: an affordance model that predicts grasp parameters followed by online fine-tuning with CEM. **Right:** Our affordance prediction setup predicts grasp location and pose.

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* [68, 152, 158, 242, 245, 272, 331, 389]. 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 [133, 190], 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 10.1) 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 [198, 210, 270]. On the other hand, data-driven approaches such as deep reinforcement learning (RL) can facilitate complex behaviors in various settings, but these methods frequently rely on lots of data, privileged reward information and struggle with sample efficiency [209, 231, 285, 291, 296]. Efforts have been made to scale end-to-end RL [52, 164, 192, 194, 221, 274] 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 [263], goal relabeling [55], trajectory guidance [218], visual imagined goals [274], or curiosity-driven exploration [264]. 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 [317] of axes aligned with the wrist and a 10-dimensional shape vector. MANOtorch from [398] aligns it with the anatomical joints. Many recent works detect MANO in monocular video [195, 320, 387]. Some also detect objects as well as the hand together [331, 400]. We use FrankMocap to detect the hand for this work. There are many recent datasets including the CMU Mocap Database [12] and Human3.6M [186] for human pose estimation, 100 Days of Hands [331] for hand-object interactions, FreiHand [420] for hand poses, Something-Something [159] for semantic interactions. ActivityNet datasets [145], or YouCook [127] are action-driven datasets that focus on dexterous manipulation. We use these three datasets: [161] is a large-scale dataset with human-object interactions, [244] for curated human-object interactions, and [122] which has many household kitchen tasks. In addition to learning exact human motion, many others focus on learning priors from human motion. [251, 276] 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 [167, 353, 371]. Moving further, human motion in internet datasets can be retargeted and used directly to pre-train robotic policies [256, 337]. Additionally, using human motion as a prior for RL can help with learning skills that are human-like [254, 289, 311]. Without using human data as priors, object reorientation using RL has been recently successful in a variety of settings [56, 108]. 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 [48, 262].

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 [62, 402]. Coupled with dynamics controllers, these robots can learn not to apply too much torque to the environment around them [200, 234, 248]. A variety of touch sensors [74, 347, 362, 407] 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.

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

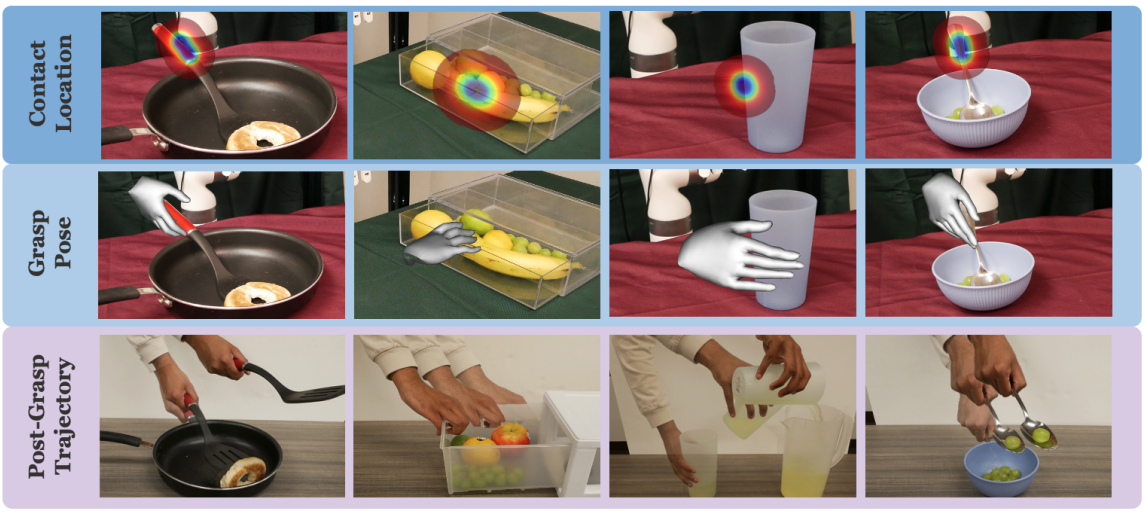

Figure 6.3: We produce three priors from human videos: the contact location (**top row**) and grasp pose (**middle row**) from the affordance prior; the post-grasp trajectory (**bottom row**) from a human demonstration of the task.

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 [331]. Similar to previous approaches [68, 158, 242, 272], 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 [417] 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 [320] 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 [161] for its large scale and the variety of different scenarios depicted, HOI4D [245] for high-quality human-object interactions, and EPIC Kitchens [122] 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 [68]. First, we use a pre-trained visual model [277] 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 [308]. 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 [208] 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 [317] 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 [169, 353]. 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 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 [320]. 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?

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.

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 [67, 190]. 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 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.

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 $k = 1 \ldots N$ **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 $i_1, i_2, \ldots, i_k$ based on rewards
    $\Omega \leftarrow \{\epsilon_{i_1}, \epsilon_{i_2}, \ldots, \epsilon_{i_E}\}$
    Fit $\mathcal{D}$ to distribution of residuals in Ω
  end if
end for
Fit $\pi(.)$ as a VAE to Ω
=0

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 [207, 314, 315, 356] 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, 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.

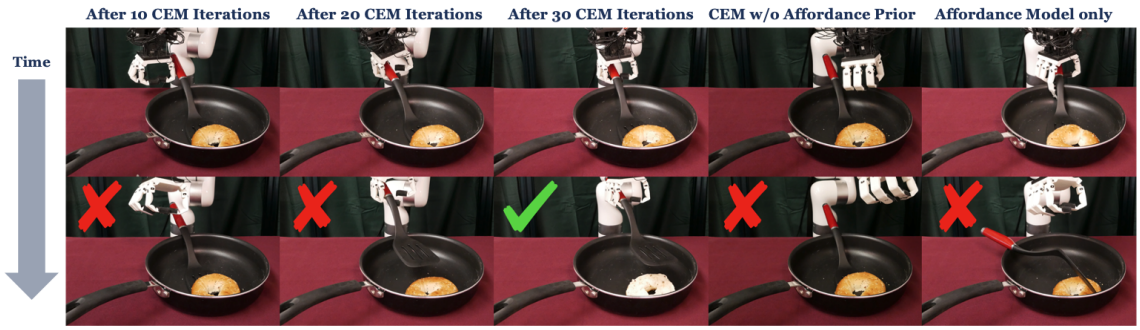

Figure 6.5: Qualitative results showing the finetuning procedure for DEFT. The model learns to hold the spatula and flip the bagel after 30 CEM iterations.

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 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

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.

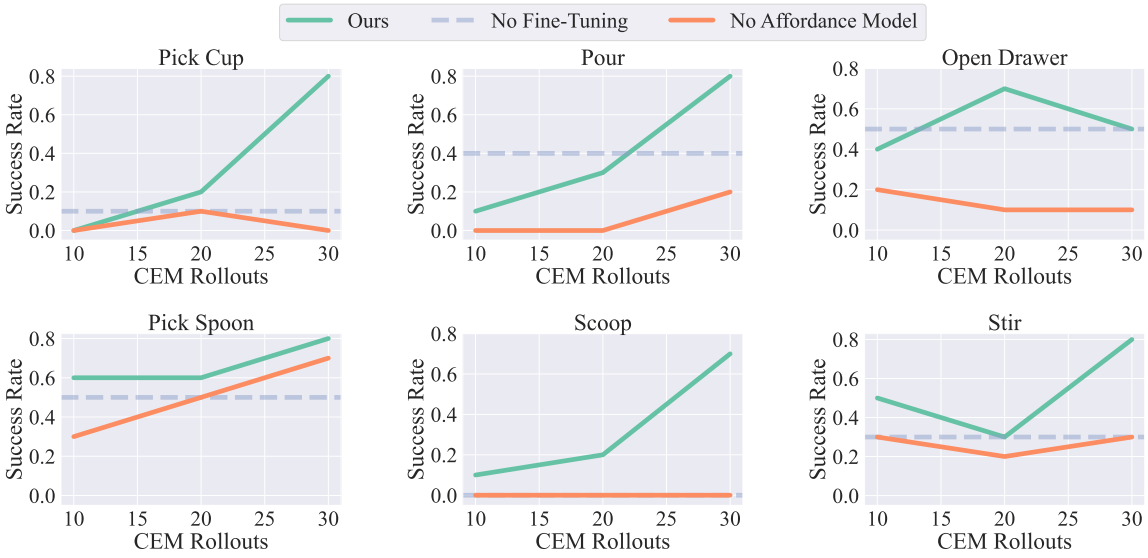

Figure 6.6: Improvement results for 6 tasks: pick cup, pour, open drawer, pick spoon, scoop, and stir. We see a steady improvement in our method as more CEM episodes are collected.

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 [208] 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 parameterized 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 [277] or a ResNet [174] 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.

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 [106] 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

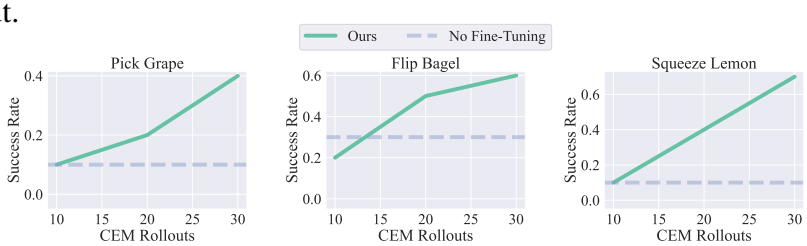

Figure 6.7: We evaluate DEFT on three difficult manipulation tasks.

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.

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.

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.

Part III

Unlocking Fine-Grained Dexterity

Chapter 7

LEAP Hand V2 Adv: Dexterous, Low-cost Hybrid Rigid-Soft Hand for Robot Learning

Figure 7.1: Presenting LEAP Hand v2 Adv, a robot hand designed to emulate the compliance of the human hand. It features a 3D-printed soft exterior skin complemented by a robust internal bone structure. The foldable palm incorporates two powered articulations—one spanning the four fingers and another across the thumb—facilitating human-like grasping. Additionally, a dexterous MCP joint enhances overall dexterity, resembling human hand movements. Its human-like size, straightforward assembly, cost-effectiveness, and open-sourced nature on our website will be useful for dexterous hand research.

7.1 Abstract

The human hand is a remarkable feat of biology, with the ability to handle intricate tools with great precision and strength yet softly handle delicate objects. Robot hands attempting to emulate this have often fallen into one of two categories: soft or rigid. Soft hands, while compliant and yielding lack the precision and strength of human hands. Conversely, rigid hands are brittle to bumps and do not conform naturally to their environment. We call our solution LEAP Hand v2 Adv, a dexterous, $3000, simple anthropomorphic hybrid rigid-soft hand that bridges this gap. First, it achieves a balance of human-hand-like softness and stiffness via a 3d printed soft exterior combined with a 3d printed internal bone structure. Next, LEAP Hand v2 Adv incorporates two powered articulations in the foldable palm: one spanning the four fingers and another near the thumb—mimicking the essential palm flexibility for human-like grasping. Lastly, LEAP Hand v2 Adv boasts a dexterous Metacarpophalangeal (MCP) kinematic structure, making it highly human-like, easy to assemble, and versatile. Through thorough real-world experiments, we show that LEAP Hand v2 Adv exceeds the capabilities of many existing robot hands for grasping, teleoperated control, and imitation learning. We release 3D printer files and assembly instructions for the dexterous hand research community to use on our website at https://v2-adv.leaphand.com/

7.2 Introduction

The human hand is a marvel of biological engineering, enabling a wide range of activities from delicate tasks like writing to robust actions such as lifting heavy objects. Despite its size, it can exert significant force and exhibit exceptional dexterity due to its multiple joints and complex network of bones, muscles, and tendons. This intricate system allows the hand to adapt and interact with various objects efficiently. Additionally, its inherent compliance and pliability prevent damage to delicate items while maintaining the ability to perform controlled, forceful actions. How can we replicate all of these characteristics in a robotic hand?

Building a dexterous hand with all of the capabilities of a human has been a longstanding challenge for the community, which has led to the emergence of two primary approaches: rigid or soft hands. **Rigid hands** are designed in a very heavy manner. They are ideal for

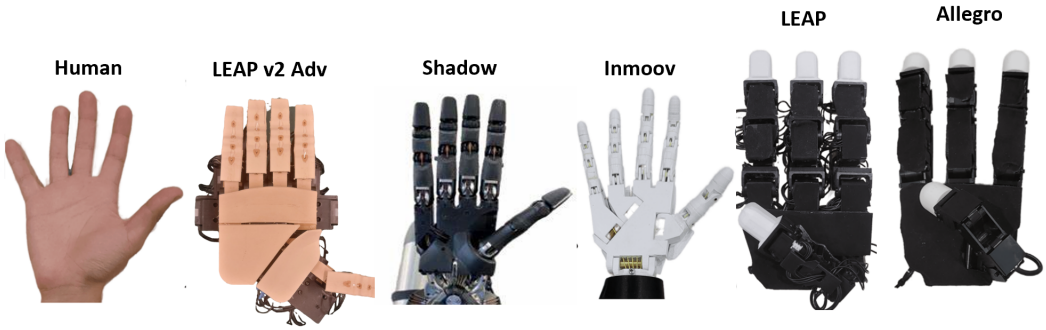

Figure 7.2: **To Scale Comparison of Robot Hands:** LEAP Hand v2 Adv has a similar size and kinematic structure to a human hand. Shadow is most similar to LEAP Hand v2 Adv in kinematic structure and strength but it costs over $150k and uses brittle metal parts. [5] Inmoov has only 5 DOF. [22] LEAP Hand and Allegro both have motors in the joints which make them larger. [216, 339]

high-precision tasks such as tool manipulation. However, much of this comes at the cost of safety, due to brittleness and a lack of compliance. **Soft hands** are made from much more flexible and softer material, which endows them with great compliance. They are safe to deploy as they cannot break the objects they interact with. Soft hands are ideal for manipulating deformable materials (for example food). Unfortunately, due to their build, they struggle at high-precision tasks and cannot exert enough force for many tasks.

In this paper, we build a new **hybrid robotic hand design** that can leverage the advantages of both soft and rigid hands. Our Dexterous, Low-cost Anthropomorphic hand has three new features that are critical to building a *hybrid soft and rigid* hand, taking qualities of both. First, we introduce 3D-printed multi-material fingers that are similar in stiffness, softness, and durability to human fingers. They have strong bones as well as soft skin that allows them to impart strong forces, yet be compliant when necessary. This compliance allows them to be resilient to bumps and bruises. Second, we introduce an **agile palm incorporating two integrated joints** meticulously designed to replicate the conforming characteristics of the human palm. This is a feature often lacking in numerous robot hands, but one we consider crucial. Our robot hand is adept at using its articulated soft palm to effectively grasp, stabilize, and support objects. Furthermore, the palm facilitates crucial opposition from the thumb to the rest of the hand. Third, we design LEAP Hand v2 Adv with finger kinematics designed to maximize dexterity and similarity to a human hand. The MCP (Metacarpophalangeal) joint is designed to be strong yet provides even more flexibility than a human hand. The MCP side joint can move the finger abduction-adduction in both the curled and open positions. The PIP (Proximal Interphalangeal)and DIP (Distal Interphalangeal) joints are articulated with one strong fishing line tendon, meant to emulate the articulations of the human hand.

Through our real-world experiments, we show that LEAP Hand v2 Adv is very suitable for robot learning research and deployment of autonomous agents. It possesses robust strength while maintaining compliance and avoiding brittleness with its soft, yet strong fingers. The hand's palm enables it to execute a range of stable grasps, unlike any other robot hand. We illustrate its capacity to handle teleoperation and autonomous policies remaining resistant to fatigue, overheating, breakage, or loss of accuracy.

Significantly, LEAP Hand v2 Adv can be entirely 3D printed and assembled within a few hours by an inexperienced roboticist using less than $3000 worth of components. This efficiency is partly attributed to the smart utilization of a $1000 multi-material 3D printer, which handles a significant portion of the manufacturing process prior to assembly, streamlining the overall assembly process. All of the 3D print files and a detailed assembly guide are on our website at https://v2-adv.leaphand.com/. The community can adopt LEAP Hand v2 Adv easily and kickstart their robot hand research.

7.3 Related Work

Robot Hand Hardware Over the years, there have been many robot hands built to emulate the human hand to varying success and availability. The Shadow hand [5, 214] has shown impressive results such as in-hand reorientation of a Rubik's cube [54]. However, it is very expensive ($150k), and involves many challenges in deployment. Allegro Hand [7, 216] has been historically a low-cost robot hand ($20k) but is known to often break down and is difficult to repair but has shown impressive results such as teleoperation from video [169, 260, 306, 338, 352] and in-hand reorientation [170]. The Psyonic Ability Hand is a prosthetic with a strong internal hard skeleton and soft exterior with 6DOF[4], limiting its precision.

Recently, with the development of rapid prototyping technologies such as 3D printers and CNCs, there has been an influx of low-cost open-source hands built for academia and research. LEAP Hand [49, 339] introduces a novel dexterous kinematic structure and is simple and easy to use. The Robel suite [53] like D'Manus which are used in reorientation [109] or grasping [76]. There are many other hands built hobbyists such as Inmoov, [22] and DexHand [13] but these are fragile.

Some other robot hands are difficult to manufacture or obtain but have shown impressive results nonetheless. Many such hands [97, 98, 135, 137] are tendon-driven, for example,

the MIT/Utah Hand [188]. The DLR hand has an impressive lineup over many iterations. [91] Recently, Faive Hand [373] shows in-hand reorientation sim2real results. Hashemi et. al. have bones in robotic fingers only. [171]

Ease of Manufacturing Aluminum machining is traditionally used to create strong parts but is very expensive. Manufacturing plastic parts used to have a long process of mold making, casting, curing, and support removal [30] which is only suitable for high volumes. On the other hand, 3D printing has revolutionized small-scale manufacturing. [34] Parts can be printed individually in a few hours automatically.

3D printing materials have advanced significantly. TPU/TPE materials, like [3] and [1], are soft and flexible. Foaming materials such as Colorfabb Varioshore [41] allow adjustments in material density and softness via flow rate, a feature used in LEAP Hand v2 Adv. Nylon and carbon-fiber materials offer strength and durability, while PVA materials serve as dissolvable supports [31]. Multi-material 3D printers, previously limited to industry from E3D [16] and Ultimaker [39], are now available to consumers. [10, 32]. We use the Snapmaker J1S, a $1000 dual-direct-drive extruder 3D printer [37], to print combinations of TPU/PVA/PLA/Nylon-CF for various parts of our hand.

7.4 Robot Hand Design

LEAP Hand v2 Adv is crafted to provide the advantage of both rigid robotic hands, as well as soft ones, while still being easy to produce. This is enabled by three distinctive features not commonly found in readily available robotic hands: (1) The hand exhibits human-like softness and stiffness achieved through the 3D printing of a soft exterior complemented by a rigid internal bone structure. (2) LEAP Hand v2 Adv incorporates two powered articulations in the palm, one spanning across the four fingers and another near the thumb, replicating the crucial palm flexibility essential for human-like grasping. (3) LEAP Hand v2 Adv boasts a dexterous Metacarpophalangeal (MCP) kinematic structure, rendering the hand highly human-like, easy to construct, and versatile for a wide variety of tasks. We find that the combination of these three allows for LEAP Hand v2 Adv to have all of the advantages of both rigid and soft robotic hands, just like a human hand.

7.4.1 3D Printed Bones and Skin

The pliability of human hands allows them to adapt to objects and their surroundings during interactions such as grasping. [278, 418] For instance, when reaching for a delicate object, the hand conforms around the object and applies safe pressure. When bumping into a table, our fingers will bend out of the way and not snap. Our goal is to fabricate conformal fingers with stiffness properties closely resembling those of a human finger, aiming to replicate both its softness and rigidity.

For the soft outer skin, Colorfabb Varioshore or Filaflex Foamy is chosen due to its ability to adjust TPU rubber density based on the flow rate and temperature of the 3D printer nozzle. [41] The change of density allows for the change of flexibility of the skin within different parts. Underneath the skin and within the fingers, we opt for PLA, known for its rigidity and smooth texture. The 3D printer used is an Independent Dual Extrusion (IDEX) printer, specifically the $1000 Snapmaker J1S [37]. This printer allows for the simultaneous use of two materials automatically in one seamless print.

To fine-tune finger stiffness and durability, the TPU material flow rate is adjusted. The exterior of the fingers uses high-flow rate Varioshore (100%) for bruise resistance, while the interior uses a lower flow rate for flexibility. The soft material near the proximal joint is printed with a much lower flow rate (60%) to enhance flexibility and allow seamless folding into the palm. For areas requiring greater resilience, such as the top of the palm, the flow rate is increased to 100

The hard material must line the entire tendon channel to give the finger further structure as seen in Figure 8.3. Without this, the tendon will tear into the soft material. The inlet and outlet positions of the channels are lined up with each other when the finger is fully contracted. This minimizes the bends and undesirable forces on the tendon channels applied by the motors. In the rest of the finger, the hard material is embedded just underneath the surface of the soft material to provide the finger somes structure. It only does not enter the flexure joints of the fingers.

7.4.2 Palm Articulation

When human hands grip objects, the palm folds to conform to the specific shape of the object being grasped. This adaptive behavior allows the hand to firmly enclose the object and support it from many angles. Two pivotal articulations contribute to this capability. Firstly, the CMC joint, situated between the thumb and the palm, facilitates both the movement of the thumb towards the rest of the hand and the articulation of the palm itself. Second,

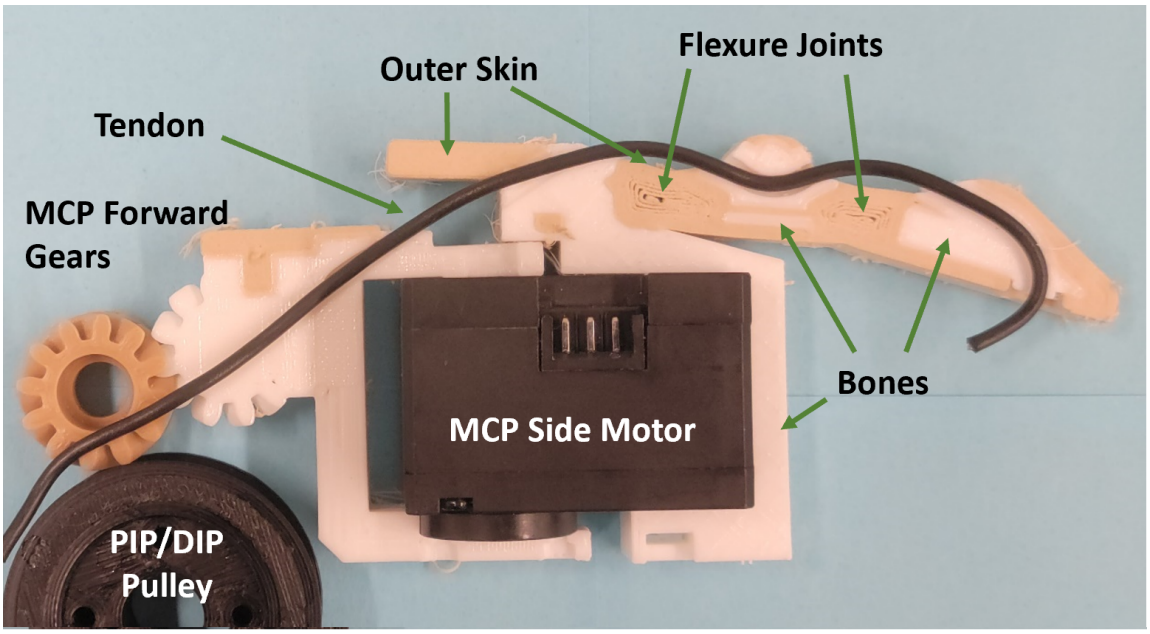

Figure 7.3: A cross-section of the 3D printed finger showing the soft rubber joints, hard PLA bones, and resilient, dense outer skin. The MCP forward is connected by gears to a motor, the MCP side rotates using an embedded motor, and the PIP and DIP joints are actuated together by a tendon connected to a pulley. The range of the MCP forward joint: 90, the MCP side: 180, PIP/DIP: 90

the MCP forward joint and CMC joint in the human fingers also apply a downward force on the palm to make it fold. The synergy of these articulated motions improves grasping capabilities and fosters a better closure around objects.

LEAP Hand v2 Adv has two articulations in the palm as seen in Figure 7.4. These points are placed in similar locations that humans have them in their hands and are independently actuated by strong motors placed just below the replaceable skin of the palm. M2 located near the thumb replicates the natural CMC fold of the thumb. This fold enables the skin in the palm area near the thumb to provide support for objects upon contact. It facilitates the thumb's movement toward the other fingers, enhancing its ability to approach and oppose them effectively for pinches. Second, M1 folds across the entire palm near the MCP joints of the hand. This emulates the human palm, where its MCP joints not only articulate the fingers but also part of the palm and allow the fingers to conform to objects better.

7.4.3 Dexterous Finger Kinematics

In LEAP Hand v2 Adv, we introduce a simple, yet effective finger kinematic design to create a small, highly actuated, strong robot hand that is very human-like. Each finger contains all of the 4 DOF of a human hand by using three motors while remaining easy to produce. The key part of this design is to store powerful Dynamixel XM430 motors densely

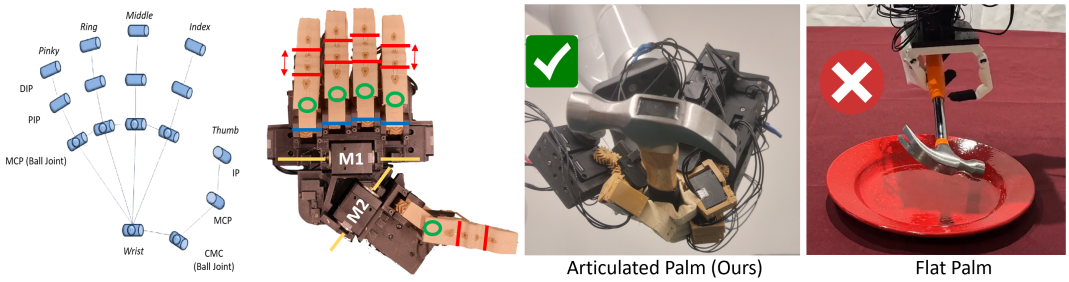

Figure 7.4: **Kinematic Tree:** From Left to right. 1) The kinematic tree of LEAP Hand v2 Adv is similar to a human hand. 2) The palm skin is removed to reveal structure with yellow lines for the palm articulations. The blue lines represent the MCP forward, the green circles represent the MCP side. The PIP and DIP are articulated together and are represented in red. Total: 22 joints powered by 17 motors. 3) This human-like palm kinematics allows it to grasp a hammer better than a hand with a flat palm. See videos at https://v2-adv.leaphand.com/

in the palm for the MCP forward and tendon curl. The XC330 motors are stored under the fingers for the abduction-adduction. This makes our robot hand useful and powerful for a large variety of different tasks.

MCP Side

The MCP side doesn't need as much power, so many motors that fit underneath the 20cm wide finger are strong enough. By placing a Dynamixel XC330 motor directly on the back

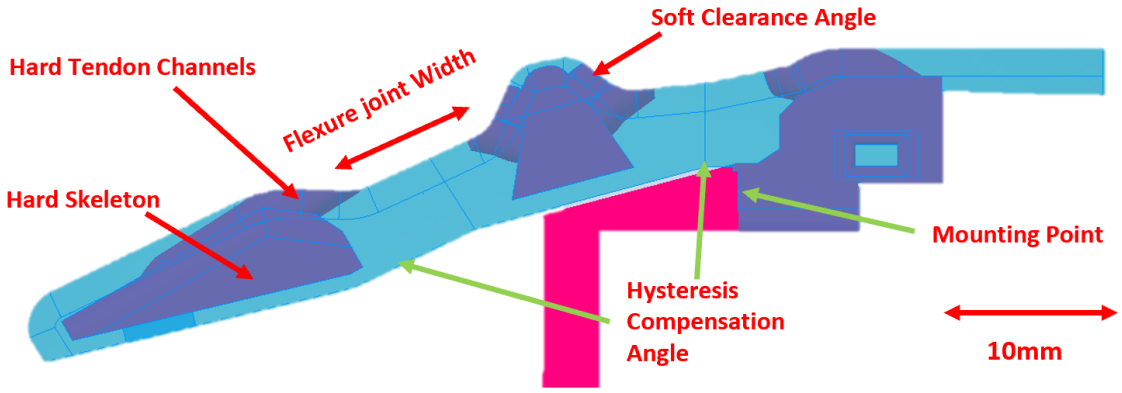

Figure 7.5: **Finger Analysis** We show the key dimensions that create our soft-hard hybrid flexure joint as described in Section 8.4.1. The flexure joint is the light blue soft material that lies between the hard skeleton. It is governed by the flexure joint width, the clearance angle, the hysteresis compensation angle and the size of the hard skeleton. The darker portions are hard tendon channels and rigid structure.

of the finger after the MCP forward joint, the motor applies power directly to the finger and is very accurate with little backlash. This design is inspired by LEAP Hand introduced by Shaw et. al. [339] By following this design, LEAP Hand v2 Adv's finger can move side to side both when the finger is in the down position and the up position which is very dexterous.

MCP Forward

The MCP forward motors must be powerful and accurate, as there is a lot of leverage and power requirements on this joint. They are powered by Dynamixel XM430 motors that are placed beneath the palm and not inside the finger itself to keep the finger slim. The power is then transferred efficiently and accurately through compact 3D-printed gearboxes using 12-tooth double helix gears. It is feasible to 3D print this gearbox using conventional materials, and it readily disassembles for maintenance purposes. A mere four screws suffice to dismantle this joint along with the attached finger, facilitating straightforward replacement and maintenance

PIP and DIP

Within the human hand, the PIP and DIP joint motions are often linked, allowing them to adapt and conform around objects. Our goal is to replicate this behavior in robot hands. To mimic human tendons in a robot hand, motors lower in the palm transmit power through a synthetic tendon (1mm diameter 100 lb. fishing line) to the soft fingers. This means that the motor can be large and powerful, but far away from the fingers themselves. Many tendon-based hands such as Shadow hand require many tendons because of the spring-back requirements of the joints to reopen them. LEAP Hand v2 Adv requires only one tendon per finger because the soft TPU rubber material acts as a flexure joint when pulled and acts as a built-in spring and returns to the open position once the tension on the tendon is released. The coupling of these joints gives a natural conforming nature around objects, which makes LEAP Hand v2 Adv very effective at grasping.

Fingertip Strength (N)	**Curled**	**Open**	**Side**
Allegro Hand [7]	8.5	10.5	7.5
Inmoov Hand [22]	5.8	2.5	-
Bauer et. al [71]	37.4	-	-
DASH Hand [260]	34.5	8.4	6.2
LEAP Hand [339]	27.5	19.5	16.2
LEAP Hand v2 Adv	42.2	30.5	27.2

Table 7.1: **Strength Test:** We push against a curled and open finger on the front of the finger and the side of an open finger. The force applied when the angle error is more than 15 degrees is recorded. LEAP Hand v2 Adv is very strong in all directions due to its stiff MCP joints, internal bone structure, and efficient power transfer from strong motors. This is tested on the index finger, but all of the fingers are produced identically to simplify fabrication.

7.5 Fabrication and Software Details

7.5.1 Assembly Information

LEAP Hand v2 Adv can easily be built by a novice roboticist in a few hours and only requires simple hand tools and a 3D printer to assemble. It weighs about 1.25kg and can be mounted on most commodity robot arms. Upon acceptance, we will release the full assembly instructions and videos on our website along with the 3D printer files needed to recreate LEAP Hand v2 Adv in only a few hours.

7.5.2 Controlling Soft Fingers

The compliance of soft material makes it impossible to know the full state of it by using sensors such as encoders. In our system, we have two soft flexure joints in between the bones of the fingers that are actuated by one tendon on one pulley. The rigid internal structure of the fingers prevents the soft fingers from flexing too much in undesirable directions and makes the characteristics of our fingers more deterministic when not under extreme load. There are two parts to this control system: calibration and retensioning. In calibration, an AR tag is attached to the fingertip and track its location relative to the base joint. A simple second-order model for each of PIP and DIP that maps motor angle to finger joint positions. This only needs to be calculated once. However, since the material and tendon could change over time. To adjust to this, the current control mode for the motor is used to find the fully-tensioned and untensioned positions of the finger automatically and this offset is saved for test time.

7.6 Hand Analysis

We analyze many properties to characterize and predict LEAP Hand v2 Adv's performance. (1) How strong is our hand compared to many other robot hands available now? (2) How susceptible is our hand to hysteresis, or the degradation of soft material performance over time? (3) Finally, how does our hand perform on the Kapandji test?

7.6.1 Strength Test

Many soft hands, while they are strong in the actuated direction, have undesirable degrees of freedom in other directions such as twisting or compression. This is not ideal for a human-like robot hand. Robot hand fingers need to have strength at the fingertips in all actuation directions to manipulate the environment effectively.

In this test, we actuate the hand force gauge against the fingertip to see how much force they can resist. Certain hands lack some articulations or are excessively delicate to undergo the test which accounts for the absence of some data points.

In the results in Table 8.1, LEAP Hand v2 Adv is one of the strongest hands despite its small size and compliant nature which makes it very useful for a variety of tasks. The strong motors underneath the palm help our hand curl with a lot of strength. The bone structure inside the fingers and the palm helps keep the hand rigid. The MCP-side motors are directly on the fingers themselves which gives them strong, efficient power transfer.

7.6.2 Kapandji Score

The Kapandji Score for robot hands tests the ability of the thumb to oppose each finger and the rim of the pinky. [197, 380] We find that LEAP Hand v2 Adv can complete this task successfully with its articulated palm. However, without the articulated palm, it cannot complete the task and reach across to the pinky. To complete this test, many other robot hands allow the thumb's MCP joint at the base of the thumb to close more than a human can. This compensates for the fact that many robot hands are missing the CMC joint in the palm. However, this is not as human-like and does not lead to as good grasping positions and diversity as LEAP Hand v2 Adv.

7.7 LEAP Hand v2 Adv Applications

An important way to assess the capabilities is to assess our hand's performance directly at its end goal: How well can it complete tasks we expect robot hands to perform? First,

LEAP Hand v2 Adv is tested on grasping against a variety of other hands. Second, LEAP Hand v2 Adv is teleoperated by an expert operator using a Manus Meta glove LEAP Hand v2 Adv and a xArm. We choose a variety of different tasks that outline different grasps that humans can perform. Finally, we collected 75 demos through teleoperation on two different tasks. We train autonomous policies and see how LEAP Hand v2 Adv performs during this whole learning process which requires resiliency and accuracy.

7.7.1 Grasping Results

We evaluate the hand's capacity to execute various grasping techniques while holding objects. We demonstrate diverse hand grasps of different objects, showcasing their respective resistance to perturbation forces. The grasp types that we perform are inspired from Liu et. al. [237] To efficiently explore and identify these positions, we employ the Manus Meta VR glove [24] for precise teleoperation. Following the object's grasp, we apply force using a

Object	Grasp Type [239]	v2 Adv	LEAP	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	20	16	7	0	Y
Softball	Large Pad	20	20	10	15	N
Drill	Trigger Press	20	20	15	0	N
Pringle Can	Power Palm	20	19	20	0	Y
Pan Handle	Hook Grasp	20	7	8	14	N
Intermediate:						
Chopsticks	Tripod Grasp	15	16	0	0	N
Wood Cylinder	Cigarette Grasp	14	5	0	0	N
Precision:						
1" Cube	2 Finger Precision	15	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	16	8	0	N

Table 7.2: We test each robot hand on a variety of objects and grasp types from taxonomy to see how much perturbation force they can resist (in newtons up to 20). [239] The dexterous morphology of LEAP Hand v2 Adv as well as its strong motors enables tight grasps like the hook grasp as well as large grasps like the softball.

Hand	Grasp Type	LEAP v2 Adv	LEAP v1
Pickup Dice	Power	1.0	1.0
M&M	Pinch	0.6	0
Hammer	Hook	1.0	0.8
Baseball	Small Pad	0.8	0.6
Chopsticks	Tripod	1.0	0.6
Wooden Cylinder	Cigarette	1.0	0.8
Egg	Pinch	0.8	0.8
Pringles Can	Power	1.0	1.0
Pan Handle	Hook	1.0	0.6
Bin Picking	Pinch	1.0	0.8

Table 7.3: We teleoperate LEAP Hand v2 Adv and LEAP Hand (another easily obtainable robot hand) on 10 different tasks and evaluate their success rate over 5 trials. We find that LEAP Hand v2 Adv continuously outperforms it due to the variety of firm human-like grasps it can successfully perform.

gauge until slippage occurs or the force surpasses 20N. Due to InMoov's fragility, we solely evaluate its ability to grasp. We find in Table 7.2 that LEAP Hand v2 Adv can continually outperform on many different types of objects and grasps due to its human-like kinematics.

7.7.2 Teleoperation Results

Because the kinematic structure is so similar between the human and LEAP Hand v2 Adv, it is easy for the operator to teleoperate it. To achieve this, we use a motion capture glove from Manus Meta to operate LEAP Hand v2 Adv. The joint outputs directly from the motion capture glove from Manus Meta are used to operate the robot hand. While an energy function such as from Dexpilot [169] or Robotic Telekinesis [353] can be used, we find that this is not required due to the similarity of kinematics between the human and robot hand. The Ufactory xArm 850 robot arm is controlled by a SteamVR-based outside-in tracking system mounted on the wrist of the human operator.

To test the effectiveness of the system, human teleoperates the hand for 10 different tasks. These tasks are inspired by Vazhapilli et. al. [380], Mannam et. al. [260], [239] and [169]. These tasks are designed to test many different types of grasps, both prehensile and non-prehensile as well as power, pinch, precision, and many others.

Thanks to its strong articulated palm, LEAP Hand v2 Adv can complete teleoperation very well as seen in Table 8.2. For instance, the drill and hammer are very firmly grasped inside the palm and MCP joints of the palm and cannot be easily removed. The pinch grasp

Task	Teloperation	Behavior Cloning
Red Cup	0.92	0.8
Hammer Pickup	0.89	0.6

Table 7.4: We collect 75 demos of LEAP Hand v2 Adv completing two different tasks. We then train policies, pre-trained on internet videos, and fine-tuned on our robot hand demos, and report our results as a percentage of success.

is very strong.

The finger is very strong but also flexible. As in the video results, one can observe that the fingers are lifting very heavy weights and objects. The finger is also curling very tight to grasp small-diameter objects. However, when the finger is in contact with the table, it smoothly bends away from the table and does not break or snap. This is thanks to its robotic skin and inner-bone structure as outlined in 8.4.1. Please see the supplemental for videos and our website https://v2-adv.leaphand.com/ of these results and to observe these characteristics.

7.7.3 Learning from Human Demonstration Results

In robot learning, researchers often want to collect human demonstrations to train autonomous policies. Learning from demonstration puts difficult requirements on robot hardware. LEAP Hand v2 Adv must collect hundreds of demonstrations consistently without overheating, losing accuracy, or breaking. LEAP Hand v2 Adv must be resistant to bumps and scrapes while researchers quickly teleoperate the hand using strong robot arms. LEAP Hand v2 Adv must be able to perform even with non-smooth and jittery behavior cloning policies.

To test this we collect 75 demonstrations per task through the teleoperation of LEAP Hand v2 Adv and the xArm. These tasks are lifting a heavy hammer and picking up a red cup. Thankfully, in the data collection process, we do not observe any degradation of the behavior. Once the policies were trained, they were rolled out on the robot. In Table 7.4 and our website at https://v2-adv.leaphand.com/ we see that LEAP Hand v2 Adv continually performs well.

7.7.4 Kinematic Simulation Modeling

In robot learning research, having an accurate kinematic model is essential, as simulation is frequently used to debug ideas, test control strategies, or explore retargeting approaches.

To support this, we create a URDF and MJCF that can be imported into MuJoCo or Isaac Gym. Simulating hybrid soft-hard finger materials is challenging, but the soft material used (Shore hardness 85A) is similar to leather or tire treads. Combined with PLA, it behaves like a rigid hand with grip tape. Thus, despite Isaac Gym's limited soft material support, the finger joints can be effectively modeled as rigid body rotational joints, like many robot hands.

Chapter 8

Demonstrating LEAP Hand v2: Low-Cost, Easy-to-Assemble, High-Performance Hand for Robot Learning

8.1 Abstract

Replicating human-like dexterity in robotic hands has been a long-standing challenge in robotics. Recently, with the rise of robot learning and humanoids, the demand for dexterous robot hands to be reliable, affordable, and easy to reproduce has grown significantly. To address these needs, we present LEAP Hand v2, a $200 8-DOF highly dexterous robotic hand designed for robot learning research. It is strong yet compliant, using a hybrid rigid-soft structure that is very durable. Its universal dexterous MCP joint provides exceptional finger mobility, enabling a variety of different grasps. The parts are all 3D printed and can be assembled very easily in under two hours using our instructions. Importantly, we offer a suite of advanced open-source software tools to support robot learning research. This includes human video retargeting code from MANO and Vision Pro, motion capture teleoperation code using the Manus Glove, and a URDF with simulation examples for various simulation engines. We will showcase LEAP Hand v2—designed specifically for this demonstration—alongside our previous robot hands with real robot interactive demos. Following our successful demos at RSS 2023 and 2024, we will again offer an engaging opportunity for attendees to get hands-on experience and information about the accessibility of low-cost, open-source robotic hands.

Figure 8.1: Our demonstration will feature four different low-cost, open-source robotic hand designs, including a new model introduced in this paper, LEAP Hand V2. (left) These hands are highly dexterous, easy to build, durable, and affordable. We also provide a suite of open-source software tools, including motion capture teleoperation, human video-based learning, and reinforcement learning capabilities. Building on our successful demonstrations at RSS 2023 and 2024, we aim to further highlight the potential of low-cost, open-source robotic hands and strengthen our open-source robot hand community at RSS 2025. Please visit our website at https://leaphand.com for more details.

8.2 Introduction

Think about activities such as typing on your keyboard, hammering a nail, or using chopsticks, and you'll realize the pivotal role our hands play in manipulating the world. With remarkable strength at the fingertips, capable of over 70 different pinching and grasping motions, our hands possess unparalleled abilities to manipulate. This extraordinary sensing and adaptability are orchestrated by both our hand "hardware" itself as well as the impressive capabilities of our brains. The development of our brains is often linked to the necessity of manipulating our surroundings with our hands [77, 157].

In the realm of robotics, manipulation has predominantly relied on claw grippers or suction cups for pick-and-place tasks in factories. However, the collective aspiration is to witness humanoid robots coexisting with humans, undertaking similar tasks in similar environments. The absence of robot humanoids with efficient and low-cost robotic hands

raises the question: Why haven't they become a reality?

One major bottleneck is that while there are a few robot hands are available today, the prevailing opinion is that they are challenging to use, expensive, and difficult to acquire. The belief has been that the human kinematic structure and strength is difficult to produce in robot hands. Some robot hands are too large, some have fewer degrees of freedom and other are extremely difficult to produce and maintain. We believe this isn't necessarily an inherent flaw in robot hands but rather a consequence of not designing them ideally for machine learning research.

To break through this prevailing belief, we introduce LEAP Hand V2: Low-cost Easy-to-Assemble high-Performance robot hand for robot learning. LEAP Hand V2 is our latest robot hand designed for this demonstration that has these characteristics:

1. **Hybrid Rigid-Soft Hand:** Combines flexibility and compliance with exceptional strength and durability in each finger.
2. **Universal MCP Joint:** This universal abduction-adduction mechanism enables extreme dexterity for many tasks.
3. **Extreme Low-cost and Reproducibility:** At $200 and with under an hour of assembly time, it makes the hand readily available for many robot learning researchers.

Robot hands designed for machine learning require both a physical design and advanced software tools to support research. By developing open-source tools, the goal is to enable the community to build upon and iterate these tools for a variety of applications and reshare them with the community. Our open-source tools serve as a solid foundation for advancing machine learning research:

1. **Motion Capture Teleoperation:** Motion-capture technologies, such as the Manus Meta Glove, enable highly precise teleoperation of the LEAP Hand v2 with any robot arms.
2. **Learning from Human Video:** A wide range of freely available human video sources can be used to teach robot hands like LEAP Hand v2 to mimic human-like behaviors.
3. **Simulation:** A kinematically accurate simulator allows us to perform both forward and inverse kinematics and geometric grasping analysis.

In summary, we present a new robot hand designed specifically for this demo, LEAP

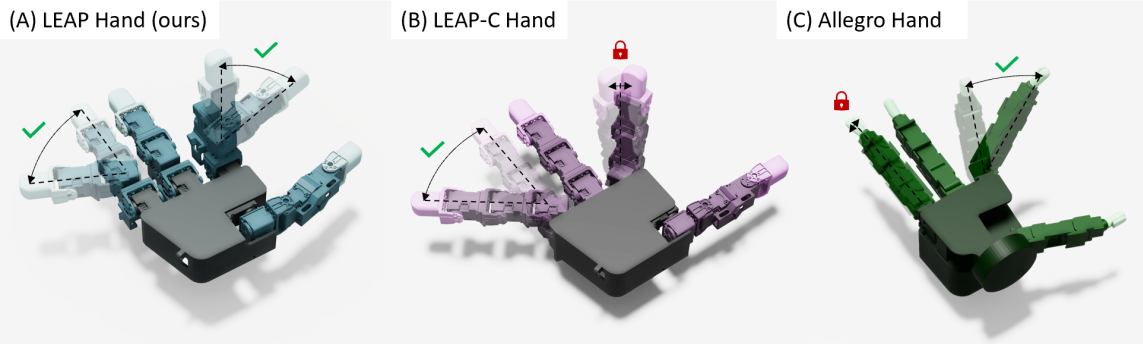

Figure 8.2: In all of our LEAP Hands we introduce a MCP joint that allows for abduction and adduction in both flexed and extended positions. In this figure we show LEAP Hand v1 on the left, where this dexterous kinematic structure is most apparent. In a conventional robot hand, LEAP-C Hand, the finger can move side to side in the open-palm, but in the flexed position it only spins in place. In Allegro, there is a large of motion at flexed but not in the extended position. [339]

Hand v2, that is affordable at under $200 and incredibly easy to produce and assemble in under 1 hour. It is extremely dexterous, durable and with a human-like size. In this RSS demo, we show its ability to handle a variety of machine learning applications such as teleoperation and real-world reinforcement learning. These tools are released on our website for the community to use. These RSS demos have played a crucial role in the success of open-source robot hands and inspire attendees to incorporate open-source, dexterous robot hands into their own manipulation projects. They have captivated conference attendees, even those not interested in dexterous manipulation. We look forward to the opportunity of showcasing VideoDex and our other robot hands at RSS 2025 and on our website at https://leaphand.com

8.3 Related Work

Dexterous Robot Hands Many robotic hands have been developed to mimic the capabilities of the human hand, with varying degrees of success and accessibility. The MIT/Utah Hand has an early tendon-driven design [188]. [97, 98, 137] used variable stiffness actuators and soft materials in their IIT hands. [135] used a novel compliant actuator and [91] developed this area of tendon-driven hands further. [98, 250] use underactuation similar to our hand to enable high dexterity without using too many actuators which makes the hand large and heavy. A dexterous all-soft hand, with palm articulations in a completely soft structure, is presented in [243].

While these robot hands are very advanced, they are difficult to reproduce and obtain for many research labs. There are now a few robot hands for purchase. The Shadow hand,

as documented in [5, 214], has demonstrated remarkable achievements such as in-hand reorientation of a Rubik's cube [54]. Despite its impressive performance, the Shadow hand is widely acknowledged for its high cost (approximately $150k) and challenging usability. Conversely, the Allegro Hand [7, 216], has been historically recognized as a more affordable option priced at $20k. However, it is often criticized for its tendency to break down and the associated difficulties in repair. Nevertheless, the Allegro Hand has showcased commendable capabilities, including teleoperation from video [169, 260, 306, 338, 352], as well as in-hand reorientation [170]. The Psyonic Ability Hand, designed as a prosthetic with a robust internal hard skeleton and soft exterior but only possesses 6 degrees of freedom (6DOF) [4]. Inspire Hand is lower in costs but is not durable and often breaks[23]. The Faive Hand [373] demonstrates noteworthy sim2real results in in-hand reorientation but is not readily available yet.

A resurgence of interest in humanoid robotics from industry players like Tesla Optimus [28], Figure [18], BD Atlas [9], 1x [27], Sanctuary AI [36], and Digit [14] has been observed. These hands are designed for strength and mass production to handle daily tasks for humanoids. However, they often feature limited degrees of freedom and are not readily available for purchase, evaluation, or research purposes.

The emergence of rapid-prototyping technologies, such as 3D printers and CNCs, has led to the development of a plethora of low-cost, open-source hands tailored for academic research purposes. The LEAP Hand, detailed in our papers [49, 339] is easy to use and has been used by many research labs around the world. The Robel suite, exemplified by D'Manus, offers large yet durable hands employed in tasks such as reorientation [109] and grasping [76]. Other hands, such as Inmoov [22] and DexHand [13], cater to hobbyists but may be limited by inexpensive motors or fragile plastic components.

Rapid Manufacturing The traditional method for creating robust components typically involves machining, such as with aluminum, which can be costly. Plastic parts, on the other hand, are generally produced through a process that includes mold creation, casting, curing, and support removal, making it suitable for large-scale production [30]. In contrast, 3D printing has transformed small-scale manufacturing by enabling the rapid, autonomous printing of individual parts [34, 71].

In recent years, the 3D printing industry has made substantial progress in material innovation. Flexible materials like TPU/TPE from Ninjatek and Filaflex have introduced new possibilities for creating more adaptable components [1, 3]. Foaming materials such

as Colorfabb Varioshore and Recreus Filaflex [1, 41] allow for the adjustment of material properties by modulating the flow rate. Additionally, the use of materials like Nylon and carbon fiber in 3D printing has resulted in components with enhanced strength and durability. As a result, consumer-friendly multimaterial 3D printers have become both affordable and widely accessible.

Learning for Dexterous Manipulation Because of the high dimensionality of dexterous manipulators, it is different to used traditional model-based controls or planning methods to achieve dexterous results. [225, 279] In robot learning Andrychowicz et al. achieved in-hand rotation for various objects using a Shadow hand and Sim2real techniques. [54, 56]. Simulation-based training that scales to thousands of objects is explored in works such as [49, 107, 170, 180, 211] which shows promise in robot learning. D'Hand is utilized by Nair et al. to reposition a valve [273]. Other notable instances of dexterous manipulation include Baoding Balls' in-hand rotation using the Shadow Hand trained exclusively in the real world [271].

Recent studies highlight the importance of supervising robot hand policies based on human actions, such as those derived from MANO [317] human hand parameters. Related work includes teleoperating robot hands through real-time video [169, 352], which can assist in learning [304, 311, 338]. Hand poses extracted from online video data are utilized for learning manipulation policies [254, 304]. Large-scale pre-training using internet videos has proven effective for training robot hands efficiently for downstream tasks with minimal task-specific demonstrations [61, 196, 338], and this approach extends to non-dexterous manipulation tasks [67, 281].

8.4 LEAP Hand v2

The LEAP Hand v2 is a low-cost, highly dexterous robotic hand designed specifically for robot learning. It combines soft and rigid materials through multi-material 3D printing to achieve human-like compliance and durability, while keeping the design simple and easy to reproduce. With underactuated joints, a novel MCP mechanism, integrated tactile sensing, and an intuitive assembly process, the hand is purpose-built for machine learning workflows. This section details the hardware innovations that enable LEAP Hand v2 to serve as a practical and accessible platform for robotic manipulation research.

8.4.1 3D printed Hybrid Rigid-Soft Hand

The flexibility of human hands enables them to adapt to objects and their surroundings during interactions like grasping [278, 418]. For example, when reaching for a fragile object, the hand molds around it and applies gentle pressure. Similarly, when encountering an obstacle like a table, our fingers bend away without breaking. To replicate this, the soft robotics field often employs a casting method to create soft robots [185, 252]. However, this approach can result in robot hands being excessively compliant in under-actuated directions, which is not ideal. This makes the hand weak and objects can slip out. In contrast, some robot hands, like the LEAP Hand V1 or Shadow Hand, consist entirely of rigid joints. While these hands can impart a lot of force, they lack the ability to conform to their environment as a human hand does. This leads to brittleness and a tendency to snap upon contact. Finally, other designs, such as those in [204, 373], attempt to replicate the softness of human hand joints but are highly complex and difficult to reproduce.

Our goal is to fabricate conformal fingers with stiffness properties closely resembling those of a human finger, aiming to replicate both its softness and rigidity. We aim to ensure our robotic hand is durable and capable of withstanding the rigors of manipulation while also being easy to produce. The careful selection of materials and a suitable 3D printer is essential to achieve this hybrid rigid soft hand finger as seen in Figure 8.3.

For the soft outer skin, Recreus Filaflex Foamy is chosen due to its strong restitution and its ability to foam up buy taking in air from the environment when leaving the nozzle. This enables one to adjust the TPU rubber density based on the flow rate and temperature of the 3D printer nozzle. [41] Underneath the skin and within the fingers, we opt for PLA, known for its rigidity and smooth texture. The 3D printer used is an Independent Dual Extrusion (IDEX) printer, specifically the $1000 Snapmaker J1S [37]. This printer allows for the simultaneous use of two materials automatically in one seamless print. The realization of these materials together in one seamless print includes intricately intertwining the soft and rigid materials in CAD and the slicer software.

These material properties are used in a few key areas. The exterior layer of the palm's skin is crafted with denser material to resist cuts and bruises. The exterior of the fingers are created using denser high-flow Foamy material which makes them resistant to bruises. The interior of the finger between the flexure joints is printed with PLA bones which resist undesirable twisting and compression of the fingers as seen in Figure 8.3. This makes the

fingers very strong. The internal flexure joints themselves are created with lower-density soft material to enable easy flexure actuation. The strength of each flexure joint in the fingers (MCP, PIP, DIP) are carefully tuned relative to each other by modulating the density as explained in the next section.

Underactuated Curling

A key factor in designing a robot hand is determining the appropriate number of underactuated and fully actuated degrees of freedom (DOF) the robot hand should have. Many works such as [99, 250] use underacted DOF to extend the capabilities of these manipulators. Recent hands designed for humanoids keep the number of motors very small to make the hands light and durable [23]. VideoDex is designed with 8 motors, two for each finger—one for curl and one for abduction—which contributes to the hand's light weight and simplicity in construction. This configuration results in substantial underactuation, with a single motor controlling three curling joints (MCP Forward, PIP, DIP) on each finger simultaneously as shown in Figure 8.4. However, after further analysis, we have identified methods to manage and adjust the underactuated behavior to improve its suitability for grasping.

Consider this example: when the curl motor activates the pulley, it shortens the tendon to a new length, applying a force F on the finger to achieve this movement. The flexure joints in the finger then deform to distribute this force from the motor. An important characteristic of these flexure joints is that they are initially easy to close, but as they approach their fully closed position, they increasingly resist additional force making it harder and harder to curl.

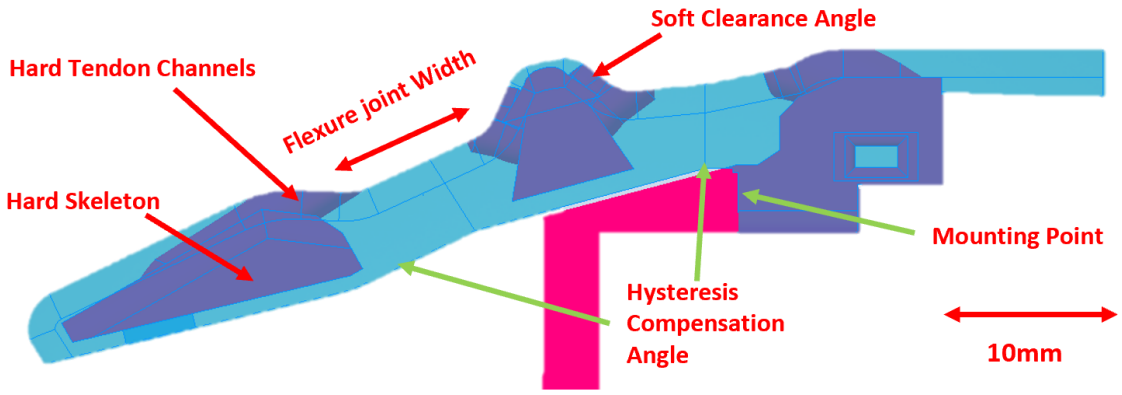

Figure 8.3: **Finger Analysis** We show the key dimensions that create our soft-hard hybrid flexure joint as described in Section 8.4.1. The flexure joint is soft material that lies between the hard skeleton. It is governed by the flexure joint width, the clearance angle, the hysteresis compensation angle and the size of the hard skeleton. The darker portions in the finger is the hard tendon channels and rigid structure. The blue material encompassing that is the soft material of the finger itself.

Suppose each joint in the finger is treated equally. In this scenario, each joint would experience the same amount of movement, with one-third of the tendon shortening force being distributed to each joint. As the finger curls, the motor increasingly exerts more effort to achieve the motion. However, this uniform motion is not ideal. In manipulation studies of the human hand, the MCP joint typically curls with a greater magnitude than the PIP and DIP joints to grasp over the palm and around objects

To replicate this behavior, our key insight is that each joint in the robot hand can be designed with different strengths. For instance, a weaker joint will generate less resistive force throughout its range of motion and will therefore actuate more. This means that a weaker joint will move further with the same tendon force compared to a stronger joint that is coupled with it. As the actuation force increases through the range of motion, the stronger joints will still actuate, but to a lesser extent relative to the weaker joint. The ratio of strength between the weaker and stronger joints determines the proportion of actuation across the different finger joints.

There are various ways to weaken a joint, one of the most straightforward being to print it thinner. However, thinning the joint makes it more vulnerable to twisting and compression, which are undesirable movements that reduce the finger's manipulation capabilities. Instead, we opt to adjust the flow rate of each hinge in the finger. By modifying the flow rate, we can easily control the relative density of the joints which affects strength of the joints while minimizing the risk of twisting and compression.

The MCP joint is designed using the lowest density material, followed by the PIP joint, and then the DIP joint. This design ensures that the MCP joint experiences the greatest actuation, with progressively smaller amounts of actuation for the PIP and DIP joints. By carefully selecting materials with varying densities for each joint, we can fine-tune the actuation behavior to better replicate natural human finger movement, where the MCP joint typically curls more significantly than the PIP and DIP joints.

A potential drawback is that a weaker joint is more compliant or less strong in the actuated direction, as an external disturbance allows the joint to move further with the same amount of force. This effect is particularly noticeable when the finger is open, as the flexure joint and tendon are not under tension. Wen the finger is curled, the tension still means that the DIP joint requires more force to actuate and becomes the most resistant to the applied force.

This effect is mitigated in two ways. First, since the tendon remains at approximately

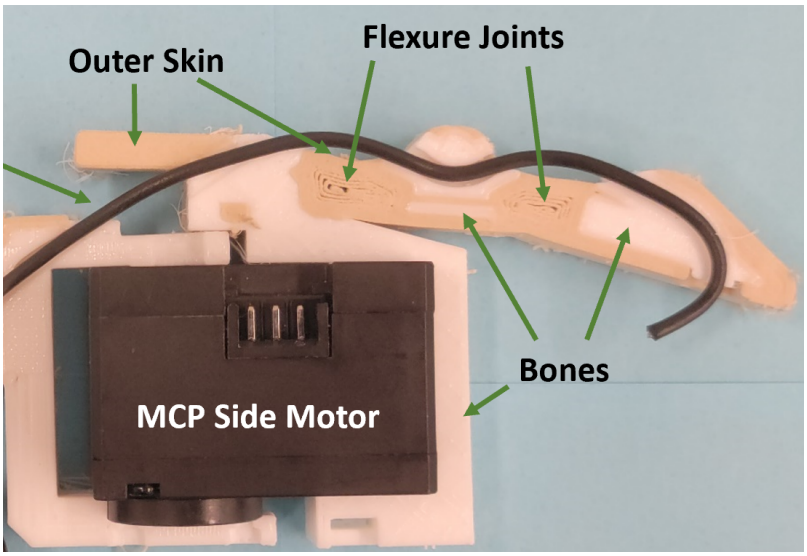

Figure 8.4: A cross-section of the 3D printed finger showing the soft rubber joints, hard PLA bones, and resilient, dense outer skin. The MCP side rotates using an embedded motor, and the PIP and DIP joints are actuated together by a tendon connected to a pulley.

the same length from the motor when an external disturbance is applied, pushing on the MCP joint causes the PIP and DIP joints to curl more in compensation. This behavior is actually beneficial, as it enables the finger to curl and conform more effectively around the object being grasped without crushing it.

Secondly, the motor's controller is adjusted. A simple approach is to use current control, which enables the motor to apply a constant force to the robot's finger. However, this method has a drawback: it allows for greater compliance because the motor doesn't pull hard to maintain tendon length when external forces are applied. This weakens the grip. Position-based PID control offers a better solution by enabling the motor to resist external disturbances. This approach allows the motor to actively reduce compliance, applying more force when the tendon is pulled. As a result, the position controller stiffens the finger joints, improving the finger's ability to grip heavier and larger objects.

While it is possible to incorporate external position sensors and set position targets for the fingertips directly, this approach is costly and challenging to implement on a low-cost, easily produced 3D-printed hand. Furthermore, the inclusion of such sensors and their control policy could result in reduced compliance for the fingers, potentially counteracting the desired flexibility.

8.4.2 Dexterous MCP Joint

Off-the-shelf motors impose limitations on kinematic structure choices, making it challenging to accurately replicate the MCP ball joint of the human hand. As a result, it is typically approximated using two motors (MCP-1, MCP-2) positioned close together [215]. Previous work has proposed two designs for this configuration (Fig. 8.2). However, both the Allegro and LEAP C-Hand designs sacrifice one degree of freedom either in the extended or closed position. Consequently, the Allegro hand is less dexterous when extended, while the LEAP C-Hand (similar to the C-Hand in [215]) 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 8.2 for a visualization of this deficiency for these baseline hands.

In LEAP Hand v2, a **universal abduction-adduction mechanism** is used for the fingers, ensuring that dexterous motion is maintained at every MCP position. Rather than fixing the MCP-2 axis to the palm (i.e., the motor responsible for adduction-abduction), the key innovation is to align the axis after the first finger joint and orient it so that it remains perpendicular to the finger at all times. This design enables the finger to achieve adduction-abduction across all positions as shown more visibly on LEAP Hand V1 in Fig. 8.2. As a result, VideoDex provides the best of both worlds and has both adduction-abduction in the extended position (similar to the LEAP C-Hand) and pronation/supination in the flexed position (similar to the Allegro).

8.4.3 Purpose built for Robot Learning

For machine learning research, a robot hand must not only be easy to assemble, maintain, and modify but also designed with flexibility in mind to accommodate rapid iterations and experimental changes. Given the iterative nature of machine learning development, having a hand that is low-cost and robust is crucial to minimize the barrier to entry and enable frequent updates without significant downtime or expense. Moreover, a highly dexterous design ensures the hand can perform a wide range of tasks, making it adaptable to various learning scenarios. Achieving these goals requires a thoughtful balance between functionality and

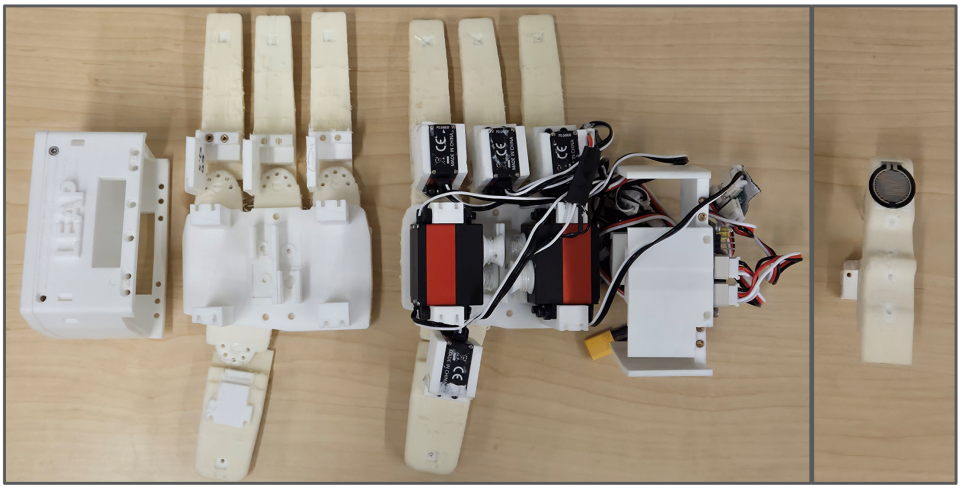

Figure 8.5: The multi-material components, shown on the left, are fabricated using a dual-extrusion 3D printer and are designed for straightforward assembly in only 1 hour. The eight actuators are inserted into designated slots within the palm and fingers, followed by basic tendon routing and wiring to complete the setup. To facilitate widespread adoption, we release all print files, assembly instructions, and accompanying software as open-source materials at https://leaphand.com, enabling rapid replication by the research community. Additionally, off-the-shelf tactile sensors can be affixed to the fingertips to enhance sensing capabilities. (right)

simplicity, ensuring the system remains reliable while offering the versatility needed for research.

To start, we have reduced the total number of parts and ensured that all components used are easy to source. By utilizing a multi-material printer, we can create complex parts with multiple integrated components in a single print. For instance, the palm piece combines a soft top, a rigid undercarriage, tendon channels, motor mounts, the MCP forward joint, and an attachment for the MCP side servo horn—all in one simple, easy-to-print part. Similarly, the finger is printed as a single piece, incorporating the PIP and DIP joints, tendon channels, and motor mounts. This streamlined design reduces the overall part count (fewer than 10 3D-printed parts in total) and simplifies the assembly process, making it more manageable for the end user as seen in Figure 8.5.

For motor selection, we opted for Feetech Bus-Based Current-Limited control servos. These servos are cost-effective (under $30 each) and easily accessible. They are simple to program, and their internal PID loop parameters can be customized for both position-based current-limited control and current-control modes. Additionally, the built-in sensors provide useful data, such as current and position readings. This functionality is comparable to the more expensive Dynamixel servos used in previous versions of the LEAP Hand and other robotic hands, but at a significantly lower cost and in more suitable sizes for this robot hand.

A common challenge with tendon-based robot hands is the need for frequent retensioning

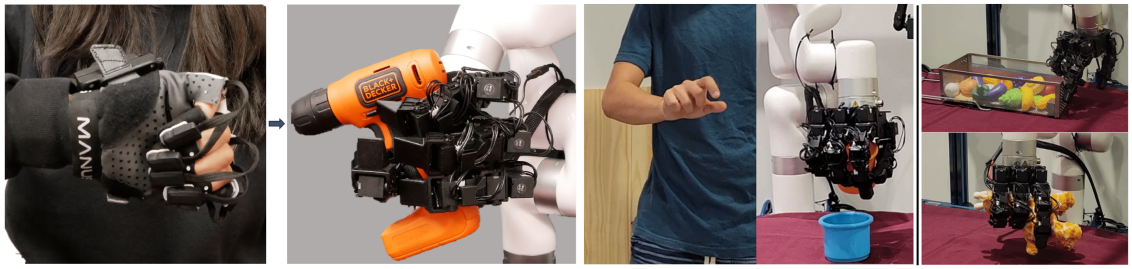

Figure 8.6: We will demonstrate real robot hands doing teleoperation from VR glove and teleoperation from human video as developed in [49, 339, 352] Attendees will be able to teleoperate a variety of low-cost robot hands such as LEAP Hand V1 shown in this figure or LEAP Hand v2 as discussed in the paper.

and replacement of worn tendons. However, our design uses only one tendon per finger, and the pulleys are securely housed within channels in the palm, preventing the tendons from slipping off the motors. Unlike designs like the Shadow Hand [5] or DASH Hand [260], which require tensioning of tendon pairs, our system simplifies calibration, allowing it to be easily adjusted through automated software.

The assembly process is designed to be simple, with no complicated steps or hard-to-reach components. The motors are easily accessible and can be installed with just a screwdriver and a basic knot on the fishing line. Thanks to the intricate multi-material 3D-printed parts, the number of screws needed for assembly is minimal. While LEAP Hand v1 requires nearly 300 screws, VideoDex uses fewer than 50 screws.

We make this comprehensive documentation available on our website at https://leaphand.com It includes an updated shopping list, detailed assembly instructions, and all the example code necessary to build VideoDex. This will enable anyone to easily replicate the robot hand and quickly begin working on their own projects.

8.4.4 Tactile Sensors

Since LEAP Hand v2 is user-friendly and affordable, we aim to ensure that the tactile sensors are equally easy to use, cost-effective, and capable. However, there is a significant challenge. Vision-based sensors like Gelsight or Digit, which rely on cameras and a gel-based touch surface, are too large for our needs. Similarly, sensors such as ReSkin, AnySkin, and Xela are designed for larger fingers. While they could theoretically be miniaturized, they often face interference issues when placed too closely together, as in a smaller hand.

For our hand, we have chosen resistive-based sensors similar to STAG [363] or 3D-ViTac [179]. These sensors are affordable ($2 each), and a LEAP Touch control board ($20) can

be mounted on the palm of the hand. While they provide only a single continuous force reading per finger, this is adequate for many machine learning applications. Additionally, these sensors can be mounted on Manus glove [24] to collect data both hand tracking and touch data in a format that closely mirrors what the robot hand will encounter.

8.5 Software for Robot Learning

VideoDex is designed specifically for robot learning, both in terms of the aforementioned hardware and the software which we describe below. The software package we release on our website includes all the essential tools that researchers need to get started quickly. First, teleoperation using motion capture gloves is a key feature for collecting high-quality data for behavior cloning and related approaches. Retargeting human video data, easily captured with web cameras or the Apple Vision Pro, is useful for control and enhancing autonomous policy training. Finally, we offer simulation tools for forward and inverse kinematics as well as kinematic geometric analysis. These software tools are crucial for VideoDex which designed for machine learning. These tools are available on our website at https://leaphand.com.

8.5.1 Learning from Human Mocap Glove Demonstrations

In conventional 2-finger gripper manipulation many teleoperation setups have worked successfully to collect demonstrations for use in behavior cloning. Kinesthetic Methods such as ALOHA [414], GELLO [393] or Da Vinci machines [299] can work accurately. With VR or camera-based hand tracking, methods such as [187, 257] work reasonably accurately. However, it is unclear how to scale these methodologies to robot hands.

Our key insight is to take inspiration from the motion-capture community. The motion capture community has been using gloves for accurate tracking for movie production or game production. These motion capture gloves often rely on EMF sensors and are relatively not bulky to the user. However, the data returned is in the human hand morphology which is not the same as the robot hand kinematics.

If the robot hand's kinematic structure is roughly similar to that of a human, one approach is to map the joint angles from human fingers to the robot's. While gloves can calculate these human hand angles using an inverse kinematics solver, differences in finger

size and proportions can lead to misalignment, especially with the complex human thumb and the differing kinematics that robot hands have. This can cause inaccurate pinch grasps which makes accurate fine-manipulation difficult.

Previous work has addressed this by optimizing joint positions for consistent pinch grasps between human and robot hands. [169, 345, 385] We employ Manus gloves [24] with an inverse kinematics approach to ensure accurate pinch grasps and proper thumb positioning, improving manipulation reliability.

We have collaborated with Manus Meta on our full-featured open-source Python/ROS2 repository that converts the MANUS data to any robot hand that has a URDF. We have examples for all LEAP Hands and are the main repository for these motion capture gloves for robotics.

Using this software to collect data, behavior cloning policies can be trained to complete a variety of different tasks that during the event such as tool use or deformable manipulation. We will show these policy rollouts, augmented with human-video pretraining on our robot hands. However, teleoperation data does not scale to novel environments so it must be collected in large quantities in many different environments. Our key insight is to leverage data from the web to enable further generalization which we will explain in the next section.

8.5.2 Learning from Human Video

Collecting teleoperated demonstrations is both costly and time-consuming, and it is impossible to capture all potential scenarios the robot may encounter. This leads to a distribution shift, where the robot's testing environments will inevitably differ from its training environments. While one might suggest that robots can adapt or generalize to unseen scenarios, this is not guaranteed. Therefore, a key question is how can we effectively expand our training set without resorting to additional laborious teleoperated data collection?

Given that robot hands share a similar embodiment and kinematic structure with the human hand, they both will naturally interact with the world and perform tasks in much the same way. By utilizing this similarity, we can learn from extensive human motion data, such as video and motion capture datasets. However, converting from human hand demonstrations to robot hand demonstrations is a non-trivial problem.

This conversion to the robot hand is particularly challenging due to the under-constrained nature of the problem, as robot hands and human hand possess numerous degrees of freedom

(DOF) and exhibit substantial differences in shape, size, and joint structure. The retargeting process must cater to any human operator attempting to execute various tasks in diverse environments. Additionally, an essential criterion is the efficiency of the solution, demanding a real-time performance at a rate exceeding 30 Hz. Unlike the glove data, this data can be exceptionally noisy and difficult to use. To handle these issues, we develop a video-to-robot hand retargeting system that is trained from a corpus of rich and diverse human hand videos. The system understands human hands and retargets the human video stream into a robot hand-arm trajectory that is smooth, swift, safe, and semantically similar to the guiding demonstration. This methodology has a few stages. First, we detect the human hand in the image by using a state of the art hand detector such as FrankMocap. [320] Then, we retarget this robot hand pose using a NN trained on an energy function and human data. This ensures that the retargeted robot hand poses are human like and semantically similar to the human hand demonstration. These elements are combined together to teleoperate the robot hand and arm in Robotic Telekinesis [352] from RSS 2022. See Figure 8.6 for an example.

We also show that these internet videos can directly support the learning of autonomous policies, rather than just assist with teleoperation. To achieve this, we can apply this Telekinesis retargeting pipeline to the EpicKitchens dataset [122] to extract actions performed by human hands and arms to the robot embodiment. This process transforms human data into a format compatible with robot embodiment data. Then, this enables us to efficiently teach robot behaviors from these human videos using behavior cloning and co-training with teleoperation data. Details of this system is available on our website at https://leaphand.com

8.5.3 Simulation Tools

In robot learning applications, a simulation model of these hands is useful in a variety of applications. The kinematic model of the hand enables forward/inverse kinematics calculations for retargeting approaches such as for teleoperation from human video. It also enables geometric methods such as for grasping.

8.6 Analysis

A common question when starting out in robot learning is, "Which robot hand should I use?" While there are quite a few choices, many robot hands are not easy to acquire [77, 373].

Fingertip Strength (N)	**Curled**	**Open**	**Side**
Allegro Hand [7]	8.5	10.5	7.5
Inmoov Hand [22]	5.8	2.5	-
Bauer et. al [71]	37.4	-	-
DASH Hand [260]	34.5	8.4	6.2
LEAP Hand [339]	27.5	19.5	16.2
VideoDex	32.2	20.5	17.2

Table 8.1: **Strength Test:** We apply force to both the curled and open parts of the finger, as well as the side of an open finger. The force is measured when the angle error exceeds 15 degrees. VideoDex is highly powerful in all directions, thanks to its rigid MCP side joint, internal rigid bone structure, and efficient power transfer from robust servo motors.

There are others which are easier to acquire but still very expensive [5, 7, 23]. There are even fewer that are open source as well as being easy to produce for machine learning research. Selecting a robot hand involves more than simply considering performance metrics. **In our live demo this year, we will showcase these robot hands so that attendees can have hands on experience with all of these robot hands. This will help demystify this choice and is a key enabler of driving the open-source robot hand community.**

8.6.1 Strength Test

Many soft hands, while strong in the actuated direction, often have undesirable degrees of freedom in other directions, such as twisting or compression. This makes them less suitable for a human-like robotic hand. For effective manipulation of the environment, robot hand fingers need to exhibit strength at the fingertips across all actuation directions.

In this test, we apply a force gauge to the fingers to measure how much force they can withstand without failing to hold their desired position. Some hands lack certain articulations or are too delicate to undergo the test, which is why some data points are missing.

As shown in the results in Table 8.1, VideoDex stands out as one of the strongest hands despite its small size and compliant nature, making it highly versatile for a wide range of tasks. The powerful motors beneath the palm enable the hand to curl with substantial strength. Meanwhile, the internal bone structure in the fingers and palm ensures rigidity, and the MCP-side motors, positioned directly on the fingers, provide efficient, high-strength

Hand	Grasp Type	LEAP V2	LEAP
Pickup Dice	Power	1.0	1.0
M&M	Pinch	0.4	0
Hammer	Hook	1.0	0.8
Wooden Cylinder	Cigarette	1.0	0.8
Egg	Pinch	0.8	0.8
Pringles Can	Power	1.0	1.0
Pan Handle	Hook	1.0	0.6
Bin Picking	Pinch	1.0	0.8

Table 8.2: We teleoperate VideoDex and LEAP Hand V1 on various different tasks and evaluate their success rate over 5 trials. We find that our new hand can still complete these grasps successfully.

power transfer.

8.6.2 Teleoperation from Motion Capture

Oftentimes we would like teleoperation to be as accurate as possible for robot hands. To do this the user wears two motion capture gloves and moves naturally to complete everyday dexterous tasks. The gloves captures accurate finger tracking to map motions naturally to the robot hands. The GELLO [393] inspired arm tracking accurately tracks the human wrist position and joint angles for the robot arm. The data collected by the system can be used to train effective policies using imitation learning, after which the bimanual robot system can perform the task autonomously. In Table 8.2 we find that VideoDex can consistently perform well due to its small size and high levels of dexterity. Its soft construction makes is very durable to bumps and scrapes with the environment. We release videos of our results and instructions to recreate the setup on our website at https://leaphand.com **Demo attendees will have the opportunity to teleoperate a live robotic hand system and ask questions about how to build a similar setup themselves.**

8.7 Conclusion

In conclusion, we introduce LEAP Hand v2, an affordable, highly dexterous, and easy-to-assemble robotic hand designed specifically for robot learning research. Priced at just $200 and with an assembly time of under one hour, it combines a hybrid rigid-soft structure for durability and strength, a universal MCP joint for exceptional dexterity, and a human-like size. Alongside the hand, we offer a suite of open-source software tools, including motion

capture teleoperation, learning from human video, and reinforcement learning capabilities, all aimed at advancing machine learning applications. Through previous demonstrations at RSS 2023 and 2024, we have shown the potential of low-cost, open-source robotic hands to inspire and support the robotics community, and we look forward to showcasing these tools again at RSS 2025 to further encourage the use of open-source dexterous robot hands in various manipulation tasks.

Chapter 9

Bimanual Dexterity for Complex Tasks

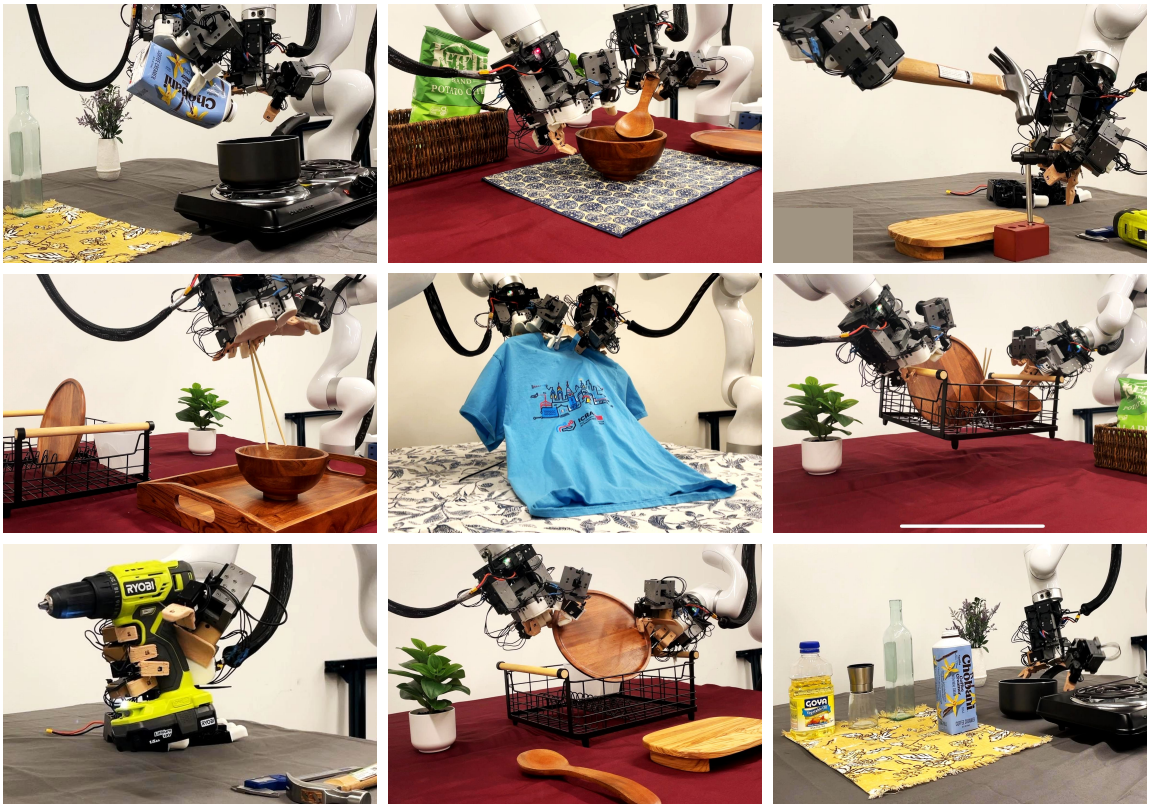

Figure 9.1: **Bimanual Dexterity**: BiDex can effortlessly teleoperate various complex tasks including pouring, scooping, hammering, chopstick picking, hanger picking, picking up basket, drilling, plate pickup and pot picking to train high-quality behavior cloning policies with over 50 degrees of freedom. Our teleoperation system and two LEAP hands [339] costs around $12k in total and is readily reproducible by academic labs.

9.1 Abstract

To train generalist robot policies, machine learning methods often require a substantial amount of expert human teleoperation data. An ideal robot for humans collecting data is one that closely mimics them: bimanual arms and dexterous hands. However, creating such a bimanual teleoperation system with over 50 DoF is a significant challenge. To address this, we introduce BiDex, an extremely dexterous, low-cost, low-latency and portable bimanual dexterous teleoperation system which relies on motion capture gloves and teacher arms. We compare BiDex to a Vision Pro teleoperation system and a SteamVR system and find BiDex to produce better quality data for more complex tasks at a faster rate. Additionally, we show BiDex operating a mobile bimanual robot for in the wild tasks. Please refer to https://bidex-teleop.github.io for video results and instructions to recreate BiDex. The robot hands ($5k) and teleoperation system ($7k) is readily reproducible and can be used on many robot arms including two xArms ($16k).

9.2 Introduction

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

With the advent of data-driven machine learning methods and low-cost hardware there has been renewed interest in humanoids and dexterous hands. There is great promise for machine learning approaches to enable effective autonomous control for high-dimensional robot systems using large amounts of data [54, 183, 193, 211, 219]. A key question remains: how do we collect high-quality expert data for bimanual robots? Such a data collection system must be low-cost, easy to setup and use, low-latency and most importantly accurate enough. It should be effortless for the teleoperator to collect high quality data of robots performing complex tasks of interest to train robot policies.

To address this problem, VR headsets have become increasingly prevalent due to their easy-to-use internal body tracking systems [140, 282]. However we find that wrist tracking is often jittery and the finger tracking is inaccurate. To mitigate this, SteamVR [38] uses LiDAR which provides less noisy estimates, but requires external tracking devices which doesn't allow for data collection for mobile robot setups. For even higher fidelity readings, there is work that uses motion capture and reflective marker-based approaches such as Vicon or Optitrack[382] but they are extremely expensive and difficult to setup. An aspect of motion capture technology that has been used in robot learning in recent years are wearable gloves [259, 339, 344, 385] for hand tracking which records human fingertip position using EMF sensors. We use the accuracte Manus Meta glove as part of our system. [24]

For arm tracking, researchers in the robotics community have been recently using joint-level teleoperation for 2-finger grippers [393, 413]. They find that a low-cost 3D printed scaled teacher arm model that has the same kinematic link structure of a large robot arm can be used for accurate and effective teleoperation. These methods only provide one DOF of finger tracking instead of the twenty two plus DOF of the human hand. Our key insight is to develop a system that combines this joint-based arm tracking along with a Mocap fingertip glove to achieve accurate low-cost teleoperation of the arm and hand.

Our contribution is BiDex, a system for dexterous low-cost teleoperation system for bimanual hands and arms in-the-wild in any environment. To operate, the user wears two motion capture gloves and moves naturally to complete everyday dexterous tasks. The gloves captures accurate finger tracking to map motions naturally to the robot hands. The GELLO [393] inspired arm tracking accurately tracks the human wrist position and joint angles for the robot arm. The data collected by the system can be used to train effective policies using imitation learning, after which the bimanual robot system can perform the task autonomously. BiDex costs around $6k for a pair of Manus gloves and few hundred dollars for the arm teleoperation which works for many existing robot arms. Even including the robot arms ($8k xArms x2) and robot hands used in our demonstration ($3k LEAP Hand V2 x2), the total cost is under $30k. We compare the accuracy and speed of data collection to other commonly used systems: the VR headset and SteamVR. We release videos of our results and instructions to recreate the setup on our website at https://bidex-teleop.github.io

9.3 Related Work

Robot Arm Teleop Many common approaches to teleoperation include using joysticks, space mouses [236], vision-based methods, [313, 353], and VR headsets [61, 140, 282] which control arms with inverse kinematics. Joint based teleoperated control has been used in areas such as kinesthetic teaching, Brantner and Khatib [86], ABB YuMi [226] and Da Vinci Machines [153], and recently [155, 414] introduced a low-cost version of this with mirroring Trossen Robot arms. GELLO [393] uses light and inexpensive teacher arms that are 3D printed to control full size robot arms, a system we use in our BiDex due to its lost-cost, accurate, portable design.

Robot Hand Teleop The high dimensionality makes tracking human hands particularly difficult. To control robot hands, many vision-based techniques such as [169, 306, 353] do not require specialized equipment but are not that accurate. Shadow Hand developed a professional system that uses SteamVR and two gloves to control two Shadow Hands [330]. Recently, bimanual robot hands and tracking have become accessible for academic labs. Dexcap uses LEAP Hand [339] and tracks the human with gloves and a SLAM-based robot camera. [385] Hato uses a VR headset and controller to control two 6 DOF Psyonic Hands [33, 233]. A key question in controlling robot hands is how to map the human hand configuration to robot hand joints. These papers introduce inverse kinematics based methods that optimize pinch grasps between the human and robot hands [169, 306, 353, 385].

Motion Capture Motion capture and graphics contributions often are useful in the robotic teleoperation domain. Outside-in mocap approaches use external sensing technology to track the human body or other objects in the scene. SteamVR uses external lasers and worn wireless laser receivers. [38] Vicon-based systems use reflective balls and external cameras to track. Inside-out approaches such as XSens [269] or Rokoko suit rely on IMUs on the body but these often drift over time and require recalibration [233, 316]. For hand data, many vision-based approaches such as Frankmocap [321] return MANO [318] parameters which can be converted to robot hand joint angles.

Learning from Expert Demonstrations Recently the robot learning community has seen notable success in learning from demonstrations driven by the development of imitation learning algorithms [113, 258]. Complementing these advances, significant efforts have been made to scale up robotic datasets to facilitate more capable robotic systems [128, 201, 280, 384]. Despite these efforts, acquiring robotics data remains an

expensive and challenging endeavor. To address these issues, developments in low-cost hardware have been instrumental in democratizing access to robotic technology, enabling more widespread research and application [115, 155, 357, 414]. However, these systems are primarily focused on simple gripper functionalities; and the challenge of achieving more intricate dexterity and intuitive control in robotic systems motivates our bimanual dexterous teleop system.

9.4 Bimanual Robot Hand and Arm System

We present BiDex, a system that allows any operator to effortlessly teleoperate a bimanual robot hand and arm setup. BiDex is designed to be exceptionally precise, affordable, low-latency, and portable, enabling control of any human-like pair of dexterous hands, even those with over 20 degrees of freedom. It achieves accurate tracking of the human hand using a Manus VR glove-based system [24] and human arm tracking through a GELLO-inspired system [393]. We outline the process for sending commands and collecting data with bimanual hands in Alg.3. Importantly, our solution functions seamlessly in both tabletop and mobile environments, as it requires no external tracking devices and is highly portable. In Section 9.6, we demonstrate that our system is highly intuitive, precise, and cost-effective compared to widely used methods today, such as VR headsets and SteamVR, across two pairs of open-source robot hands, LEAP Hand [339] and LEAP Hand V2 [336].

9.4.1 Multi-fingered Hand Tracking

A hand tracking system must deliver accurate, low-latency joint information for the human hand, which has over 20 degrees of freedom. Many current vision-based tracking solutions, such as those using FrankMocap [288, 321] or VR headsets, often face significant inaccuracies due to occlusions and varying lighting conditions, as discussed in Section 9.6. In contrast, recent motion capture gloves that utilize EMF sensors provide significantly more accurate tracking without being overly expensive. They avoid the occlusion issues common in vision-based methods and offer detailed data on the skeletal joint structure of the human hand. Additionally, these gloves can be worn comfortably without hindering movement. In BiDex, we opt for the Manus Glove [24], which has demonstrated reliable tracking performance without overheating or suffering from calibration problems as seen with alternatives like the Rokoko gloves [233, 316]. However, mapping this data from the

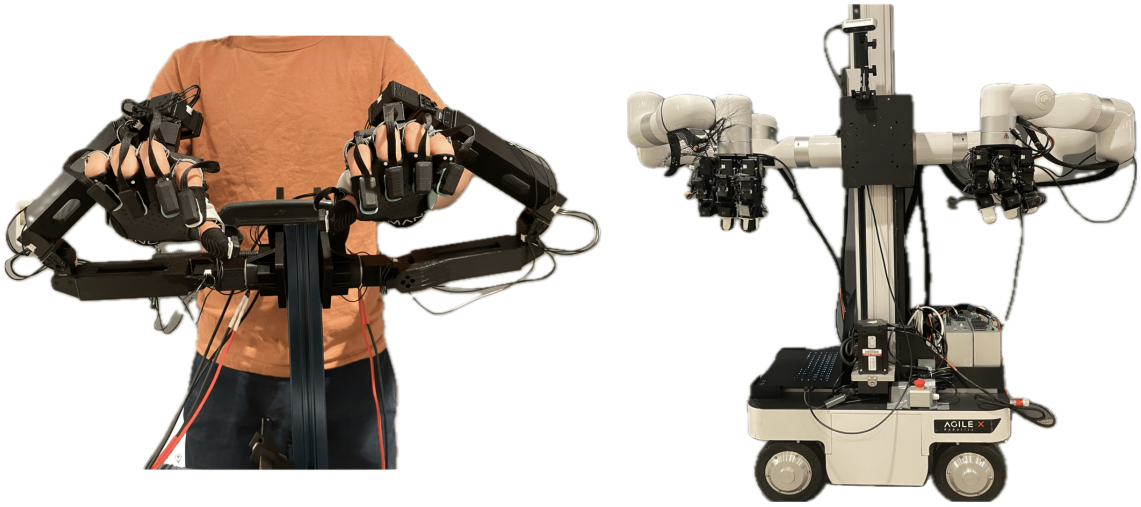

Figure 9.2: **Mobile bimanual teleoperation system** Left: An operator strapped into BiDex. Right: Our bimanual robot setup including two xArm robot arms, two LEAP Hands [339] and three cameras on an AgileX base.

human hand to a robot hand remains challenging due to differences in their morphological structures. How can we translate commands from the operator's fingers to a robot hand that may have a different kinematic configuration?

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

Previous work [169, 353] has addressed this challenge by ensuring that pinch grasps are consistent between human and robot hands. Wang et al. [385] demonstrated that effective mapping can be achieved by optimizing the joint positions of the fingertip and penultimate joint (DIP) on each finger in relation to the wrist, ensuring similarity between the human and robot hands through an SDLS IK solver [90]. We employ the Manus gloves [24] using a similar inverse kinematics approach, which allows for precise pinch grasps and proper

thumb positioning.

9.4.2 Arm Tracking

Our system must accurately track the human wrist pose to control two robot arms. Traditionally, various methods have been developed by both the motion capture and robotics communities. However, many of these approaches rely on calibrated external tracking devices which either are costly or have high latency. These external tracking devices are also non-portable, making it hard to scale to mobile systems. Instead, we leverage key insights from Wu et al. [393], Zhao et al. [413] which both use lighter teacher arms attached to the human arm to control a robot arm and hand system. Specifically we follow the GELLO system from Wu et al. [393] to teleoperate a full size robot arms. A key question is how to mount this arm-tracking system on a human wrist and hand. If the robot hand is mounted on the arm in a human-like way, then the glove needs to be mounted in a human-like way on the teacher arm to match. However, this orientation means that the human arm will be parallel with the teacher arm and constantly collide with it. This is jarring for the operator and uncomfortable.

In BiDex, we mount the robot hands underneath the arms in the same orientation as if it were a gripper as in Figure 9.2. When mirroring this in the teacher arm, the human arm and teacher arm output are perpendicular to each other and do not collide which is more comfortable. Because of the weight of the motion capture glove, we must adjust the teacher arm to be more robust. This includes adding a strong bearing to the base joint of the teacher arm and adding rubber bands to bias additional joints back to the center of the joint range.

9.4.3 Robot Configurations

Tabletop Manipulation For our tabletop setup, the robotic arms are positioned to face each other similar to Zhao et al. [413] while the GELLO teaching arms mirror this configuration. Compared to a side-by-side configuration like in Wang et al. [385], this setup has three main benefits: 1) the human operators avoid collisions with the teacher arms; 2) the setup allows better visibility of the workspace, which will be otherwise occluded by a side-by-side robot arm configuration; 3) and finally, the robot arms have a wider shared workspace.

Mobile Manipulation BiDex operates without the need for external tracking systems and is lightweight, making it well-suited for mobile settings. The teacher arms are mounted on a compact mobile cart. For the mobile robot, we attach two robot arms to an articulated torso capable of moving upward to reach high objects and downward toward the ground,

Figure 9.3: **All Tasks**: Teleoperation of the mobile robot systems with BiDex. **Top**: Picking up trash from a table and discarding it into a bin. **Bottom**: Grasping a chair and moving it to align with a table.

similar to the PR2 [79], as shown in Figure 9.2. The robotic assembly, which includes the arms and torso, is mounted on an AgileX Ranger Mini, allowing for movement in any SE(2) direction [51, 395]. A secondary operator uses a joystick to control the mobile base and manage task resets.

Algorithm 3 Teleoperation Data System

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

Initialize Data buffer $\mathcal{D}$
for trajectory 1:N **do**
  Initialize trajectory $\mathcal{T} = \{\}$
  while task not completed **do**
    Read camera images $\{I_c\}_t$
    Read arm joints Al_t, Ar_t
    Read robot hand joints Hl_t, Hr_t
    Observation $o_t = \{Al_t, Ar_t, Hl_t, Hr_t, \{I_c\}_t\}$

    Read kinematic arm model joints Kl_t, Kr_t
    Read glove fingertip positions Gl_t, Gr_t
    Finger joints $ql_t = Q_l(Gl_t), qr_t = Q_r(Gr_t)$
    Action $a_t = \{Kl_t, Kr_t, ql_t, qr_t\}$
    Add $(o_t, a_t) \mapsto \mathcal{T}$

    Set joints of Al using Kl_t and Ar using Kr_t
    Set joints of Hl using ql_t and Hr using qr_t
  end while
  Add $\mathcal{T} \mapsto \mathcal{D}$
end for
return Data buffer $\mathcal{D}$
=0

9.5 Experiment Setup

9.5.1 Baseline Teleoperation Approaches

Vision based VR Headset In recent years, the accessibility of low-cost VR headsets using multi-camera hand tracking has made them popular for teleoperation such as in [140, 282] As a baseline we use the Apple Vision Pro which returns both finger data similar to MANO parameters [288] and wrist coordinate frame data. The finger data is used in the same way as with BiDex through inverse kinematics-based retargeting and commanded onto the robot hands. The wrist data is reoriented, passed through inverse kinematics and the final joint configuration is commanded to the arms.

SteamVR Tracking SteamVR, commonly used in the video gaming community has also seen recent interest in the robotics community from industry [330] and academia alike. [50, 259] It uses active powered laser lighthouses that must be carefully placed around the perimeter of the workspace. Wearable pucks with IMUs and laser receivers are worn on the body of the operator. In our experiment the operator wears one tracker on each wrist and one tracker on their belly. The wrist position is determined relative to the belly pose, mapped to the robot arm, and the joint angles are computed using inverse kinematics. The hand tracking gloves are the same as in BiDex.

9.5.2 Choice of Dexterous End-Effectors

Leap Hand LEAP Hand, introduced by Shaw et al. [339] is a low-cost, easy-to-assemble robot hand with 16 DOF and 4 fingers. LEAP Hand introduces a novel joint configuration that optimizes for dexterity as well as human-like grasping. We use this hand for many experiments in the paper as it is a readily available dexterous hand available for comparison studies.

LEAP Hand V2 We would like a hand that is smaller and more compliant than LEAP Hand. LEAP Hand V2 is crafted to mimic the suppleness and strength of the human hand with fingers that have a 3D-printed flexible outer skin paired with a sturdy inner framework resembling bones. These fingers do not break but instead bend and flex upon impact. We also introduce an active articulated palm which integrates two motorized joints, one spanning the fingers and another for the thumb, enabling natural tight grasping. LEAP

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

Table 9.1: **Tabletop Teleoperation**: We compare BiDex on the handover, cup stacking, and bottle pouring tasks to two baseline methods, SteamVR and Vision Pro. BiDex enables more reliable and faster data collection, especially for harder tasks like bottle pouring.

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

Table 9.2: **Mobile Teleoperation**: Completion rate and time taken averaged across 20 trials using a mobile bimanual system with LEAP Hand [339], for different tasks. BiDex is versatile and compact enough to be adopted to successfully collect data for mobile tasks.

Hand V2 contains 21 degrees of freedom and is sized to resemble a human hand, easy to assemble and is economical. Because of the human-like size and kinematics, it is easy to retarget to and can complete many more dexterous tasks successfully.

9.5.3 Task Descriptions

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

Mobile In ***Transport Box***, the agent moves a box from one table to another table using two hands. In ***Mobile Chair Push***, the agent needs to grasp a chair, and then align it with a table. In ***Clear Trash***, the robot clears trash from the table into a dustbin. We visualize the chair push and clear trash tasks in Figure 9.3.

9.6 Results

We investigate BiDex teleoperating in both the tabletop scenario and in the mobile in-the-wild scenario against a few baselines. For these comparisons, we use LEAP Hand by Shaw et al. [339] because it is an open source easy to assemble robot hand that is readily attainable

by any robotics lab. We also use BiDex with the more recent LEAP Hand v2, which is made of a combination of rigid and soft material and capable of performing more complex tasks.

9.6.1 Bimanual Dexterous Teleoperation Results

BiDex provides more stable arm tracking. The teacher arm system makes BiDex highly reliable, with minimal jitter, low latency and high uptime, making it ideal for arm tracking. The teacher arm is lightweight and doesn't impede the user more than the gloves' weight. The kinematic feedback from arm resistance is subtle but helps operators navigate around arm singularities intuitively. As shown in Tables 9.1 and 9.2, BiDex achieves a higher completion rate in less time for teleoperators.

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

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

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

In contrast, the Vision Pro can struggle with hand size variability under different lighting conditions, making the retargeting process to robot hands more challenging. Additionally, finger abduction and adduction estimates can be impacted

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

Table 9.3: **Imitation learning**: We train ACT from [413] using data collected by BiDex and find that our system can perform well even in this 44 dimension action space. This demonstrates that our robot data is high quality for training robot policies.

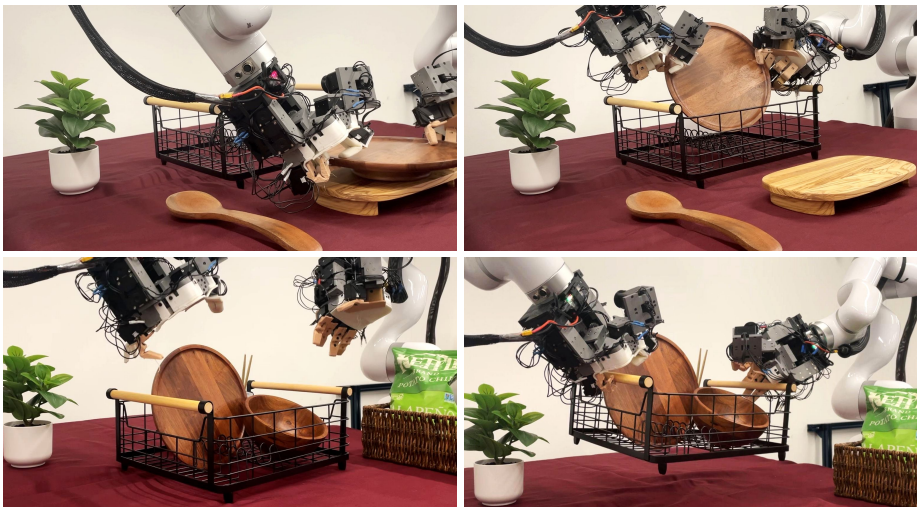

Figure 9.4: **Clearing the Dishes**: In this task, we use BiDex to perform a long horizon task to place bowls and spoons into a drying rack and lift the drying rack away from the table.

by occlusions, which complicates the performance of intricate tasks. The latency is noticeable and can pose a significant issue for teleoperation.

9.6.2 Training Dexterous Visuomotor Policies with BiDex

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

Specifically, we train an action chunking transformer from [413] with a horizon length of 16 at 30hz using pretrained weights from [132] on around 50 demonstrations. The state space is the current joint angles of the robot hand and the images from the camera. The action space in the case of LEAP Hand is 16 dimensions for each hand and 6 dimensions for each arm for a total of 44 dimensions. During rollouts, the behavior of the policies are very smooth, exhibiting the high quality of the teleop data. In tasks such as the YCB Pringles can [92] handover, we even see good generalization of the policy to different initial locations of the can.

9.6.3 Extreme Dexterity using LEAP Hand V2

To push BiDex to its dexterous limits, we use LEAP Hand V2: an extremely dexterous 21 DOF hybrid low-cost hand which is explained in Section 9.5.2.

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

Table 9.4: **Imitation learning LEAP v2**: We also train ACT using LEAP Hand v2 and show task completion on more dexterous tasks.

To do this, we show a variety of very challenging tasks as in Figure 10.1 and Figure 9.4. These tasks include pouring, scooping, hammering, chopstick picking, hanger picking, picking up basket, drilling, plate pickup and pot picking. In these experiments we find that BiDex scales well to this high DOF hand and it feels very natural to control this soft-rigid robot hand. We provide Table 9.4 and video results on our website at https://bidex-teleop.github.io

9.7 Discussion and Limitations

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

Chapter 10

DexWild: Dexterous Human Interactions for In-the-Wild Robot Policies

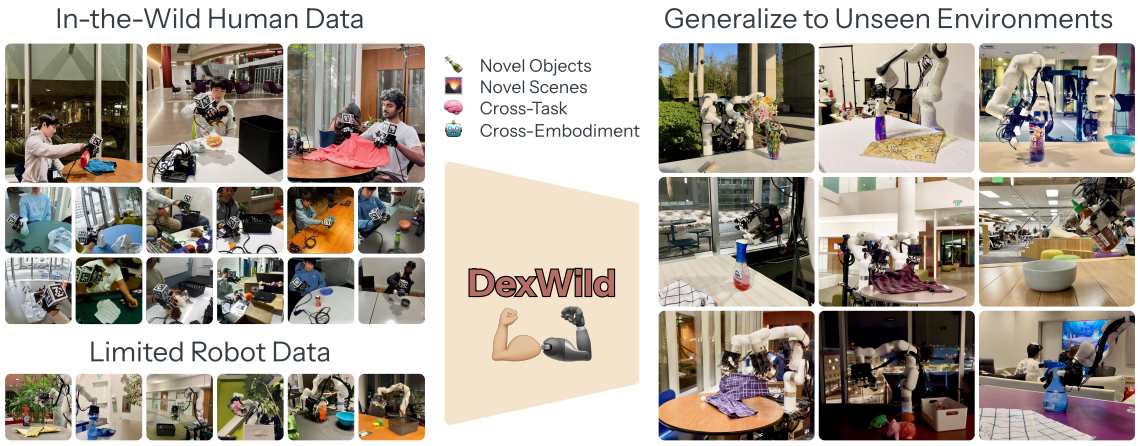

Figure 10.1: **Bimanual Dexterity**: BiDex can effortlessly teleoperate various complex tasks including pouring, scooping, hammering, chopstick picking, hanger picking, picking up basket, drilling, plate pickup and pot picking to train high-quality behavior cloning policies with over 50 degrees of freedom. Our teleoperation system and two LEAP hands [339] costs around $12k in total and is readily reproducible by academic labs.

10.1 Abstract

Large-scale, diverse robot datasets have emerged as a promising path toward enabling dexterous manipulation policies to generalize to novel environments, but acquiring such

datasets presents many challenges. While teleoperation provides high-fidelity datasets, its high cost limits its scalability. Instead, what if people could use their own hands, just as they do in everyday life, to collect data? In DexWild, a diverse team of data collectors uses their hands to collect hours of interactions across a multitude of environments and objects. To record this data, we create DexWild-System, a low-cost, mobile, and easy-to-use device. The DexWild learning framework co-trains on both human and robot demonstrations, leading to improved performance compared to training on each dataset individually. This combination results in robust robot policies capable of generalizing to novel environments, tasks, and embodiments with minimal additional robot-specific data. Experimental results demonstrate that DexWild significantly improves performance, achieving a 68.5% success rate in unseen environments—nearly four times higher than policies trained with robot data only—and offering 5.8× better cross-embodiment generalization. Video results, codebases, and instructions at https://dexwild.github.io

10.2 Introduction

Roboticists have long dreamed of creating robots that can perform tasks with the same dexterity and adaptability as humans. We would like robots to deftly generalize to many different objects, environments, and embodiments-yet this vision of truly versatile robot behaviors remains a formidable challenge. While there have been many breakthroughs in large language models (LLMs) [87, 374, 379] and vision language models (VLMs) [235, 359], the key to their success lies in harnessing vast datasets. In contrast, robotics faces a critical hurdle: large-scale, diverse robot datasets needed to train foundation models do not yet exist.

In recent years, a key approach to collecting robot datasets has been through teleoperation, which provides high-precision, high-quality action data that a policy can directly train on. [117, 201, 384]. However, acquiring this data requires highly-trained human operators working with specialized robot setups. Gathering data in diverse environments presents additional challenges such as physically relocating the robot to each new location. This data collection process is both labor-intensive and expensive, making it difficult to scale to the volume of data needed for dexterous generalization in unseen environments.

Another approach to scaling robot datasets is to leverage internet-scale video data from platforms like YouTube, which provide vast and diverse visual grounding in real-world

environments [123, 160]. However, utilizing this data effectively presents significant challenges. First, publicly available videos often lack the fine-grained accuracy needed to capture detailed hand states because vision-based body detection modules are noisy and unreliable. Additionally, these videos are not inherently structured with categorized episodes for task-specific learning, further complicating their direct application in robotics. [68, 178, 343]. While some data collection efforts exist with more accurate and structured data, [69, 421], they do not have enough environment diversity. We seek to collect data with **tracking accuracy and environment diversity** to enable generalizable dexterous behavior.

To overcome these barriers, some have explored collecting accurate in-the-wild human demonstrations by equipping users with a wearable gripper that directly maps their hand movements to robot actions [115]. However, this approach is cumbersome, ill-suited for natural, everyday interactions, and constrains the collected data to a specific embodiment. Other works [385] propose using dexterous hands and gloves, but they do not scale to in-the-wild environments.

In this paper, we present DexWild, a system that enables effective learning of robust dexterous manipulation policies through co-training on human and robot demonstrations. Our key contributions include:

1. **Scalable Data Collection System**: A novel human-embodiment DexWild-System that enables untrained operators to quickly collect 9,290 demonstrations across 93 diverse environments, achieving 4.6× speedup over conventional robot-based methods
2. **Efficient Co-training Framework**: An approach that optimally combines human and robot demonstrations, significantly improving policy generalization to achieve 68.5% success rate in novel environments, nearly four times higher than robot-only policies.
3. **Strong Cross Embodiment and Cross Task Performance**: Our data collection system combined with our co-training framework achieves of 5.8× improvement in cross-embodiment transfer over baselines and effective skill transfer across tasks.

10.3 Related Works

10.3.1 Generalization for Imitation Learning

Learning generalizable policies for robot manipulation has seen rapid progress, driven largely by advances in visual representation learning and imitation learning from large-scale datasets. On the visual side, embodied representation learning has benefited from egocentric datasets

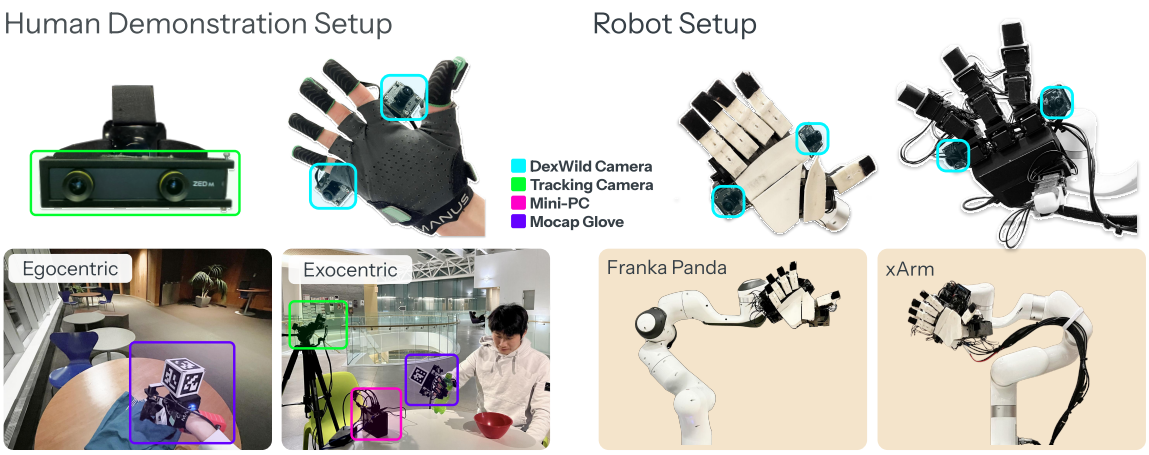

Figure 10.2: **Left:** DexWild efficiently capture high-fidelity data using an individual's own hands across various environments. **Right:** Robot hands are equipped with cameras aligned with the human cameras. We test DexWild on two distinct robot hands and robot arms.

such as Ego4D [160] and EPIC-KITCHENS [123], with recent methods [131, 276, 342, 358] leveraging these datasets to train scalable visual encoders. However, these approaches still require substantial downstream robot demonstrations to train control policies.

In parallel, robot-only demonstration datasets have grown significantly in scale and diversity [117, 201, 384], fueling research in behavior cloning and enabling generalist policy architectures [117, 202, 369]. While these policies show impressive performance across many tasks, they often struggle to generalize to unseen object categories, scene layouts, or environmental conditions [258]. This lack of robustness remains a key limitation of current systems.

10.3.2 Data Generation for Robot Manipulation

Overcoming the robot data bottleneck has become a central challenge in robot learning.

One approach leverages internet videos to extract action information. Several works, such as VideoDex [343] and HOP [350], utilize large scale human videos to learn an action prior through retargeting, which they use to bootstrap policy training. Others, such as LAPA [399], use unlabelled videos to generate latent action representations that can be used for downstream tasks. While these video-based schemes enjoy vast visual diversity, they typically fall short at capturing the precise, low-level motor commands needed for real-world manipulation.

Simulation enables rapid generation of action data at scale. However, creating diverse,

realistic environments for many tasks and addressing the sim-to-real gap is challenging. Recent successes in transferring manipulation policies from simulation [351] have been confined to tabletop settings and lack the generalization needed for deployment in diverse environments.

Direct teleoperation on physical robots yields the highest fidelity, but scales poorly. Recent works have shown impressive dexterity and efficient learning in fixed scenarios [187, 346, 393, 414], yet collecting enough demonstrations to generalize across diverse scenes quickly becomes prohibitively expensive.

Recently, there has been a growing body of work that utilizes purpose-collected high quality human embodiment data without the tedious teleoperation. We discuss these approaches in the next section.

10.3.3 Human Action Tracking Systems

In order to acquire high-quality data from human motions, accurate hand and wrist tracking is of paramount importance. To bypass the complexities of hand pose estimation, several works equip users with handheld robot grippers [115, 144, 357]. While this approach simplifies retargeting, it constrains users to the specific morphology of the robot gripper, limiting the diversity of captured behavior. Moreover, many of these systems rely on SLAM-based wrist tracking, which can fail in feature-sparse environments or when occlusions occur [115, 232]—such as during drawer opening or tool use.

Other approaches aim to estimate both hand and wrist poses directly from visual input [112, 199, 286, 288, 307, 321, 354]. These methods are easy to deploy and require no instrumentation, but their performance degrades significantly under occlusion—an unavoidable situation in manipulation. Alternative strategies for wrist tracking, such as IMU-based [120, 370] and outside-in optical systems [292], come with their own limitations: IMUs are lightweight and portable but prone to drift, while optical systems are accurate yet require laborious calibration and controlled environments. DexWild leverages calibration-free Aruco tracking—significantly improving reliability and minimizing setup time as it requires a single monocular camera.

While vision-based methods often attempt to track both the wrist and fingers simultaneously, many recent systems decouple the two to improve accuracy. Kinematic exoskeleton gloves can provide high-fidelity joint measurements and even haptic feedback [409], but

are bulky and uncomfortable for long-term use. Instead, DexWild, along with prior works [346, 385], adopts a lightweight glove-based solution that uses electromagnetic field (EMF) sensing to estimate fingertip positions. This allows for accurate, real-time hand tracking that is robust to occlusions and readily retargetable to a wide range of robot hands.

10.4 DexWild

Many believe that leveraging large, high-quality datasets is the key for creating dexterous robot policies that generalize [117, 131, 343, 369]. We introduce DexWild-System, a user-friendly, high-fidelity platform for efficiently gathering natural human hand demonstrations across diverse real-world settings. Compared to traditional teleoperation-based approaches, DexWild-System enables 4.6× faster data acquisition at scale.

Building on this system, we propose DexWild, an imitation learning framework that co-trains on large-scale DexWild-System human demonstrations alongside a small number of robot demonstrations. This approach combines the diversity and richness of human interactions with the grounding of the robot embodiment, enabling policies to robustly generalize across new objects, environments, and embodiments. Figure 10.1 displays our high level approach.

10.4.1 Data Collection System

A scalable data collection system for dexterous robot learning must enable natural, efficient, and high-fidelity collection across diverse environments. To this end, we design DexWild-System: a portable, user-friendly system that captures human dexterous behavior with minimal setup and training. While previous in-the-wild data collection approaches have typically relied on sensorized grippers, we aimed to create a more intuitive hardware interface that mirrors how humans naturally interact with the world. From delicate fine-motor actions to powerful grasps, humans possess dexterity across a wide range of manipulation tasks. By learning from this intrinsic capability, DexWild-System captures rich, diverse data applicable to a broad range of robot embodiments.

DexWild-System is designed around three core objectives:

- **Portability:** Allow rapid, large-scale data collection across diverse environments without requiring complex calibration procedures.

- **High Fidelity:** Accurately capture fine-grained hand and environment interactions essential for training precise dexterous policies.
- **Embodiment-Agnostic:** Enable seamless retargeting from human demonstrations to a wide variety of robot hands.

Portability:

To collect data in diverse real-world settings, a system must be portable, robust, and usable by anyone. We design DexWild-System with these goals in mind: it is lightweight, easy to carry, and can be set up in just a few minutes—enabling scalable data collection across many locations.

As shown in Figure 10.2, DexWild-System consists of only three components: a single tracking camera for wrist pose estimation, a battery-powered mini-PC for onboard data capture, and a custom sensor pod comprising a motion-capture glove and synchronized palm-mounted cameras.

Unlike traditional motion capture systems [38, 103, 146, 421] that often rely on complex outside-in tracking setups that require calibration, DexWild-System is truly calibration free, making it versatile for any scenario and foolproof for untrained operators.

This is achieved by adopting a relative state-action representation, where each state and action is captured as the relative difference from the previous time step's pose. This eliminates any need for a global coordinate frame, allowing the tracking camera to be freely placed—either egocentrically or exocentrically. Additionally, the palm cameras are rigidly mounted in fixed positions across both human and robot embodiments. This ensures visual observations are aligned across domains, eliminating the need for further calibration at deployment. The external tracking camera, when carefully positioned, can also capture supplementary environmental context useful for learning robust policies.

High Fidelity:

To learn dexterous behaviors, fine-grained, nuanced motions must be captured in the training dataset. Although DexWild-System consists of only a few portable components, we make no compromises on data fidelity. Our system is designed to accurately capture both hand and wrist actions, paired with high-quality visual observations.

For wrist and hand tracking, vision-only methods are easy to setup. However, what they gain in portability, they often lose in accuracy and robustness—yielding noisy pose estimates that degrade policy learning [115, 154, 307, 346].

For hand pose estimation, we use motion capture gloves, which offer high accuracy, low latency, and robustness against occlusions [346]. For wrist tracking, we mount ArUco markers on the glove and track them using an external camera. This avoids the fragility of SLAM-based wrist tracking, which often fails in feature-sparse environments or during occlusion-heavy tasks (e.g., drawer opening).

Unlike many datasets that rely on egocentric or distant external cameras, we place two global-shutter cameras directly on the palm. As illustrated in Figure 10.2, these stereo cameras capture detailed, localized interaction views with minimal motion blur and a wide field of view. This wide field of view enables policies to operate using only the onboard palm cameras, without any reliance on static viewpoints.

Embodiment-Agnostic:

To ensure the longevity and versatility of DexWild data, we aim for it to remain useful across different robot embodiments—even as hardware platforms evolve. Achieving this goal requires careful alignment of both the observation space and the action space between humans and robots.

We begin by standardizing the observation space. Although our palm-mounted cameras have a wide field of view, we intentionally position them to focus primarily on the environment, minimizing the visibility of the hand itself. Importantly, the camera placement is mirrored between the human and robot hands. As shown in Figure 10.3, this design yields visually consistent observations across embodiments, allowing the policy to learn a shared visual representation that generalizes across both human and robot domains.

For action space alignment, we build on insights from prior work [169, 352], optimizing robot hand kinematics to match the fingertip positions observed in human demonstrations. We note that this method is general and can work for any robot hand embodiment. It operates with fixed hyperparameters across users and is robust to variations in hand size—eliminating the need for user-specific tuning.

Collecting data using natural human hands offers benefits beyond ease of use. The diversity in hand morphology across human demonstrators introduces useful variation, which we hypothesize helps policies learn more generalizable grasping strategies—particularly important given the inherent mismatch between human and robot hand kinematics.

In summary, DexWild is a portable, high quality, human-centric system that can be worn by any operator to collect human data in real-world environments. Next, we explain how we use the data collected by DexWild to enable dexterous policies to generalize to

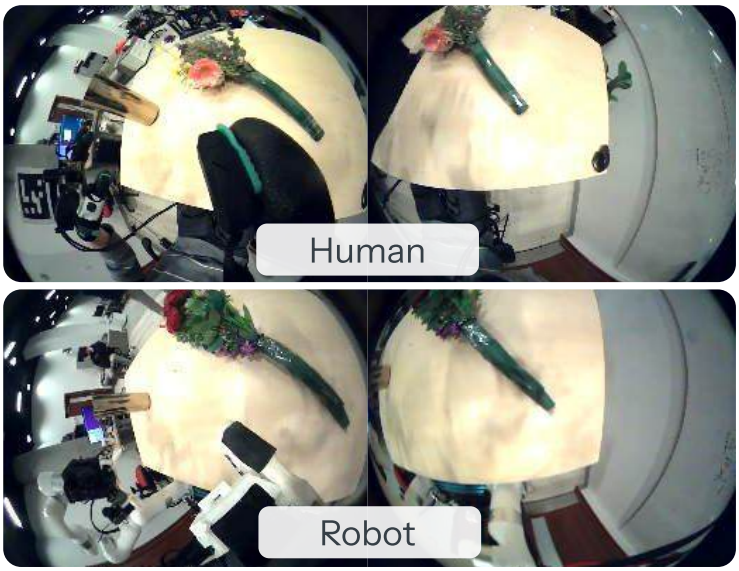

Figure 10.3: DexWild aligns the visual observations between humans and robots to bridge the embodiment gap. This incentivizes the model to learn a task-centric rather than embodiment-centric representation.

in-the-wild scenarios.

10.4.2 Training Data Modalities and Preprocessing

Generalization in dexterous manipulation demands both scale and embodiment grounding. With this goal, DexWild collects two complementary datasets: a large-scale human demonstration dataset D_H using DexWild-System, and a smaller teleoperated robot dataset D_R. Human data offers broad task diversity and ease of collection in real-world settings, but lacks embodiment alignment. Robot data, while limited in scale, provides crucial grounding in the robot's action and observation spaces. To harness the strengths of both, we co-train policies using a fixed ratio of human and robot data within a batch, (w_h, w_r)—balancing diversity with embodiment grounding to enable robust generalization during deployment.

At each training iteration, we sample a batch consisting of transitions x_h and x_r from D_H and D_R, respectively, according to the co-training weights. Each transition x_i at timestep i contains:

- **Observation** o_i: An observation at a given timestep consists of two synchronized palm camera images I_{pinky} and I_{thumb} captured at the current timestep, as well as a sequence of historical states, sampled at a step size up a given horizon H, comprising of $\{\Delta p_i, \Delta p_{i-\text{step}}, ..., \Delta p_{i-H}\}$. Each Δp consists of relative historical end-effector positions.

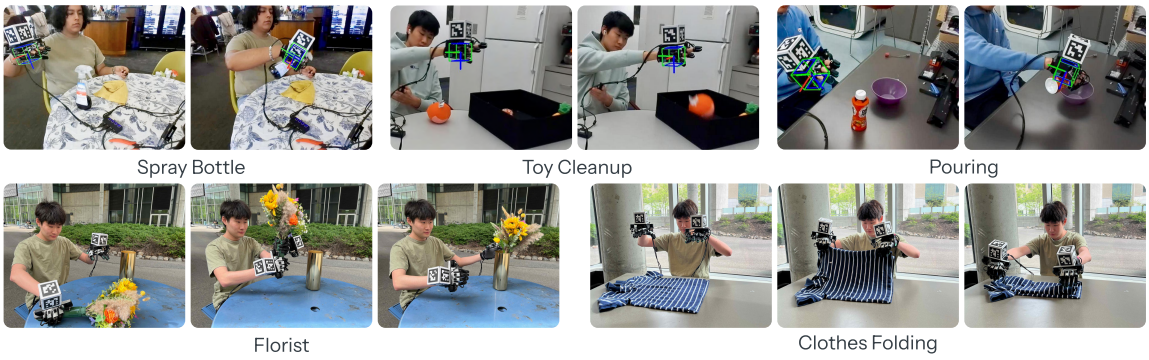

Figure 10.4: Using DexWild-System, humans can effortlessly collect accurate data with their own hands across a wide range of environments. This data is directly used to train any robot hand to perform dexterous manipulation in a human-like way in any environment. We validate this approach on five representative tasks. Please see videos of these tasks on our website at https://dexwild.github.io

- **Action** $a_{i:i+n-1}$: An action chunk of size n that includes actions $\{a_i, a_{i+1}, \ldots, a_{i+n-1}\}$, where a_i is the action at the current timestep. Specifically, a_i is a 26-dimensional vector consisting of:
  - a_{arm}: A 9-dimensional vector describing relative end-effector position (3D) and orientation (6D).
  - a_{hand}: A 17-dimensional vector describing the finger joint position targets of the robot hand.

  For bimanual tasks, the observation and action spaces are duplicated, and the inter-hand pose is appended to the observation to facilitate coordination.

While our retargeting procedure brings human and robot trajectories into a shared action space, a few additional steps are necessary to make the human and robot datasets compatible for joint training:

- **Action Normalization**: The actions of human and robot data are normalized separately to account for inherent distribution mismatches.
- **Demo Filtering**: Since human demonstrations are collected by untrained operators in uncontrolled environments, we apply a heuristic-based filtering pipeline to automatically detect and remove low-quality or invalid trajectories. This filtering step significantly improves dataset quality without manual labeling.

Figure 10.5: We collect data using a diverse set of objects across categories. *Spray Bottle Task* – 25 Train, 11 Test; *Toy Cleanup Task* – 64 Train, 9 Test; *Pour Task* – 35 Train, 5 Test; *Florist Task* - 6 Train, 2 Test; *Clothes Folding Task* - 17 Train, 6 Test.

10.4.3 Policy Training

Through the careful design of our hardware, observation, and action interfaces, we are able to train dexterous robot policies using a simple behavior cloning (BC) objective [295, 322, 323]. To effectively learn from our multimodal, diverse data, our training pipeline leverages large-scale pre-trained visual encoders and shows strong performance across different policy architectures.

Visual Encoder: Training on DexWild data exposes our policy to significant visual diversity—across scenes, objects, and lighting—requiring an encoder that generalizes well to such variability. To address this, we adopt a pre-trained Vision Transformer (ViT) backbone, which has shown superior performance over ResNet-based encoders on in-the-wild manipulation tasks [163, 232]. Pre-trained ViTs, especially those trained on large internet-scale datasets, are particularly effective at extracting rich, transferable features [131, 276, 310, 358], making them well-suited for our setting.

Policy Class: While several imitation learning architectures have been proposed recently [114, 414], we adopt a diffusion-based policy. Diffusion models are particularly well-suited for dexterous manipulation, as they can capture multi-modal action distributions more effectively than alternatives such as Gaussian Mixture Models (GMMs) or transformers. This capability becomes increasingly important in DexWild, where demonstrations are collected from multiple humans with diverse strategies, resulting in inherently multi-modal behaviors. As the dataset scales, modeling this variability becomes critical for robust policy learning. Specifically, DexWild uses a diffusion U-Net model [114] to generate action chunks.

Concretely, the training procedure is outlined in Algorithm 4.

Algorithm 4 DexWild Imitation Learning Procedure

Require: Human dataset $\mathcal{D}_H$, Robot dataset $\mathcal{D}_R$, Co-training weights $\{\omega_h, \omega_r\}$

Initialize policy π_θ with ViT encoder ϕ_{vit}
while not converged **do**
  Sample a batch of transitions $\{x_h\}, \{x_r\}$ from $\mathcal{D}_H, \mathcal{D}_R$ using weights $\{\omega_h, \omega_r\}$
  for each transition x_i in the batch **do**
    Extract observation o_i
    Encode images: $Z_i = \phi_{\text{vit}}(o_i)$
    Extract ground truth action chunk $a_{i:i+n-1} = \{a_i, \ldots, a_{i+n-1}\}$
    Sample noise scale $t \sim \mathcal{U}(1, T)$
    Add noise $\epsilon_t \sim \mathcal{N}(0, \sigma_t)$ to $a_{i:i+n-1}$
    Predict noise $\hat{\epsilon}_\theta = \pi_\theta(Z_i, a_{i:i+n-1} + \epsilon_t, t)$
    Compute diffusion loss $\mathcal{L}_\theta = \|\epsilon_t - \hat{\epsilon}_\theta\|_2^2$
  end for
  Update policy parameters θ
end while=0

An important finding in our training framework is that tuning the human-to-robot data weighting significantly affects real-world performance. We discuss these effects in Section 10.6.1.

10.5 Experiments

Our experimental evaluation encompasses extensive real-world deployment across diverse environments and robots, utilizing both human demonstrations and robot teleoperation data. Below, we outline our data collection process, experimental setup, and evaluation tasks.

10.5.1 Scaling up Data Collection

Our hardware system was deployed to 10 untrained users to collect data across a wide range of real-world environments. These settings included indoor and outdoor locations, day and night conditions, crowded cafeterias and quiet study areas, with varied tables, objects, and lighting setups. The collectors themselves varied in hand sizes and demonstration styles, enabling us to learn from a wide distribution of environments and interactions.

We constructed two datasets through our collection efforts: D_H (human-collected data) and D_R (robot-collected data). The human dataset D_H comprises 9,290 demonstrations across five tasks: 3,000 demonstrations from 30 different environments for each of the *Spray Bottle* and *Toy Cleanup* tasks, 621 trajectories from 6 environments for the *Pour* task, 1,545 demonstrations from 15 environments for the *Florist* task, and 1,124 demonstrations from 12 environments for the *Clothes Folding* task.

The robot dataset D_R includes 1,395 demonstrations: 388 for *Spray Bottle*, 370 for *Toy Cleanup*, 111 for *Pour*, 236 for *Florist*, and 290 for *Clothes Folding* tasks. Robot data was collected using an xArm and LEAP hand V2 Advanced. Our training and test objects are detailed in Figure 10.5.

10.5.2 Evaluation Tasks

We evaluate our approach on five diverse manipulation tasks, each designed to assess specific aspects of dexterous manipulation: functional grasping, long-horizon planning, cross-task transfer, bimanual coordination, and deformable object manipulation. A task visualization is provided in Figure 10.4.

In the Spray Bottle task, the robot grasps a spray bottle by the handle and sprays a target cloth, testing functional grasping and affordance understanding. In Toy Cleanup, the robot picks up scattered toys and places them in a bin, evaluating generalization and long-horizon planning. The Pouring task involves tilting a bottle to pour into a container, demonstrating skill transfer from the spray bottle task. In Bimanual Florist, the robot hands over a flower between its arms and inserts it into a vase, testing precise bimanual coordination. Finally, in Bimanual Clothes Folding, the robot uses both hands to fold a clothing item, assessing manipulation of deformable objects. Full task specifications and scoring criteria for all tasks are provided in Appendix E.1.

These tasks systematically evaluate DexWilds *functional grasping* capabilities, *generalization* across object types, *transferal* of skills across tasks, *coordination* between arms, and *adaptability* to deformable objects. Success requires the policy to adapt to varying object properties, environmental conditions, and task constraints.

10.5.3 Evaluation Environments

For robot experiments, we employed an xArm robot and Franka system, both equipped with either LEAP hand or LEAP hand V2 Advanced [340, 346]. Unless explicitly mentioned, xArm and LEAP hand V2 Advanced was used. We evaluate our approach across three

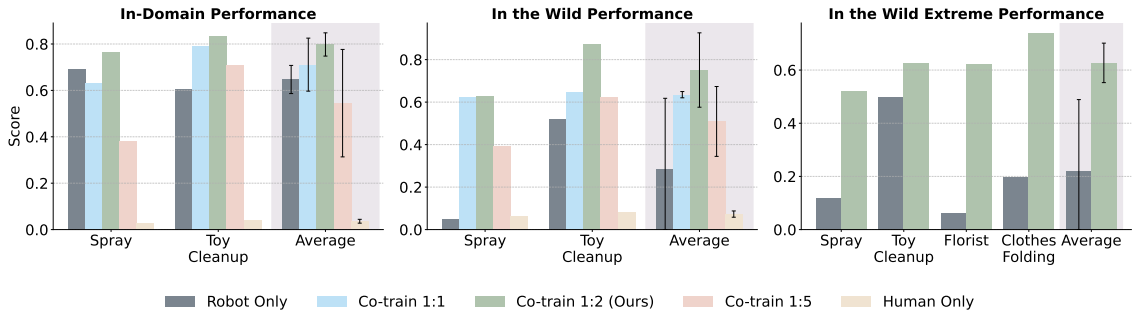

Figure 10.6: **How does co-training help with scaling up in the wild performance?** We evaluate our policy across three scenarios: (a) In-Domain scenes where robot training data was collected but with novel objects, (b) In-the-Wild scenes present in DexWild but not in robot data, and (c) In-the-Wild Extreme scenes absent from both datasets. Displayed ratio is Robot:Human.

scenarios:

1. In-Domain: Environments where robot training data was collected, testing with novel objects
2. In-the-Wild: Environments present in DexWild but absent from robot training data
3. In-the-Wild Extreme: Unseen environments absent from both datasets.

10.6 Analysis and Results

In our evaluations, we seek to investigate the following key questions:

1. How effectively does DexWild leverage human data to achieve strong in-the-wild performance?
2. Does DexWild enable policy transfer across tasks and robot embodiments?
3. Does policy performance scale effectively with increasing amounts of DexWild-System data?

Please see videos of our results at https://dexwild.github.io.

10.6.1 Zero Shot In the Wild Policies w/ DexWild

DexWild enables strong policy generalization in novel scenes. We evaluate policies in environments with increasing novelty to assess their generalization. As shown in Figure 10.6, policies trained exclusively on robot data perform well in in-domain settings (64.7%

success rate) but degrade significantly in more challenging scenarios—in-the-wild (28.5%) and in-the-wild extreme (22.0%). This 36-point performance drop suggests that robot-only policies overfit to environment-specific features and fail to develop robust, transferable representations. In contrast, policies trained only on human data learn high-level object affordances and approach objects reliably, even in complex scenes. However, without robot-specific action grounding, they struggle to execute precise manipulation, resulting in poor performance across all scenarios (3.6% in-domain, 7.3% in-the-wild).

To combine the strengths of both modalities, we adopt a co-training strategy—jointly training on both robot and human data—a method validated in prior works [117, 199, 201, 307, 369]. This encourages the policy to learn task-relevant features rather than overfitting to specific embodiments or environments. We experiment with different **robot-to-human** data ratios (1:1 to 1:5) per training batch. Our empirical analysis reveals that a 1:2 ratio yields optimal performance across all scenarios:

1. In Domain: 79.8% vs. 64.7% (robot-only)
2. In-the-wild: 75.1% vs. 28.5% (robot-only)
3. In-the-wild Extreme: 62.7% vs. 22.0% (robot-only)

Interestingly, increasing the human data ratio further (e.g., 1:5) degrades performance (54.5% in-domain, 50.9% in-the-wild), indicating that robot data remains essential for grounding fine-grained control.

DexWild extends to complex bimanual coordination tasks. To evaluate whether DexWild generalizes beyond single-arm tasks, we test it on bimanual tasks that demand precise coordination between two hands. We compare co-trained policies (1:2 ratio) against robot-only policies in in-the-wild extreme settings. DexWild policies achieve a strong 68.1% average success rate, compared to just 13% for the robot-only baseline. Even when failures occur, DexWild policies exhibit meaningful attempts at task execution—while robot-only policies often produce erratic or unstructured behavior.

These results demonstrate that DexWild not only enables robust generalization across environments but also scales to more complex manipulation behaviors.

10.6.2 Robust Cross-Task and Cross-Embodiment Generalization

DexWild enables transfer of low-level skills across tasks. Many manipulation tasks share foundational motor skills—such as lifting, orienting, and rotating objects—which opens the

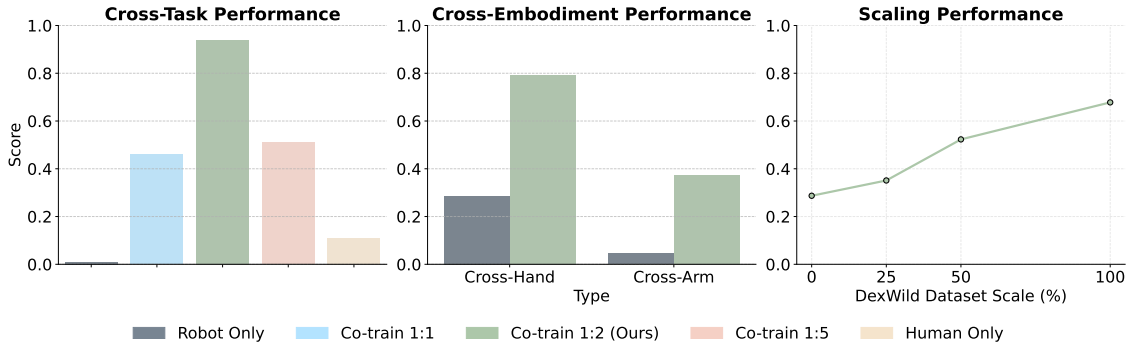

Figure 10.7: Left: **Cross-Task Performance** – Evaluating DexWild on the pour task using robot data exclusively from the spray task. Middle: **Cross-Embodiment Performance** – Testing DexWild policy on the Original LEAP hand and a Franka robot arm. Right: **Scaling Performance** – Demonstrating improved DexWild performance as dataset size increases. Displayed ratio is Robot:Human.

door to skill reuse across related tasks. For example, opening a microwave and opening a cupboard both involve similar coordination and control. We evaluate this form of cross-task transfer using the *pouring* task, which shares many motion primitives with the *spray* task. Crucially, we use no robot data for pouring and instead combine human (DexWild-System) demonstrations of pouring with robot demonstrations from spraying. This setup enables **zero-shot generalization** to pouring in in-the-wild extreme environments. Using a 1:2 robot-to-human co-training ratio, our policy achieves a **94% success rate**, far exceeding policies trained with only robot (0%) or only human data (11%).

DexWild enables transfer across robot embodiments. Since DexWild data is not tied to any specific embodiment, it naturally supports cross-platform transfer. This prolongs the value of our data, as collecting platform-specific data for every new robot is resource-intensive and impractical. We test two transfer scenarios in in-the-wild extreme scenes:

- **Cross-arm**: Transferring from an xArm to a Franka Panda arm. We achieve a 37.5% success rate, compared to 4.5% for the robot-only baseline—an **8.3× improvement**.
- **Cross-hand**: Transferring from the LEAP Hand V2 Advanced to the original LEAP Hand. We achieve 65.3% success versus 13.3% for the baseline, showing that DexWild generalizes not only across arms, but across dexterous hands as well.

These results, shown in Figure 10.7, demonstrate that DexWild enables zero-shot generalization to new tasks and hardware embodiments **without any additional robot-**

specific data, making it an efficient and general framework for dexterous policy learning on many robots.

10.6.3 Scalability of DexWild

Policy performance scales with dataset size. To understand how data scale impacts policy performance in the wild, we randomly sample subsets of the full human dataset at varying sizes and evaluate the resulting policies. We fix the size of the robot dataset. As shown in Figure 10.7, there is a clear positive correlation between dataset size and average task performance—rising from 28.7% at 20% dataset size to 67.8% with the full dataset, marking a 2.36× improvement. Interestingly, the learning curve is nonlinear, with especially steep gains in the 25–50% range, suggesting a critical threshold where the policy begins to reliably learn generalizable behaviors.

Importantly, performance continues to improve all the way to 100% data usage, indicating that the system has not yet plateaued. This suggests that even more capable policies could be learned with continued data collection.

DexWild-System enables fast and scalable data collection. Given the observed benefits of scaling, we evaluate the data collection efficiency of DexWild-System via a comparative user study measuring demonstrations per hour. As shown in Figure 10.8, DexWild-System achieves an average collection rate of **201 demos/hour** across five representative tasks—nearly matching the rate of demonstrations collected using bare hands and **4.6× faster** than a traditional robot teleoperation system based on Gello [346, 393], which achieves just 43 demos/hour.

We identify three key limitations of Gello-based collection that our system overcomes:

1. **Lack of haptic feedback:** Operators cannot feel objects, making fine manipulation difficult for certain tasks.
2. **Scene reset:** Resetting the environment is cumbersome and often requires a second operator or pauses in data collection.
3. **Hardware setup overhead:** Robots are heavy and require time-consuming setup at each new location, whereas DexWild-System is portable and can be set up in minutes.

DexWild not only demonstrates strong scaling trends with increasing data volume, but also offers a practical and efficient path to collecting diverse, high-quality data at

scale—crucial for real-world generalization.

10.7 Conclusion and Limitations

We introduce DexWild, a scalable framework for learning dexterous manipulation policies that effectively generalize to new tasks, environments, and robot embodiments. We introduce DexWild-System, a portable, human-centric data collection device that significantly accelerates dataset creation (4.6× faster than conventional robot teleoperation). We propose DexWild cotraining method, which leverages large scale human demonstrations alongside minimal robot data to achieve robust generalization-reaching a success rate of 68.5% in completely unseen environments, nearly four times higher than methods using robot data only. Furthermore, DexWild's embodiment-agnostic design enables strong cross-embodiment and cross-task transfer capabilities, reducing the need for robot-specific data.

Despite these strengths, several limitations remain that motivate future research: First, our approach still depends on a limited number of teleoperated robot data to bridge the gap between human and robot actions. Future work could explore improved retargeting or online policy adaptation to remove the need for teleoperated data. Next, because humans typically perform these tasks successfully, their demonstrations seldom include error recovery—causing trained policies to struggle to recover from unexpected failures. Adding recovery examples or adaptive strategies could boost real-world robustness. Finally, our method uses only visual and kinematic data, which limits its performance in contact-rich tasks. Incorporating tactile or haptic sensing could improve the handling of delicate interactions.

In summary, DexWild represents a significant step toward scalable, generalizable robot manipulation policies. Our results highlight the promise of leveraging human interaction

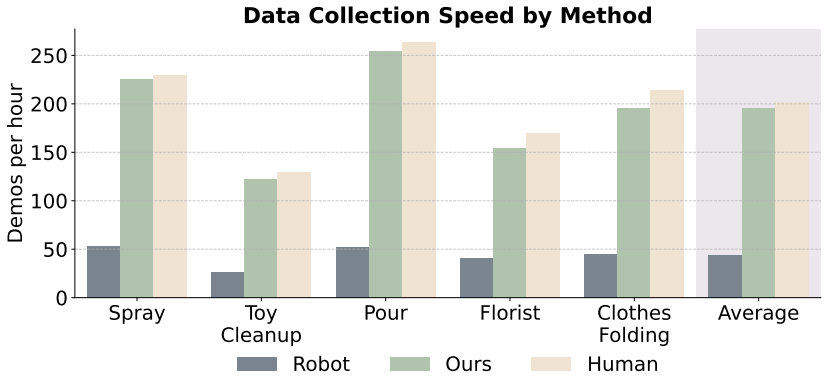

Figure 10.8: DexWild-System offers **4.6**× improvement over robot data collection speed and nearly matches the human bare hands data collection speed.

data at scale, offering an exciting avenue toward truly dexterous and versatile robots operating in diverse, real-world environments.

Videos, code, and hardware instructions are available on our website at https://dexwild.github.io

Chapter 11

IFG: Internet-Scale Guidance for Functional Grasping Generation

11.1 Abstract

Large Vision Models trained on internet-scale data have demonstrated strong capabilities in segmenting and semantically understanding object parts, even in cluttered, crowded scenes. However, while these models can direct a robot toward the general region of an object, they lack the geometric understanding required to precisely control dexterous robotic hands for 3D grasping. To overcome this, our key insight is to leverage simulation with a force-closure grasping generation pipeline that understands local geometries of the hand and object in the scene. Because this pipeline is slow and requires ground-truth observations, the resulting data is distilled into a diffusion model that operates in real-time on camera point clouds. By combining the global semantic understanding of internet-scale models with the geometric precision of a simulation-based locally-aware force-closure, IFG achieves high-performance semantic grasping without any manually collected training data. For visualizations of this please visit our website at https://ifgrasping.github.io/

11.2 Introduction

Recent advances in vision-language models (VLMs) have led to impressive results across a range of perception tasks, including image captioning, visual question answering, and open-world object recognition. Trained on large-scale datasets pairing images with natural language, these models exhibit a strong ability to align visual and linguistic information, enabling semantic understanding that generalizes across diverse contexts. This success has inspired interest in leveraging VLMs for robotics applications such as instruction following, semantic goal specification, and high-level planning.

While these initial applications show promise, significant limitations remain. Most notably, current VLMs lack a grounded understanding of physical space—they cannot reason about 3D geometry, spatial relationships, or the dynamics of physical interaction. Consequently, they struggle with planning or executing precise motor actions in the real world. Although VLMs can identify and describe visual content, they do not inherently understand how to interact with it. This disconnect between perception and control poses a major challenge in robotic grasping systems.

We seek an approach that avoids manual data collection methods like teleoperation while enabling this geometric understanding. A promising direction involves synthetic grasp generation frameworks, which produce large datasets of grasp poses through an optimization process guided by energy functions that approximate force closure, along with evaluation pipelines in simulation. These datasets are often used to train diffusion-based grasp samplers. However, a significant portion of the generated grasps are physically implausible or unnatural. Because grasp proposals are initialized by sampling points around the object's convex hull, many grasps target physically inaccessible or unsuitable regions.

Moreover, downstream manipulation tasks require the hand to interact with specific, task-relevant regions of objects such as a handle or button. Existing synthetic grasping pipelines generate grasps indiscriminately over the object surface, leading datasets that are poorly aligned with the needs of task-conditioned manipulation.

Our approach addresses this gap by combining the high-level semantic understanding of VLMs with physically grounded, task-aware synthetic grasping generation. To this end, we propose a pipeline that first translates semantic input specifying a task into predictions of useful regions on objects using a VLM. Then, we seed the grasp generation process with this prior to enable semantic-guided grasp synthesis, producing stable, natural grasps

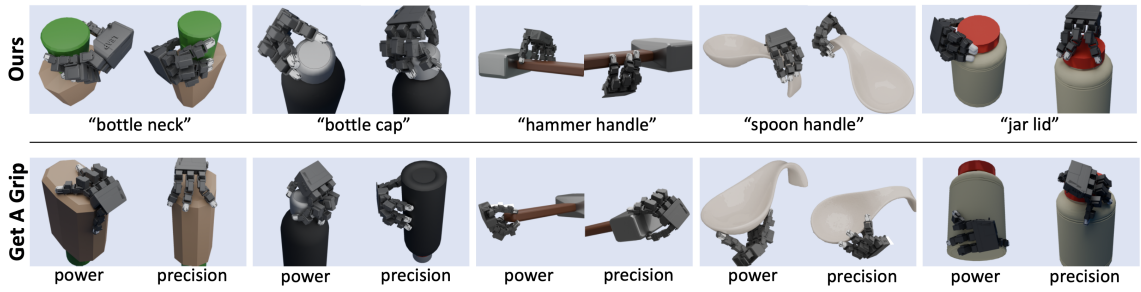

Figure 11.1: Compared to Get a Grip's synthetic grasp generation method, our method produces more human-like grasps. For instance, Get a Grip often grasp on the bottom of the bottle, while our method knows to robustly grasp the neck. Please see our website for 3D visualizations.

aligned with the demands of the task. Our pipeline is highly parallelizable, efficient, and compatible with arbitrary objects, scenes (including cluttered environments), and dexterous hands. This pipeline generates semantically meaningful grasps without any teleoperation or video data.

11.3 Related Works

11.3.1 Dexterous Grasping Generation in Simulation

GraspIt! adapts Eigengrasps from a manual database for fast grasp planning, but it doesn't generalize to any object category [266]. Similarly, [111] also uses shape augmentations. A line of works optimize for grasps contacts using differentiable simulation [63, 375, 376, 377] or by matching hand-object geometry [63]. DexGraspNet and its follow-up initialize the hand poses in the convex hull of the object and then optimize using the force closure objective [388, 410]. We use this objective function to generate grasps as well in IFG. Concurrently developed with our work, [173] uses VLMs [241, 367] to produce semantic regions for grasp generation.

Many grasping pipelines use the data generated to train neural network models. These models can aid in faster generation, the removal of privileged information, or enable generalization. Many use a VAE model [206] to enable this faster generation [147, 189, 222, 247, 410]. Other newer works use the more powerful diffusion model [177] to improve results. [390] To remove the reliance on privileged geometry, point-cloud conditioning with Pointnet [301, 302] are used [386]. A parallel technique is NeRF [247]. Finally, some use quick test time adaptation to improve the grasping quality past the learned model [222, 396].

11.3.2 Vision-based Dexterous Grasping

Instead of generating grasps in simulation there are many datasets that have human hand and object interaction in them. Some datasets have ground truth data using motion capture devices [346, 368] but are more limited in size and variety of grasps [70, 103]. Adding contact information between the object and the hand can help with fine-grained control [83, 84, 146]. While there are larger datasets available, [124, 160], they do not have ground truth hand and object poses. This means that vision methods must be used to extract this, which works in varying accuracy [319, 400, 401]. Once extracted, these grasps can be used in robot hand systems.

A wide range of recent studies have tapped into large-scale human activity datasets to improve various aspects of robot learning. Some focus on deriving cost functions from human behavior [67, 104, 251, 332], while others map human and robot actions to one another [169, 353, 411], whether through aligned demonstrations [284, 333, 343], unaligned examples [355], or direct action correspondences [325]. In addition, the inherent structure of certain datasets—such as those involving tool use [404], or temporal sequences of hand-object interactions [196, 217]—has been used to infer actions or detect salient features like keypoints.

11.3.3 VLMs for Robotic Grasping

Recent works have explored integrating large-scale models with robotic grasping, particularly for two-finger grippers. For example, [182] and [334] extract affordances and constraints from LLMs and VLMs to build 3D value maps, which are then used by motion planners to synthesize trajectories in a zero-shot manner. Similarly, [49] incorporates large-scale models but relies on simulation to train downstream policies. Other approaches [227, 275, 405] generate vision-language-action (VLA) representations or language-based plans that can be executed on robotic systems.

11.4 Method

The goal of IFG is to learn a general-purpose dexterous grasping affordance model that takes as input a scene point cloud and a text prompt specifying the object to grasp, and outputs a feasible grasp for a robot hand. To enable this, we must first generate a large grasping dataset with geometrically accurate and semantically meaningful grasps. As

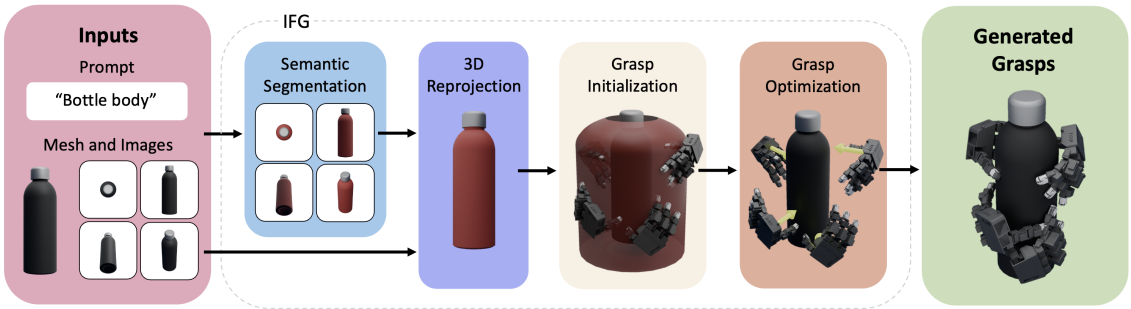

Figure 11.2: IFG takes an object mesh and a task prompt as input. To incorporate semantic understanding, it renders the object from multiple viewpoints, applies a VLM-based segmentation model combining SAM[208] and VLPart[360], and reprojects the results into 3D space to identify task-relevant regions. For geometric grounding, it initializes a force closure objective at these regions and optimizes for functional grasps. The resulting data is then used to train a diffusion model for fast grasp synthesis from depth.

shown in Figure 11.2, given an object mesh and a task prompt, our data generation pipeline identifies task-relevant regions by rendering the object from multiple viewpoints, applying a VLM-based segmentation model, and reprojecting the results into 3D. These semantic regions then guide a grasp optimization process that enforces both stability and functional relevance. The resulting diverse set of robust grasps is distilled into a diffusion model that predicts executable grasps directly from depth input, enabling fast and deployable grasp synthesis in real-world scenarios. We present the pseudocode of our method in Algorithm 5.

11.4.1 Dexterous Grasp Formulation

We formulate dexterous grasps as follows. A dexterous grasp g is defined as $g = (T, R, \theta)$, where $T \in \mathbb{R}^3$ and $R \in SO(3)$ represent the translation and rotation of the wrist pose, and $\theta \in \mathbb{R}^{\text{DoF}}$ denotes the joint angles of the hand (DoF = 16 for LEAP Hand [341]).

11.4.2 Useful Region Proposal

IFG leverages knowledge from a VLM f to identify objects of interest and part-level useful regions. To extract 2D semantic knowledge to 3D scenes, we also use a language-conditioned segmentation model g to isolate the object and a part-level segmentation model h to identify regions of interest on the object. Given an object mesh O with k faces $F = \{f^{(1)}, ..., f_{(k)}\}$, we take a set of n RGB images $V = \{v^{(1)}, \ldots, v^{(n)}\}$ from angles

uniformly sampled on a camera initialization surface S. For single object settings S is a spherical surface, and for clustered scenes it is the dome segmented from a sphere to avoid visual occlusion. A VLM is prompted with V to produce m semantic labels of useful regions of O denoted as $f(V) = R = \{r_1, \ldots, r_m\}$. For each label $p_i \in P$, g and h together produces part-level segmentation masks for each image in V conditioned on r_i, represented as $h \circ g(r_i) = S_i = \{s_i^{(1)}, \ldots, s_i^{(n)}\}$, where $s_i^{(j)}$ segments the regions of $v^{(m)}$ that belong to r_i. We use SAM [208] as the object segmentation model and VLPart [360] as the part-level segmentation model.

The 2D segmentation masks S_i are deprojected back to 3D points on the object mesh $P_i = \{p_i^{(1)}, \ldots, p_i^{(n)}\}$. However, from certain camera angles a part may be occluded, leading to incorrect segmentation. To address this, P_i further undergoes heuristic-based filtering.

A two-means based clustering process assigns each p_i to one of two groups based on the size of its segmentation mask s_i in terms of masked pixel number to filter out unsuccessful segmentation masks. The larger group from the clustering process $\hat{P}_i$ is considered the valid deprojected points. Each point is then associated with the closest face on the object mesh to produce a tally of face counts $T_i = \{t_i^{(1)}, ..., t_i^{(k)}\}$, where $t_i^{(j)} \in \mathbb{N}$. A voting algorithm selects the 60% top faces of the object mesh as the useful region of the object used for the next stage, which we refer to as U.

11.4.3 Geometric Grasp Synthesis

We compute the segmented convex hull of the object to include only faces projected from U. For each grasp, the hand is initialized on the inflated convex hull by farthest point sampling with random noise added to the pose and finger joint angles. An optimization process performs gradient descent against an energy term

$$E = E_{\text{fc}} + w_{\text{dis}}E_{\text{dis}} + w_{\text{joints}}E_{\text{joints}} + w_{\text{pen}}E_{\text{pen}} + w_{\text{spen}}E_{\text{spen}}$$

where E_{fc} approximates force closure of the grasp, E_{dis} encourages hand-object proximity, based on the contact points of the hand, E_{joints}, E_{pen}, and E_{spen} respectively penalizes joint violations, hand-object penetration, and self-penetration of the hand. For the single object setting, we exclude the tabletop by setting $w_{\text{spen}} = 0$ to produce more diverse grasps. This pipeline is similar to Get a Grip's synthetic pipeline except for a few key modifications.

Instead of using precision grasps, which sample contact points only on the fingertips of the hand, we use power grasps by sampling over the inside regions of all fingers, which produce more stable grasps and thus yield a higher success rate. For each grasp, we initialize hand positions on the segmented convex hull instead of the entire hull.

11.4.4 Simulation Evaluation

To ensure the robustness of generated grasps, we perform tasks with them in a simulation environment. Each evaluation proceeds in three phases: (1) the grasp and object are initialized in a simulation environment, (2) fingers are closed to secure the object, and (3) task execution is performed. Following Get a Grip, we use a smooth label for each grasp by applying slight perturbations on the finger joint angles to produce d associating grasps, all of which are evaluated in simulation. The hard success rates of all $d + 1$ grasps are averaged to produce the smooth label for the grasp. Grasps with low success rates are filtered out to

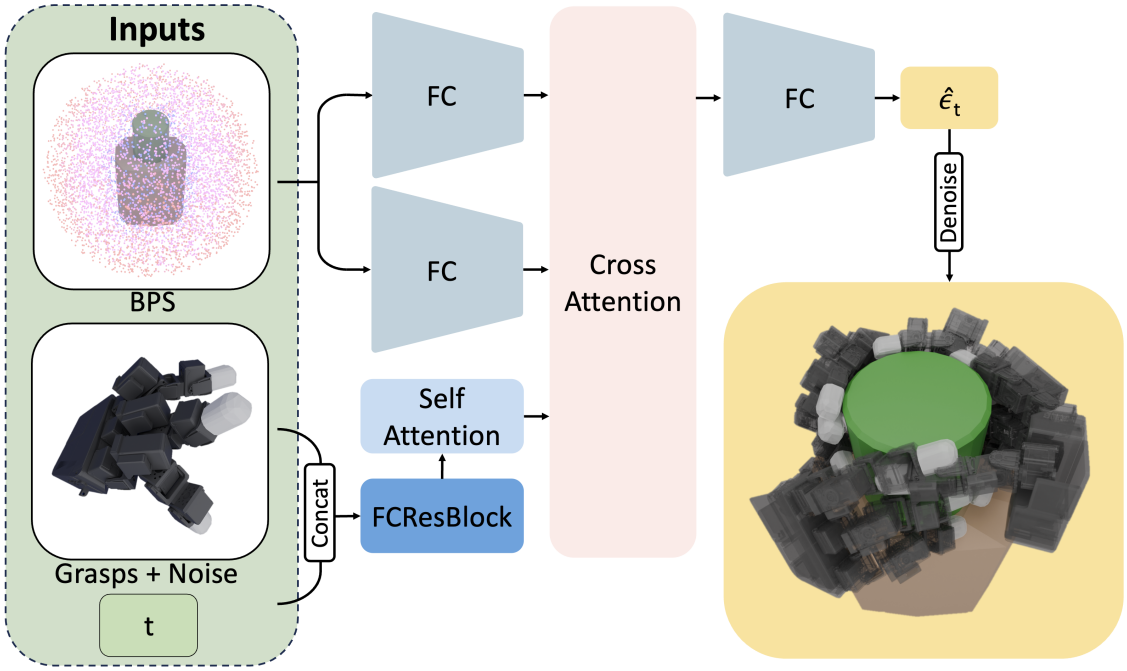

Figure 11.3: To enable real-world deployment, the generated grasp data is distilled into a diffusion model. This model is conditioned on a Basis Point Set (BPS) computed from depth camera data, along with a noisy grasp input. Through the denoising process, the model produces refined grasps on the object. The architecture of the diffusion model follows a similar design to DexDiffuser [390].

Object	GET A GRIP	OURS
water bottle	49.1	**62.8**
large detergent bottle	51.2	**62.5**
spray bottle	43.1	**54.5**
pan	48.1	**52.1**
small lamp	56.8	**85.7**
spoon	42.7	**50.9**
vase	32.2	**55.9**
hammer	45.8	**45.8**
shark plushy	19.8	**25.1**

Table 11.1: A selection of individual success rates out of the 35 objects we generate on in single-object scenes. Ours generation outperforms the baseline Get a Grip [247] due to improved grasp initializations from the VLM.

produce a dataset G of robust, force-closure power grasps. For our experiments, $d = 5$.

11.4.5 Diffusion Model Distillation

While the grasping pipeline can generate numerous candidate grasps from an object mesh, it is not directly deployable in real-world scenarios due to practical constraints. The generation process is quite slow, object mesh is not readily available, and the generation process often does not always return successful grasps. Inspired by [390], our diffusion model takes as input a Basis Point Set (BPS) which is a structured point cloud that can be readily obtained from the object mesh using a depth camera [298]. Additionally, the model receives a noisy grasp hypothesis. Through the denoising process, the diffusion model refines this noisy input into a feasible and executable grasp. This downstream diffusion model inherits both the geometric reasoning capabilities of the training pipeline and the semantic understanding provided by the vision-language model (VLM), as illustrated in Figure 11.3.

11.5 Experimental Setup

Datasets of grasps are generated on diverse objects in both single-object and clustered-scene settings, followed by extensive simulation to evaluate robustness. The evaluation addresses four key questions: (1) Can robust and stable grasps be produced on individual objects? (2) In clustered scenes, can the object of interest be identified and grasped without collision? (3) Do the resulting grasps exhibit natural, human-like qualities suitable for functional manipulation? (4) To what extent does semantic, part-level conditioning via segmentation

improve grasp robustness and naturalness?

Task Setup. Two evaluation settings are considered. In the single-object case, 24 diverse objects from Get a Grip's dataset are used; each object is sampled at 5 scales, with 200 grasps generated by both our method and the baseline. For clustered scenes, 35 dense scenes from DexGraspNet2 are selected. Each scene contains on average 3–4 objects, with 3–4 segmentation prompts per object, and 200 grasps generated for each prompt–object pair. Baselines sample 256 grasps per scene. All objects are drawn from common daily manipulation tasks, and all grasps are executed using the LEAP Hand [341]. Finally, a diffusion model is trained to verify that the generated grasps can be distilled into a policy operating directly on proprioceptive data obtainable in the real-world. Please also see our website at https://ifgrasping.github.io/ for more visualizations of these results.

Simulation Evaluation. We test our grasps in IsaacGym [253]. For the single-object setting, two tasks are designed: Lift, which raises the wrist vertically to test grasp firmness, and Pick & Shake, which lifts the object slightly and applies perturbations to the wrist. A task is considered successful if the object's relative pose to the palm remains stable throughout execution. Collision checking is enforced during the entire process. For clustered scenes, we evaluate grasps only on the Lift task, since shaking in a dense environment often leads to trivial collisions.

11.6 Results

11.6.1 Single Object Grasping

A useful grasp is not only robust but also natural, both of which can be achieved through our method. To demonstrate this, we evaluate our method against Get a Grip [247] on 35 diverse objects from their dataset used in daily scenarios. To assess robustness, grasps are evaluated in simulation under two tasks, Pick & Shake and Lift. As illustrated in Table 11.2, IFG achieves higher success rates on both tasks, demonstrating that conditioning grasp generation on part-level segmentation produces more robust grasps. Table 11.1

Method	Pick & Shake (%)	Lift (%)
Ours	16.14	51.11
Get a Grip	11.82	50.93

Table 11.2: Single-object grasp generation evaluation in Isaac Gym. Our method outperforms Get a Grip by leveraging VLM-based part-level awareness. Successful grasps are filtered and used to train the diffusion model.

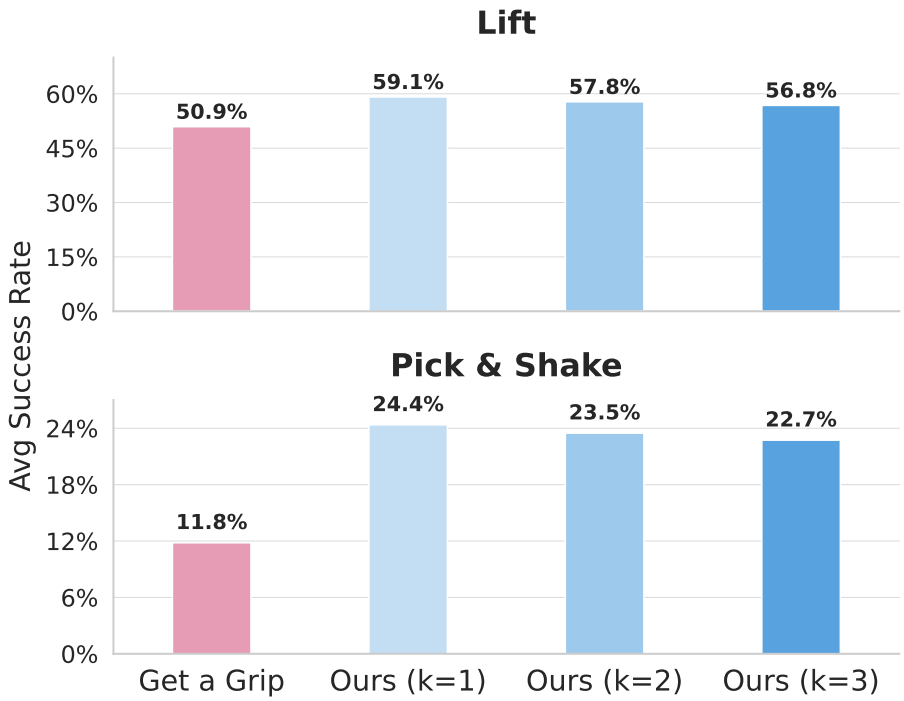

Figure 11.4: Single Object evaluation in the Lift and Pick and Shake Task. Ours outperforms on the top three segmentation prompts compared to the Get a Grip baseline generation process due to the guidance that the prompt and the VLM provide on the grasping generation process.

further presents detailed success rates of a diverse set of objects from our data. Moreover, from qualitative comparisons, our grasps are more natural: they are concentrated on the object regions that humans typically interact with in real-world use, while many of Get a Grip's grasps that pass simulation checks are not aligned with functional usage due to the absence of guidance during initialization. Shown in Figure 11.1, their grasps tend to grasp the head of a hammer since it covers a high percentage of the convex hull, while our grasps initialized on the segmented convex of the handle are functionally correct. With our method outperforming the baseline on both robustness and naturalness, we hypothesize that semantically conditioned grasps improve robustness because everyday objects are designed with affordances that support secure functional grasping, and semantic conditioning aligns grasp generation with these regions.

Method	Lift (%)
Ours	32.23
GraspTTA	25.64
ISAGrasp	32.51
DexGraspNet2	36.71

Table 11.3: IFG can generate grasps with similar lift success rates to baseline models trained on preprocessed and filtered DexGraspNet2's Dataset, showing our strong grasp generation capabilities.

Object	DexGraspNet2	GraspTTA	ISAGrasp	Ours
Tomato Soup Can	47.8	38.3	**52.0**	45.5
Mug	33.2	26.9	22.6	**60.4**
Drill	32.1	20.8	36.4	**57.5**
Scissors	9.7	0.0	**33.7**	20.2
Screw Driver	0.0	8.3	**40.0**	22.0
Shampoo Bottle	50.6	25.4	18.8	**53.1**
Elephant Figure	23.6	29.6	24.2	**35.8**
Peach Can	**61.8**	28.0	55.3	60.3
Face Cream Tube	32.1	22.5	20.7	**35.5**
Tape Roll	22.7	13.9	9.8	**43.2**
Camel Toy	12.8	14.3	21.3	**21.8**
Body Wash	40.2	22.3	29.4	**58.3**

Table 11.4: Grasp success rates for crowded-scene evaluation on the lift task. The VLM enables IFG to focus on objects of interest and exceeds them in performance compared to baselines [111, 189, 410]

11.6.2 Multi-object Dense Scene Grasping

Daily scenarios are often not so simple as single object settings because they involve many clustered objects. A grasp proposal pipeline must therefore be able to identify the object of interest and generate firm grasps while avoiding collision with others. Get a Grip does not address multi-object scenes, so we compare against the crowded scene grasp generation models in DexGraspNet2 [410]. DexGraspNet2 retargets GraspNet-1Billion data [147] into a diffusion model and adapts several single-object networks as baselines. Their approach ranks points on the scene point cloud with an MLP to propose grasp seeds, but cannot control which object is grasped. In addition, their ranking method tends to be biased toward easy targets, as shown in Figure 11.5. In contrast, our method selects via semantic

segmentation prompts and avoids overfitting to easy-to-grasp regions. We evaluate IFG on clustered, dense scenes with harder objects from DexGraspNet2. Figure 10.1 shows our grasps on four scenes on the sides. Impressively, our synthetic generation method achieves a similar success rate compared to the baseline models distilled from preprocessed and filtered data, which is shown in Table 11.3. A more detailed analysis done on individual objects across scenes is shown in Table 11.4.

The differences between ours and DexGraspNet2's reported performance is due to two reasons. (1) both methods lift objects by 20 cm, but DexGraspNet2 counts a grasp as successful if the object rises just 3 cm, even if it slips onto nearby objects. (2) More comprehensive testing: DexGraspNet2 reports only the top-confidence grasp per scene, usually on easy-to-grasp objects on the peripheral of the scene. We evaluate over 200 grasps per scene for their baselines. As shown in Figure 11.6, on harder, we outperforms them on occluded, harder-to-grasp objects that they grasp less frequently.

11.6.3 Grasp Generative Model

The method is modular, enabling plug-and-play replacement of both the segmentation and generation modules. For grasp generation, an attention-based conditional diffusion transformer (DiT) is trained to produce grasps conditioned on the object's BPS [298] representation computed from its point cloud, following an architecture similar to that used

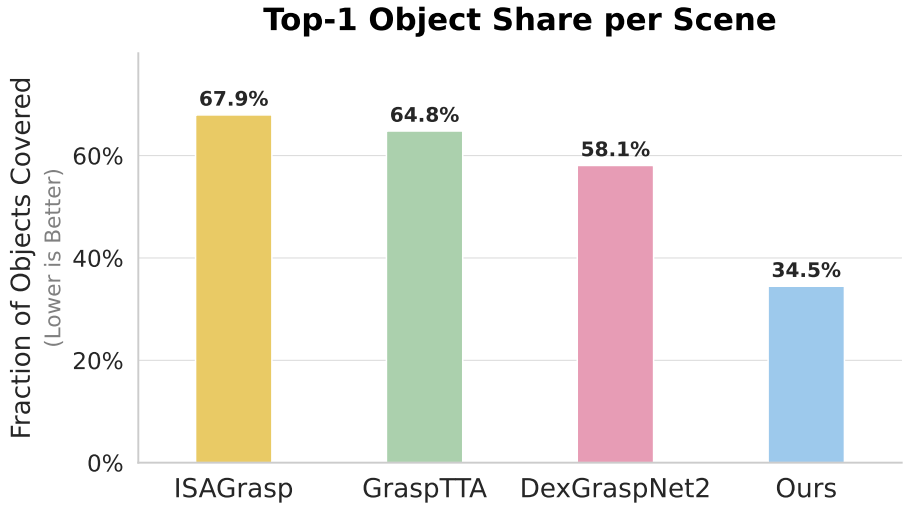

Figure 11.5: When generating grasps, confidence-based methods generate most of their grasps on the easiest-to-grasp object. On the other hand, our method can be controlled to grasp any specific object due to segmentation conditioning. Therefore the easiest object is grasped less often.

in Get a Grip. Grasps generated by this model, trained on semantically meaningful data, are compared against those produced by Get a Grip to highlight the benefits of semantic conditioning. (The model is trained on a single object at one scale, a bottle.)

11.7 Conclusion and Limitations

We introduced IFG, a pipeline that combines the semantic understanding of vision-language models with the geometric precision of force closure grasping to generate functional and robust dexterous grasps. IFG harnesses internet-scale models to identify task-relevant object regions from visual information, uses them as semantic conditioning for energy based force closure optimization, and leverages simulation evaluation as a metric for robustness. As a result, IFG produces more natural and effective grasps than prior methods, particularly in cluttered environments. The resulting data is distilled into a diffusion model, enabling real-time grasp prediction from camera input without relying on hand-collected data.

Nonetheless, our work has limitations. First, our method does not account for dynamic objects since our segmentation is performed on images from a single timestep. Potential

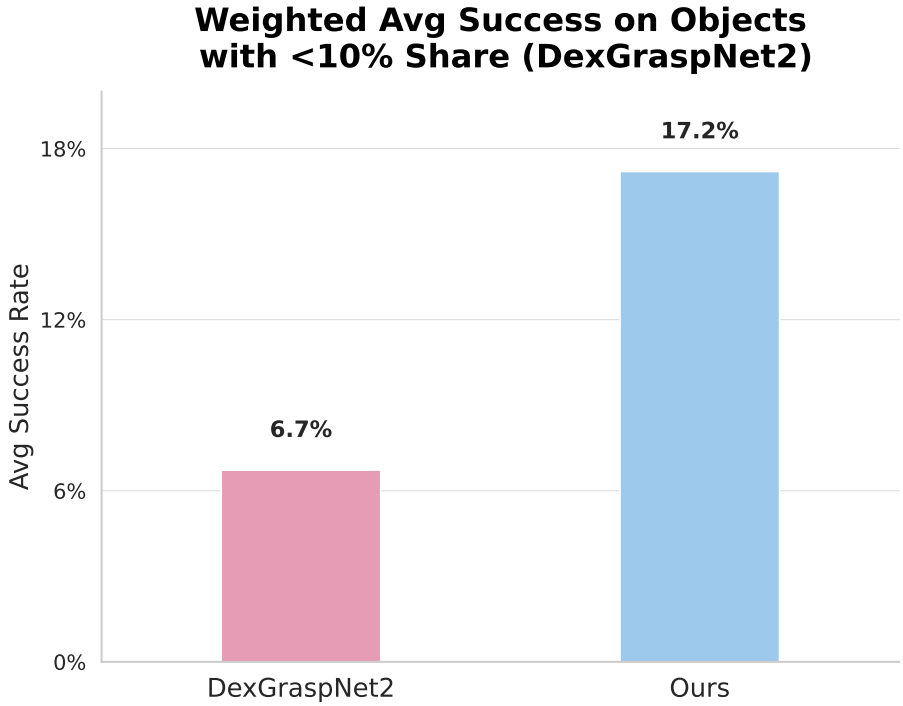

Figure 11.6: DexGraspNet2's grasp generation model avoids hard-to-grasp objects. Our method concentrates more on these objects and achieves a higher success rate due to functional guidance from VLM-based segmentation.

work can be done on extending our semantic segmentation pipeline for continuous video streaming. Additionally, our method is not suited for scenarios where non-force closure grasps are required. There is still much to be done in optimization based dexterous grasp generation.

Algorithm 5 IFG

Require: VLM f, segmentation models g, h, object mesh O with faces $F = \{f^{(1)}, \ldots, f^{(k)}\}$, n views $V = \{v^{(1)}, \ldots, v^{(n)}\}$ from camera surface S

Semantic Segmentation Region Extraction

0: Query VLM: $R = f(V) = \{r_1, \ldots, r_m\}$ semantic labels
0: **for** each label $r_i \in R$ **do**
0: Obtain masks $S_i = h \circ g(r_i) = \{s_i^{(1)}, ..., s_i^{(n)}\}$
0: Deproject masks: $P_i = \{p_i^{(1)}, ..., p_i^{(n)}\}$
0: Filter P_i with two-means clustering by mask size
0: Map filtered points $\hat{P}_i$ to nearest faces, tally counts T_i
0: **end for**
0: Select top 60% faces $U \subseteq F$ as useful regions

Geometric Grasp Synthesis

0: Build convex hull from U; inflate for sampling
0: **for** each grasp initialization **do**
0: Place hand on hull via farthest point sampling + random noise
0: Optimize energy

$$E = E_{\text{fc}} + w_{\text{dis}}E_{\text{dis}} + w_{\text{joints}}E_{\text{joints}} + w_{\text{pen}}E_{\text{pen}} + w_{\text{spen}}E_{\text{spen}}$$

0: Obtain candidate grasp
0: **end for**

Simulation Evaluation

0: **for** each grasp **do**
0: Generate d perturbed grasps by varying joint angles
0: Simulate Lift / Pick & Shake tasks in IsaacGym
0: Assign smooth label as mean success over $d + 1$ trials
0: **end for**
0: Filter grasps with low success $\rightarrow G$ =0

Chapter 12

Conclusions

In this thesis, we investigated how to build and teach anthropomorphic robot hands. The first part of this problem is the hand itself. We created open-source robot hands that are stronger, more dexterous, lower-cost, and easier to use than conventional alternatives. These hands were designed specifically for robot learning: they are robust enough for repeated real-world interaction, simple enough for other researchers to build and repair, and anthropomorphic enough to support human-like grasping and learning from human data. Through LEAP Hand, DASH, and the later LEAP Hand v2 systems, we showed that careful hand design, soft compliance, tendon-driven actuation, and accessible fabrication can make dexterous manipulation practical for a much broader research community.

The second part of the problem is teaching these hands how to act. Rather than learning every task from scratch, this thesis explored how robot hands can benefit from the same source of experience that humans constantly produce: demonstrations, interaction, and video of people using their hands. Through Robotic Telekinesis, VideoDex, and DEFT, we showed that internet-scale human behavior can provide useful priors for teleoperation, policy learning, and affordance learning. Robots can learn to retarget human motion into their own embodiment, use this retargeted behavior as supervision, and then improve their policies through physical interaction in the real world.

Finally, this thesis showed that dexterous manipulation cannot be solved by hardware or learning alone. Fine-grained dexterity requires capable hands, scalable data, and learning systems that can turn human behavior and robot experience into useful action. We therefore extended these ideas toward more complex manipulation by combining stronger and more

compliant hands, motion-capture-based teleoperation, bimanual data collection, in-the-wild human demonstrations, and simulation-based grasp generation. Together, these works show that progress in dexterous manipulation comes from jointly improving the hand, the data, and the learning system.

Although this chapter is titled as a conclusion, it is really a checkpoint in a longer research journey. In the future, I will join the University of Illinois at Urbana-Champaign as an Assistant Professor, where I hope to continue pursuing this direction as a faculty member. I believe that capable robots will emerge from the intersection of hardware and machine learning: better robot bodies, better sensing, and better learning systems designed together rather than in isolation. My future lab will continue to build high-performance, open-source mechanisms, especially for manipulation, while exploring touch sensing, learning from human data, large-scale training, simulation, and real-world experience. The goal is to develop robots that are not only more capable in controlled environments, but more useful, robust, and adaptable in the real world.

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 [94] from the authors of FrankMocap [320]. 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 [320]. This network uses a ResNet50 trunk [174], followed by a Multi-Layer Perceptron (MLP) regression head, which outputs three relevant parameters of the SMPL-X model [287]. (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 [168], 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 [168], 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 [378] 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 $\text{concat}(\beta_h, \theta_h) \in \mathbb{R}^{55}$ 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 [420] and "in-the-wild" human hand poses from the 100 Days of Hands dataset [331]. 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 [320]). 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 [320]. This network uses a ResNet50 trunk [174], 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 [320], 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

transformation 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 [89], implemented in PyBullet [121], 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 [320].
- **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 [174] features initialized from ImageNet [138] training instead of the R3M [277] features, and the results in `VideoDex-ImageNet`. We can see that performance drops off, which indicates that the visual priors are important.

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.

Note that all of the reported numbers are on test objects. We present the results in Table B.1.

B.3 Retargeting Details

We first retarget human videos from Epic-Kitchens [122]. 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 [326] 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 [93] 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 [326] can take hours to process larger video clips, but ORBSLAM3 [93] 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. [352] and Handa et al. [169]. Specifically, we use the the detected human hand poses using MANO Romero et al. [317] (and FrankMocap Rong et al. [320]) 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. [352], we distill this into a single neural network

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

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

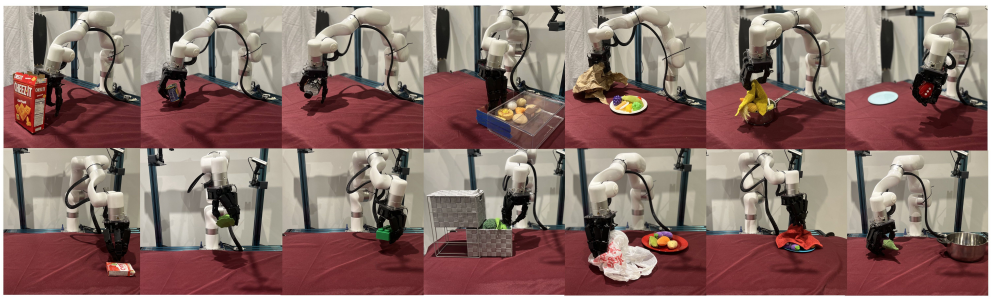

Figure B.1: **Task Images**. A more detailed look at the tasks completed by VideoDex: push, pick, rotate, open, cover, uncover and place. and our website at https://videodex.github.io for further details.

	Pick		Rotate		Open		Cover		Uncover		Place		Push	
	train	test	train	test	train	test	train	test	train	test	train	test	train	test
BC-NDP [66]	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[130]	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 [130]	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.

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 [122] that were as close as possible to these tasks, and doing similar types of actions. More details can be found in Table D.3.

B.4 Learning Pipeline Details

Learning Setup For our approach, we use the ResNet18 from R3M [277] weights as the visual backbone. This produces an intermediate feature vector of size 512. This is processed 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 [65]. The NDP processes the input with a single hidden layer to project it into the desired size (parameters W and g).

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 [65] Basis Functions N	300
NDP [65] Global Parameter α	15

Table B.4: Parameter List

For more information we point the readers to Bahl et al. [65]. We use the implementation from Dasari et al. [130]. 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 [277]) 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 D.3.

B.5 Experimental Setup

We collect data using a dexterous hand robotic teloperation setup [169, 352]. 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.

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. [352] and Handa et al. [169]. We use

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.

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 [320], https://github.com/facebookresearch/frankmocap
- NDP and Behavior Cloning [130] code from https://github.com/AGI-Labs/robot_baselines
- Code for R3M [277] https://github.com/facebookresearch/r3m
- CQL baseline from Takuma Seno [366] (https://github.com/takuseno/d3rlpy)
- COLMAP from Schönberger et al. [326] (https://github.com/colmap/colmap)
- ORBSLAM3 from Campos et al. [93] (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) [261] 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 [161], EpicKitchens-100 [122], and HOI4D [245]. 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 [277] 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 [308] 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 [320], 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 [205] 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.

Appendix D

Experimental Details for Bimanual Dexterity for Complex Tasks

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

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

D.2 Detailed Cost Analysis

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

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

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

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

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

D.3 User Study

To further evaluate BiDex, we conducted a user study involving novice users who tested both BiDex and the Apple Vision Pro system. Each participant performed the Pringles handover task over 10 trials, following a 3-minute practice session with each system. We collected data on average task completion times and success rates, as well as users' ratings (on a scale from 1 to 5) regarding accuracy, responsiveness, ease of use, and their confidence in each system. The results are summarized in Fig. D.7.

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

responsive, and easier to use, and they expressed greater confidence in using it.

D.4 Policy Performance Comparison

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

In general, we found that policy performance scales with the number of demonstrations, an unsurprising result which highlights the need of efficient and effective teleoperation systems to collect large amounts of robotic data. Before conducting the evaluation, we hypothesized that policies trained on different data sources would achieve similar performance given the same number of demonstrations. However, to our surprise, policies trained from the Apple Vision Pro perform worse, mainly due to abrupt wrist movements, as shown in the "Apple Vision Pro vs Ours" section. We hypothesize that the difference in action spaces between two teleoperation systems results in the performance differences. The Apple Vision Pro commands in end-effector space, and a small prediction error can result in large errors in joint space. While BiDex commands directly in joint space, which results in smoother actions.

D.5 About Manus Glove

We use the Manus Meta Quantum Metagloves [24] which is an $6000 tracking Mocap glove. Each finger is tracked by the glove and returns the fingertip positions as xyz-quaterion and also 4 different angles for each finger $\theta_{\text{MCP}_{\text{side}}}, \theta_{\text{MCP}_{\text{fwd}}}, \theta_{\text{PIP}}, \theta_{\text{DIP}}$ using hall effect sensors

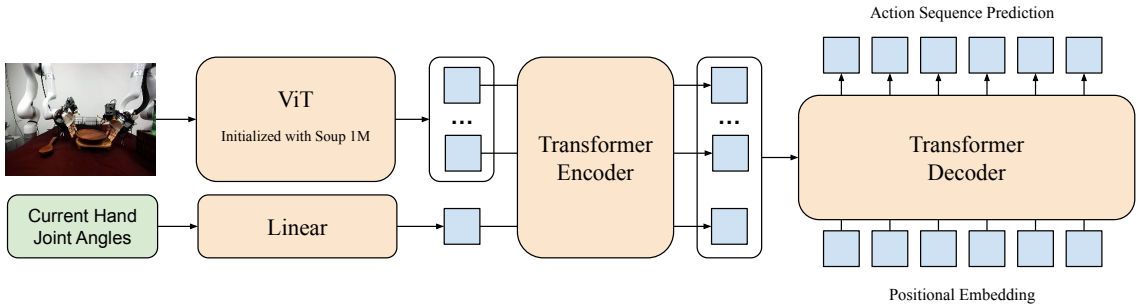

Figure D.1: Behavior Cloning Policy Architecture

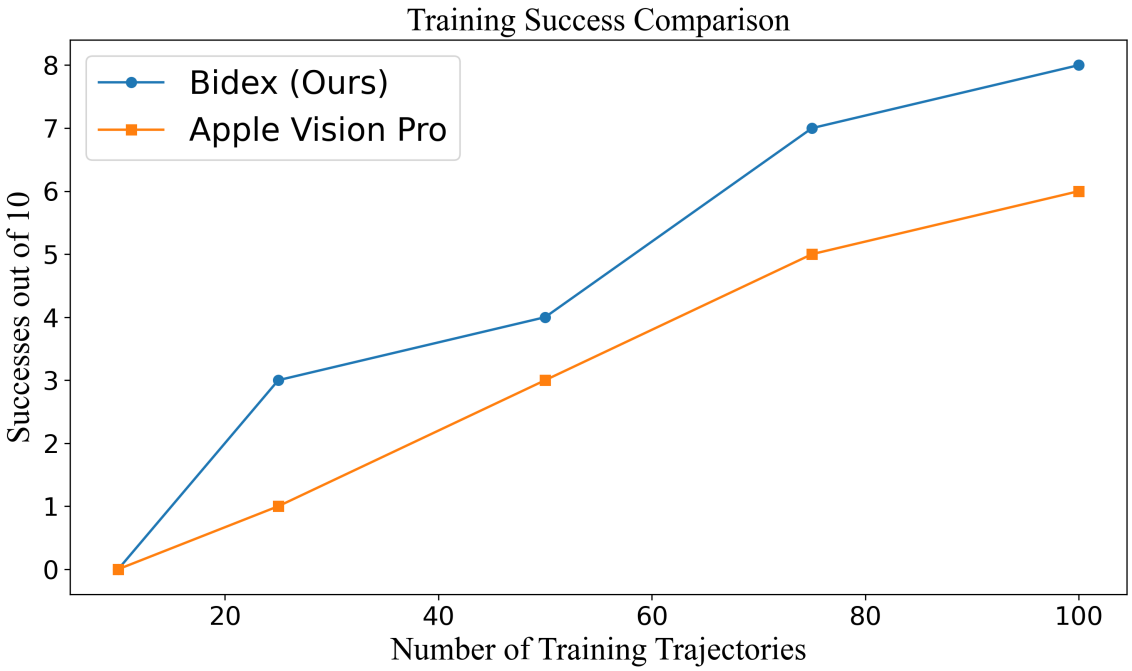

Figure D.2: Training autonomous policies with BiDex enables a higher success rate with less data than an Apple Vision Pro baseline. This is thanks to the higher quality data.

with very high accuracy and at 120hz. We use their Windows API (Linux is not available at time of release) and release our version of that which sends the software to a Linux machine running ROS. These gloves are available for purchase at https://www.manus-meta.com/ and our software is available on our project website

D.6 SteamVR Baseline

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

D.7 Apple Vision Pro Baseline

The Apple Vision Pro baseline is based off of [282]. With this data, we control the hand using the same inverse kinematics as with the Manus Glove. For the arm, we scale, translate

and rotate for the robot embodiment and then pass through inverse kinematics to control the arm.

D.8 Behavior Cloning Policy Architecture and Hyperparameters

We illustrate our policy architecture in Figure D.1. Our behavior cloning policy takes as input a RGB image and current hand joint angles (proprioception). We obtain tokens for the image observation via a ViT [141] and a token for joint proprioception via a linear layer. The weights of ViT is initialized from the Soup 1M model from [132]. The tokens then pass through a action chunking transformer [413], a encoder-decoder transformer, to output a sequence of actions. The action space is the absolute joint angles of two arms and two hands. A key decision that greatly improves policy generalization is to exclude current arm joints from the proprioception. Intuitively, this may force the model to extract object information from image observations, rather than overfitting to predict actions close to current arm states.

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

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

Table D.3: Hyperparameters for Behavior Cloning Policy Training

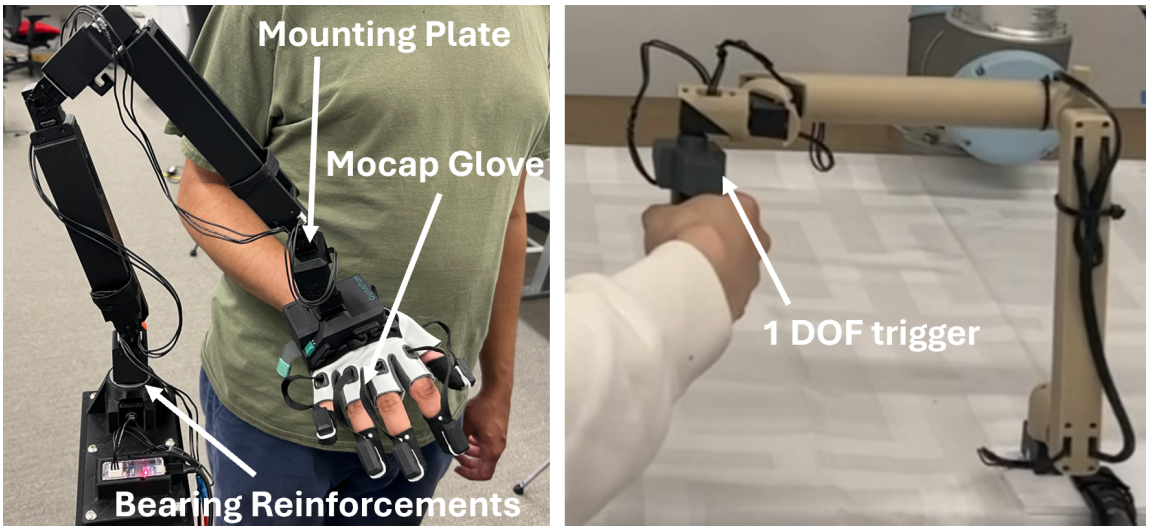

Figure D.3: We compare BiDex to the GELLO system designed for 1 DOF grippers. Our system uses a motion capture glove to capture full fingertip information and is reinforced to handle the wears and tears of this additional weight.

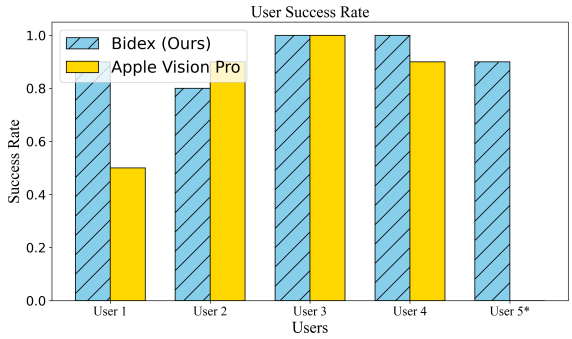

Figure D.4: Bidex has consistently better user success rates.

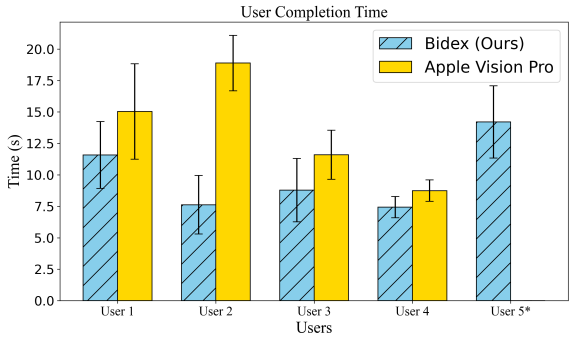

Figure D.5: Bidex is faster as seen in these user mean and standard deviation completion times.

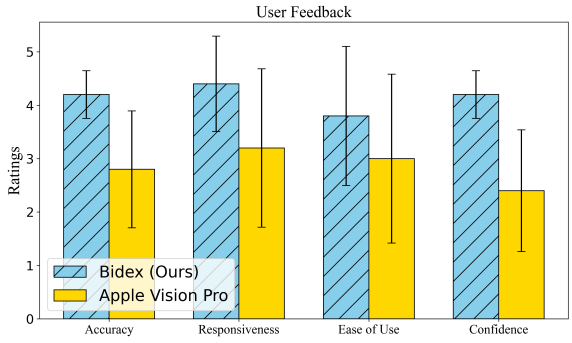

Figure D.6: Users feedback on accuracy, responsiveness, ease of use, and confidence in using each system shows that Bidex is empirically easier to use.

Figure D.7: Novice operators were asked to complete the Pringles can handover tasks for ten trials with BiDex and the Apple Vision Pro. * User 5 found it hard to control the system with the Apple Vision Pro and broke robot hands during operation, resulting in zero success rate.

Appendix E

Experimental Details for DexWild

Videos of our results, code to recreate our system, and hardware instructions are available on our website at https://dexwild.github.io

E.1 Detailed Task Description and Scoring Criteria:

We evaluate five dexterous manipulation tasks, each designed to assess different capabilities such as functional grasping, long-horizon planning, precision, bimanual coordination, and deformable object manipulation. Each task is scored according to a structured rubric based on discrete completion milestones.

The task scoring criteria are designed to quantify the performance of different robot tasks based on specific completion milestones. Each task has a set of defined actions with corresponding point values. Higher scores are assigned to more complex or functionally successful actions, while partial completions and failed attempts receive lower scores. This structured scoring system allows for consistent evaluation and comparison of task performance.

Spray Bottle

This task evaluates functional grasping and affordance understanding. The robot must grasp a spray bottle and orient it to spray over a target cloth.

- 0.00: Nothing
- 0.15: Tries functional grasp but fails

- 0.25: Grasp bottle
- 0.75: Grasp bottle, orient over cloth
- 0.75: Grasp bottle, use functional grasp
- 1.00: Grasp bottle, use functional grasp, orient over cloth

Toy Cleanup

This task tests long-horizon planning and generalization. The robot must collect scattered toys and deposit them in a designated bin.

- 0.00: Nothing
- 0.25: Tries for grasp but fails
- 0.50: Grasp object
- 1.00: Grasp object, drop into bin

Pouring

This task assesses precise motion control and transfer learning from the spray bottle task. The robot must pour liquid from a bottle into a container.

- 0.00: Nothing
- 0.15: Tries functional grasp but fails
- 0.25: Grasp bottle
- 0.75: Grasp bottle, pour into container
- 0.75: Grasp bottle, use functional grasp
- 1.00: Grasp bottle, use functional grasp, pour into container

Bimanual Florist

This task evaluates coordinated control of both hands. The robot must pick up a flower, hand it to the other arm, and insert it into a vase.

- 0.00: Nothing
- 0.15: Tries grasp but fails
- 0.25: Grasp the bouquet
- 0.75: Grasp the bouquet, handover
- 1.00: Grasp the bouquet, handover, insert into vase

Clothes Folding

This task tests manipulation of deformable objects using both hands. The robot must fold a clothing item placed on a surface.

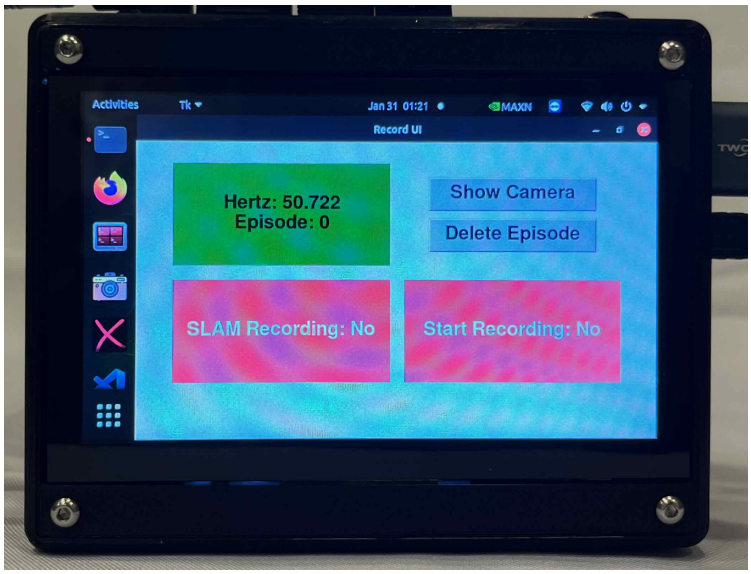

Figure E.1: DexWild-System features a simple and easy-to-use interface for deployment by untrained data collectors.

- 0.00: Nothing
- 0.25: Tries grasp but fails
- 0.50: Grasp with one hand
- 0.75: Grasp with both hands
- 1.00: Grasp and fold

E.2 Data Collection Procedure

To deploy DexWild-System with untrained data collectors, we provide a one-page instruction sheet outlining the task, object setup, and system startup/shutdown. DexWild-System includes three core components: a wrist-tracking camera, a battery-powered mini-PC for onboard data capture, and a custom sensor pod with a motion-capture glove and palm-mounted cameras. At a new site, users simply wear the mocap glove and power on the mini-PC with a provided power bank. For egocentric tracking, a headstrap holds the tracking camera; for exocentric tracking, we provide a collapsible tripod. Once booted, users launch our custom desktop app and control recording via a Bluetooth clicker or foot pedal. The UI (Fig. E.1) shows sensor status, SLAM recording, and data capture indicators, along with

buttons to view the tracking camera feed and delete the last episode. Collectors gather 100 episodes per location. After the day is finished, we upload the data to our remote machine for processing.

E.3 Downstream Data Processing

Each episode is stored in its own folder, with subfolders organizing individual actions and observations. SVO recordings from the Zed Mini camera—used for SLAM and wrist pose tracking—are saved separately, with each file covering five episodes. To begin data processing, we use the Zed SDK to decode these SVO files, reconstruct the camera's motion, and perform ArUco cube tracking and wrist pose estimation using both the left image and stereo depth data. We then apply a filtering pipeline to assess tracking quality; episodes are discarded if the wrist pose cannot be reliably tracked for more than 75% of the duration. Next, we compute the action distribution and clip outliers outside the 2nd and 97th percentiles. We smooth the trajectories using interpolation and Gaussian filtering to ensure fluid motion. Hand motions are then retargeted using inverse kinematics in PyBullet, following the method in [346]. The entire pipeline is parallelized using Ray for efficiency.

E.4 Behavior Cloning Policy Architecture and Training Hyper-Parameters

Our behavior cloning policy takes as input RGB images and relative state history. We obtain tokens for the image observation via a ViT and tokens for relative states via linear layers. The weights of ViT is initialized from the Soup 1M model from [131]. We decide to include relative states as we found it greatly increases the robustness of the policy, and enables smoother motions. In particular, for bimanual tasks, we find that including the interhand pose (pose of left hand relative to right hand) greatly increases success rate in tasks like Florist We implement both Action Chunking Transformer [414] and Diffusion U-Net [114] as policy classes, which output a sequence of actions. The network outputs actions which consists of relative end effector actions and absolute hand joint angles.

We list the hyper-paramaters that we used for policy training using behavior-cloning in this Table E.5

E.5 Low Level Motion Control

For optimal smoothness of our policies and safety, we employ a Riemannian Motion Policy (RMP) [312] implemented in Isaac Lab [267], where the RMP dynamically generates joint-space targets given end effector targets. RMP also has the added benefit of incorporating real-time collision avoidance, preventing self-collision between the arms and a set table height. Although our policies does not rely on RMP to prevent collisions, the peace of mind is appreciated.

E.6 Comparing Policy Classes

Does DexWild work with different behavior cloning policy classes? Table E.1 compares the performance of ACT and Diffusion—across both the In-the-Wild and In-the-Wild Extreme settings. Each policy is evaluated in a robot-only setting and a co-trained (1:2) setting using the DexWild dataset. Notably, Diffusion policies benefit more from DexWild co-training, achieving the highest scores in all tasks, including substantial improvements on the Pour task where the policy must generalize across tasks. These results suggest that DexWild co-training enables stronger generalization, especially when paired with expressive policy architectures like Diffusion.

E.7 Cross Hand Extended Results

Does DexWild generalize across different robot hands? Table E.2 reports LEAP Hand performance under both *In the Wild* and *In the Wild Extreme* conditions. In every case, DexWild co-training substantially outperforms the robot-only baseline. These results highlight the effectiveness of DexWild in cross embodiment generalization even when using a completely different robot hand.

E.8 Scaling Extended Results

Does DexWild improve as more DexWild data is added? Table E.3 shows steady gains as we scale from 0% to 100% of the DexWild dataset. Performance increases steadily

with more human demonstrations, with a notable jump between 25% and 50% of the dataset. These results demonstrate that DexWild enables scalable learning, where even comparably smaller data scales yields substantial gains, and additional data continues to enhance generalization

E.9 Cotraining Extended Results

How does DexWild react to different cotraining ratios? Table E.4 groups all three raw metrics: (a) In-Domain, (b) In-the-Wild, and (c) In-the-Wild Extreme. All evaluations were run on xArm + LEAP Hand V2 Advanced.

Task	Policy Class	In the Wild		In the Wild Extreme	
		Robot Only	1:2	Robot Only	1:2
Spray	ACT	0.000	**0.680**	0.115	**0.395**
	Diffusion	0.050	**0.628**	0.120	**0.520**
Toy Cleanup	ACT	0.458	**0.583**	0.125	**0.458**
	Diffusion	0.521	**0.875**	0.500	**0.625**
Pour (Cross Task)	ACT	0.025	**0.508**	0.000	**0.350**
	Diffusion	0.000	**0.958**	0.000	**0.917**

Table E.1: DexWild Performance on Different Policy Classes

	In the Wild		In the Wild Extreme	
Task	Robot Only	1:2	Robot Only	1:2
Spray	0.305	**0.805**	0.150	**0.600**
Toy Cleanup	0.500	**0.656**	0.250	**0.542**
Pour (Cross Task)	0.050	**0.917**	0.000	**0.817**

Table E.2: LEAP Hand Performance on In-the-Wild and In-the-Wild Extreme Tasks. Ratio is Robot:Human

Scale	**0%**	**25%**	**50%**	**100%**
Spray	0.060	0.260	0.605	0.565
Toy Cleanup	0.514	0.442	0.440	0.792
Average	**0.287**	**0.351**	**0.523**	**0.678**
Std	**0.321**	**0.129**	**0.116**	**0.160**

Table E.3: Performance Scaling with DexWild Dataset Size

Task	Robot	1:1	1:2	1:5	Human
Spray	0.690	0.630	0.763	0.381	0.030
Toy Cleanup	0.604	0.792	0.833	0.708	0.042
Average	**0.647**	**0.711**	**0.798**	**0.545**	**0.036**
Std	**0.061**	**0.114**	**0.050**	**0.232**	**0.008**

(a) In Distribution Task Performance

Task	Robot	1:1	1:2	1:5	Human
Spray	0.050	0.625	0.628	0.393	0.063
Toy Cleanup	0.521	0.646	0.875	0.625	0.083
Average	**0.285**	**0.635**	**0.751**	**0.509**	**0.073**
Std	**0.333**	**0.015**	**0.175**	**0.164**	**0.015**

(b) In-the-Wild Task Performance

Task	Robot	1:2
Spray	0.120	0.520
Toy Cleanup	0.500	0.625
Bimanual Florist	0.063	0.623
Bimanual Clothes Folding	0.198	0.740
Average	**0.220**	**0.627**
Std	**0.195**	**0.090**

(c) In-the-Wild Extreme Task Performance

Table E.4: Performance Across Cotrain Ratios for Varying Deployment Conditions. Ratio is Robot:Human

Hyperparameter	Value
Training Configuration	
Optimizer	AdamW
Base Learning Rate	3e-4
Optimizer Momentum	$\beta_1, \beta_2 = 0.95, 0.999$
Learning Rate Schedule	Cosine (diffusers)
Warmup Steps	2000
Total Steps	70000
Batch Size	256
Environment Frequency	30 Hz
Observation Settings	
Proprioception Horizon	1 (Spray, Toy, Pour) 3 (Florist, Clothes)
Image Horizon	1 (all tasks)
Observation Resolution	224×224
Observation Dim	9 (Spray, Toy, Pour) 27 (Florist, Clothes)
Action Dimension	26 (Spray, Toy, Pour) 52 (Florist, Clothes)
Action Chunk Size	48

Hyperparameter	Value
Action Chunking Transformer	
# Encoder Layers	4
# Decoder Layers	6
# MHSA Heads	8
Feed-Forward Dim	3200
Hidden Dim (Token Dim)	768
Dropout	0.1
Feature Norm	LayerNorm
Diffusion U-Net Policy	
Train Diffusion Steps	100
Eval Diffusion Steps	16
Down Channels	[256, 512, 1024]
Kernel Size	3
Groups (GN)	8
Dropout	0.1
Feature Norm	None

Table E.5: Full training and architecture settings used across our experiments.

Bibliography

[1] Recreus filaflex. https://recreus.com/gb/. 7.3, 8.3

[2] Haptx. https://haptx.com/. 3.2

[3] Ninjatek ninjaflex edge. https://ninjatek.com/shop/edge/. 2.3, 7.3, 8.3

[4] Ability hand. https://www.psyonic.io/ability-hand. 7.3, 8.3

[5] Shadowhand. https://www.shadowrobot.com/. (document), 1.2, 2.2, 2.3, 2.4, 2.5.1, 7.2, 7.3, 8.3, 8.4.3, 8.6

[6] Unitree a1. https://www.unitree.com/en/a1. 2.2, 2.5.1

[7] Allegro hand. https://www.wonikrobotics.com/research-robot-hand. 1.2, 2.1, 2.2, 2.3, 2.4, 2.5.1, 2.5, **??**, 2.7, 4.4.2, 4.7, 7.3, **??**, 8.3, **??**, 8.6

[8] Asimo by honda. https://asimo.honda.com/. 1.1

[9] Boston dyanmics atlas. https://bostondynamics.com/atlas/. 8.3

[10] Bambu ams. https://bambulab.com/en. 7.3

[11] Clone robotics. https://www.clonerobotics.com/. 2.3

[12] Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/. 3.3, 4.3, 6.3

[13] Dex hand. https://www.dexhand.org/. 1.2, 7.3, 8.3

[14] Digit from agility robotics. https://agilityrobotics.com/. 8.3

[15] Robotis dynamixel. https://www.robotis.us/dynamixel-xc330-m288-t/. Accessed on 2022-11-26. 2.6

[16] E3d toolchanger. https://e3d-online.com/. 7.3

[17] Creality ender 5 3d printer. https://www.creality.com/. 2.6

[18] Figure.ai. https://www.figure.ai/. 8.3

[19] Franka panda. https://www.franka.de/. 2.2, 2.5.1

[20] Unitree go1. https://www.unitree.com/en/go1. 2.2, 2.5.1

[21] Healthline. https://www.healthline.com/health/average-hand-size#adults. Accessed

on 2022-11-29. 5.5.2

[22] Inmoov hand. https://inmoov.fr/. (document), 1.2, 2.1, 2.2, 2.3, **??**, 7.2, 7.3, **??**, 8.3, **??**

[23] Inspire hand. https://en.inspire-robots.com/. 8.3, 8.4.1, 8.6

[24] Manus. https://www.manus-meta.com, note=Accessed on 2022-11-28. 2.7.1, 5.4.3, 5.7, 7.7.1, 8.4.4, 8.5.1, 9.2, 9.4, 9.4.1, D.5

[25] Ninjaflex edge. https://ninjatek.com/shop/edge/. Accessed on 2022-11-26. 5.4.2

[26] Oculus rift. https://www.oculus.com/rift/. 3.2

[27] 1x eve. https://www.1x.tech/. 8.3

[28] Tesla optimus. https://www.tesla.com/AI, . 8.3

[29] Tesla optimus block sorting. https://youtu.be/D2vj0WcvH5c?si=KVGRy4Hvx0YhcZb3, . 1.1

[30] Pneuflex tutorial. https://www.robotics.tu-berlin.de/menue/software_tutorials/pneuflex_tutorial/. 2.3, 7.3, 8.3

[31] Polymaker. https://polymaker.com/. 7.3

[32] Prusa xl. https://www.prusa3d.com/. 7.3

[33] Ability hand. https://www.psyonic.io/ability-hand. 9.3

[34] Reprap open-source 3d printer. https://www.reprap.org/wiki/RepRap. 2.3, 7.3, 8.3

[35] Robotis dynamixels. https://www.robotis.us/. 2.5.2

[36] Sanctuary ai. https://sanctuary.ai/. 8.3

[37] Snapmaker j1s. https://snapmaker.com/. 7.3, 7.4.1, 8.4.1

[38] Steam vr. https://www.steamvr.com/en/. Accessed on 2023-02-03. 5.4.3, 9.2, 9.3, 10.4.1

[39] Ultimaker. https://ultimaker.com/. 7.3

[40] Ur5. https://www.universal-robots.com/products/ur5-robot/. 2.5.1

[41] Colorfabb varioshore. https://colorfabb.com/varioshore-tpu-medium-brown. 7.3, 7.4.1, 8.3, 8.4.1

[42] Htc vive. https://www.vive.com/. 3.2

[43] xarm6 by ufactory. https://www.ufactory.cc/xarm-collaborative-robot, . 2.2, 2.5.1, 4.3, 4.6, 4.2

[44] Ufactory xarm6. https://www.ufactory.cc/product-page/ufactory-xarm-6, . Accessed on 2023-02-03. 5.4.2

[45] 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. 5.2

[46] 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. 1.1

[47] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. *CoRL*, 2022. 2.2

[48] 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. 6.3

[49] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *Conference on Robot Learning*, pages 3453–3467. PMLR, 2023. (document), 7.3, 8.3, 8.6, 11.3.3

[50] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *Conference on Robot Learning*, pages 3453–3467. PMLR, 2023. 9.5.1

[51] AgileX Robotics. Ranger mini. https://global.agilex.ai/products/ranger-mini, 2023. Omnidirectional mobile robot platform. 9.4.3

[52] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *NIPS*, 2016. 6.3

[53] Michael Ahn, Henry Zhu, Kristian Hartikainen, Hugo Ponte, Abhishek Gupta, Sergey Levine, and Vikash Kumar. ROBEL: RObotics BEnchmarks for Learning with low-cost robots. In *Conference on Robot Learning (CoRL)*, 2019. 7.3

[54] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 2.2, 2.3, 2.3, 7.3, 8.3, 9.2

[55] 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. 6.3

[56] 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. 2.2, 2.3, 2.3, 5.3, 6.3, 8.3

[57] 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. 3.3

[58] 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. 2.3

[59] 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. 4.3

[60] 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. 5.3

[61] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969. IEEE, 2023. 8.3, 9.3

[62] Haruhiko Asada and J-JE Slotine. *Robot analysis and control*. John Wiley & Sons, 1991. 6.3

[63] Maria Attarian, Muhammad Adil Asif, Jingzhou Liu, Ruthrash Hari, Animesh Garg, Igor Gilitschenski, and Jonathan Tompson. Geometry matching for multi-embodiment grasping. *arXiv*, 2023. 11.3.1

[64] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *NeurIPS*, 2020. 2.7.3

[65] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *NeurIPS*, 2020. 4.2, 4.1, 4.4.1, 4.5.3, 4.5.3, B.4, **??**, **??**

[66] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Hierarchical neural dynamic policies. *RSS*, 2021. 4.2, 4.4.1, 4.5.3, **??**, **??**, **??**

[67] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *RSS*, 2022. 2.3, 2.4, 4.3, 6.4.2, 8.3, 11.3.2

[68] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. 2023. 6.2, 6.4.1, 10.2

[69] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Shangchen Han, Fan Zhang, Linguang Zhang, Jade Fountain, Edward Miller, Selen Basol, et al. Hot3d: Hand and object tracking in 3d from egocentric multi-view videos. *arXiv preprint arXiv:2411.19167*, 2024. 10.2

[70] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Fan Zhang,

Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, et al. Introducing hot3d: An egocentric dataset for 3d hand and object tracking. *arXiv preprint arXiv:2406.09598*, 2024. 11.3.2

[71] 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. 2.3, **??**, 5.2, 5.3, 5.7, **??**, 8.3, **??**

[72] 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. 2.2

[73] 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. 4.5.2

[74] Raunaq Bhirangi, Tess Hellebrekers, Carmel Majidi, and Abhinav Gupta. Reskin: versatile, replaceable, lasting tactile skins. *CoRL*, 2021. 6.3

[75] 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. 2.1, 2.3, **??**, 2.5.2

[76] Raunaq Bhirangi, Abigail DeFranco, Jacob Adkins, Carmel Majidi, Abhinav Gupta, Tess Hellebrekers, and Vikash Kumar. All the feels: A dexterous hand with large-area tactile sensing. *IEEE Robotics and Automation Letters*, 2023. 7.3, 8.3

[77] Antonio Bicchi. Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 16(6), 2000. 8.2, 8.6

[78] 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. 3.2

[79] Jonathan Bohren, Radu Bogdan Rusu, E Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the pr2. In *2011 IEEE International Conference on Robotics and Automation*, pages 5568–5575. IEEE, 2011. 9.4.3

[80] 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`. 4.3

[81] 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. 3.5

[82] 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. 3.3

[83] Samarth Brahmbhatt, Cusuh Ham, Charles C Kemp, and James Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8709–8719, 2019. 11.3.2

[84] Samarth Brahmbhatt, Chengcheng Tang, Christopher D Twigg, Charles C Kemp, and James Hays. Contactpose: A dataset of grasps with object contact and hand pose. In *Computer Vision–ECCV 2020: 16th European Conference 2020, Proceedings*, pages 361–378. Springer, 2020. 11.3.2

[85] Paul W Brand. Clinical mechanics of the hand. In *Hand Rehabilitation in Occupational Therapy*, pages 183–184. Routledge, 2012. 1.1

[86] Gerald Brantner and Oussama Khatib. Controlling ocean one: Human–robot collaboration for deep-sea manipulation. *Journal of Field Robotics*, 38(1):28–51, 2021. 9.3

[87] 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. 4.2, 10.2

[88] Tina Bruce. *Learning through play, for babies, toddlers and young children*. Hachette UK, 2012. 1.1, 2.2

[89] 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. 3.4.2, A.6

[90] Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics tools*, 10(3):37–49, 2005. 9.4.1

[91] 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. 1.2, 7.3, 8.3

[92] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3): 261–268, 2017. 9.6.2

[93] 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. 4.5.2, B.3, B.6

[94] 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. 3.3, 3.4.1, 3.4.2, 4.5.2, A.1

[95] 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. 3.3

[96] 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. 3.4.1

[97] Manuel G Catalano, Giorgio Grioli, Edoardo Farnioli, Alessandro Serio, Cristina Piazza, and Antonio Bicchi. Adaptive synergies for the design and control of the pisa/iit softhand. *The International Journal of Robotics Research*, 33(5):768–782, 2014. 7.3, 8.3

[98] Manuel G Catalano, Giorgio Grioli, Edoardo Farnioli, Alessandro Serio, Manuel Bonilla, Manolo Garabini, Cristina Piazza, Marco Gabiccini, and Antonio Bicchi. From soft to adaptive synergies: The pisa/iit softhand. In *Human and Robot Hands*, pages 101–125. Springer, 2016. 7.3, 8.3

[99] M.G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi. Adaptive synergies for the design and control of the pisa/iit softhand. *The International Journal of Robotics Research*, 33(5):768–782, 2014. doi: 10.1177/0278364913518998. URL https://doi.org/10.1177/0278364913518998. 8.4.1

[100] 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. 2.3

[101] 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. 2.4.2

[102] 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. 3.6.1

[103] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, et al. Dexycb: A benchmark for capturing hand grasping of objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9044–9053, 2021. 10.4.1, 11.3.2

[104] 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. 4.3, 11.3.2

[105] 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. 5.2, 5.3

[106] 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. 6.6

[107] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. *Conference on Robot Learning*, 2021. 2.2, 2.3, 8.3

[108] 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. 2.2, 6.3

[109] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023. 7.3, 8.3

[110] 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. 4.2

[111] Zoey Qiuyu Chen, Karl Van Wyk, Yu-Wei Chao, Wei Yang, Arsalan Mousavian, Abhishek Gupta, and Dieter Fox. Learning robust real-world dexterous grasping policies via implicit shape augmentation. *arXiv preprint arXiv:2210.13638*, 2022.

(document), 11.3.1, 11.4

[112] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024. 10.3.3

[113] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. 9.3

[114] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. 10.4.3, E.4

[115] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024. 9.3, 10.2, 10.3.3, 10.4.1

[116] 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. 2.7.4

[117] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan

Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models, 2023. 10.2, 10.3.1, 10.4, 10.6.1

[118] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark. https://github.com/open-mmlab/mmaction2, 2020. 4.5.2

[119] 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. 5.3

[120] Juan Antonio Corrales, Francisco A Candelas, and Fernando Torres. Hybrid tracking of human operators using imu/uwb data fusion by a kalman filter. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 193–200, 2008. 10.3.3

[121] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2020. A.6

[122] 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. 2.7.3, 3.3, 3.4.1, 4.2, 4.3, 4.5.2, 4.5.3, 4.8, 6.3, 6.4.1, 8.5.2, B.3, B.3, C.4

[123] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Epic-kitchens: A large-scale dataset for recognizing, anticipating, and retrieving hand-object interactions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 802–819, 2018. 10.2, 10.3.1

[124] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4125–4141, 2020. 11.3.2

[125] Charles Darwin. On the origin of species. 1859. 1.1

[126] 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. 4.3

[127] 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. 4.3, 6.3

[128] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *CoRL*, 2019. 9.3

[129] 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 Process-*

ing Systems Track on Datasets and Benchmarks, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/3988c7f88ebcb58c6ce932b957b6f332-Paper-round2.pdf. 2.2

[130] 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. 4.4.1, 4.7, **??**, **??**, **??**, **??**, **??**, **??**, **??**, B.4, B.6

[131] Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*. PMLR, 2023. 10.3.1, 10.4, 10.4.3, E.4

[132] Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, pages 1183–1198. PMLR, 2023. 9.6.2, D.8

[133] Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration. *arXiv preprint arXiv:2008.07682*, 2020. 6.2

[134] Roger Davies. Technology versus terrorism. *Jane's International Defence Review*, pages 36–43, 2001. 3.2

[135] Raphael Deimel and Oliver Brock. A novel type of compliant and underactuated robotic hand for dexterous grasping. *The International Journal of Robotics Research*, 35(1-3):161–185, 2016. doi: 10.1177/0278364915592961. URL https://doi.org/10.1177/0278364915592961. 7.3, 8.3

[136] 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. 5.3

[137] 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. 5.2, 7.3, 8.3

[138] 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. B.2

[139] 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. 4.2

[140] Runyu Ding, Yuzhe Qin, Jiyue Zhu, Chengzhe Jia, Shiqi Yang, Ruihan Yang, Xiaojuan

Qi, and Xiaolong Wang. Bunny-visionpro: Bimanual dexterous teleoperation with real-time retargeting using vision pro. 2024. 9.2, 9.3, 9.5.1

[141] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. D.8

[142] 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. 2.3

[143] 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. 5.3

[144] Haritheja Etukuru, Norihito Naka, Zijin Hu, Seungjae Lee, Julian Mehu, Aaron Edsinger, Chris Paxton, Soumith Chintala, Lerrel Pinto, and Nur Muhammad Mahi Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments, 2024. 10.3.3

[145] 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. 4.3, 6.3

[146] Zicong Fan, Omid Taheri, Dimitrios Tzionas, Muhammed Kocabas, Manuel Kaufmann, Michael J Black, and Otmar Hilliges. Arctic: A dataset for dexterous bimanual hand-object manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12943–12954, 2023. 10.4.1, 11.3.2

[147] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020. 11.3.1, 11.6.2

[148] 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. 2.3

[149] 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. 5.3

[150] 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. 2.3, 3.3

[151] 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. 4.5.2

[152] David F Fouhey, Xiaolong Wang, and Abhinav Gupta. In defense of the direct perception of affordances. *arXiv preprint arXiv:1505.01085*, 2015. 6.2

[153] Cinzia Freschi, Vincenzo Ferrari, Franca Melfi, Mauro Ferrari, Franco Mosca, and Alfred Cuschieri. Technical review of the da vinci surgical telemanipulator. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 9(4): 396–406, 2013. 9.3

[154] Authors from UC San Diego and MIT. Open-television: An open-source immersive teleoperation system with stereo visual feedback. *The Robot Report*, 2024. 10.4.1

[155] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024. 9.3

[156] 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. 1.1, 2.2

[157] Sam Goldstein, Dana Princiotta, and Jack A Naglieri. Handbook of intelligence. *Evolutionary theory, historical perspective, and current concepts*, 10:978–1, 2015. (document), 1.1, 8.2

[158] Mohit Goyal, Sahil Modi, Rishabh Goyal, and Saurabh Gupta. Human hands as probes for interactive object understanding. In *CVPR*, 2022. 6.2, 6.4.1

[159] 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. 4.2, 4.3, 6.3

[160] Kristen Grauman, Michael Ryoo, Aljoša Smolić, Minh Vo, and 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 (CVPR)*, pages 11743–11753, 2022. 10.2, 10.3.1, 11.3.2

[161] 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. 4.2, 4.3, 6.3, 6.4.1, C.4

[162] 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. 2.3

[163] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. UMI on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. In *Proceedings of the 2024 Conference on Robot Learning*, 2024. 10.4.3

[164] 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. 6.3

[165] 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. 3.3

[166] 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. 4.3

[167] 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. 5.4.4, 6.3

[168] 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. 3.2, 3.3, 3.4.1, 3.1, 3.5, 3.5, 4.3, 4.5.2, A.3.1

[169] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020. 2.3, 2.3, 2.5.2, 6.4.1, 7.3, 7.7.2, 7.7.2, 8.3, 8.5.1, 9.3, 9.4.1, 10.4.1, 11.3.2, B.3, B.5, B.6

[170] 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. 2.2, 2.7.2, 7.3, 8.3

[171] Saeed Hashemi, Darrin Bentivegna, and William Durfee. Bone-inspired bending soft robot. *Soft Robotics*, 8(4):387–396, 2021. 7.3

[172] 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. 3.3

[173] Jiawei He, Danshi Li, Xinqiang Yu, Zekun Qi, Wenyao Zhang, Jiayi Chen, Zhaoxiang Zhang, Zhizheng Zhang, Li Yi, and He Wang. Dexvlg: Dexterous vision-language-grasp model at scale, 2025. URL https://arxiv.org/abs/2507.02747. 11.3.1

[174] 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. 4.5.1, 4.5.3, 4.5.3, 4.7, 6.6, A.2, A.4.1, B.2

[175] 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. 4.2

[176] 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. 4.7

[177] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 11.3.1

[178] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Zhibo Zhao, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023. 10.2

[179] Binghao Huang, Yixuan Wang, Xinyi Yang, Yiyue Luo, and Yunzhu Li. 3d-vitac: Learning fine-grained manipulation with visuo-tactile sensing. *arXiv preprint arXiv:2410.24091*, 2024. 8.4.4

[180] 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. 2.3, 8.3

[181] 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. 2.2

[182] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023. 11.3.3

[183] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. 9.2

[184] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 2013. 4.4.1

[185] Filip Ilievski, Aaron D Mazzeo, Robert F Shepherd, Xin Chen, and George Mc-Clelland Whitesides. Soft robotics for chemists. *Angewandte Chemie International Editiion*, 2011. 8.4.1

[186] 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. 3.3, 4.3, 6.3

[187] Aadhithya Iyer, Zhuoran Peng, Yinlong Dai, Irmak Guzey, Siddhant Haldar, Soumith Chintala, and Lerrel Pinto. OPEN TEACH: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024. 8.5.1, 10.3.2

[188] S.C. Jacobsen, J.E. Wood, D.F. Knutti, and K.B. Biggers. The utah/m.i.t. dextrous hand: Work in progress. *The International Journal of Robotics Research*, 3(4):21–50, 1984. doi: 10.1177/027836498400300402. URL https://doi.org/10.1177/027836498400300402. 7.3, 8.3

[189] Hanwen Jiang, Shaowei Liu, Jiashun Wang, and Xiaolong Wang. Hand-object contact consistency reasoning for human grasps generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11107–11116, 2021. (document), 11.3.1, 11.4

[190] 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. 6.2, 6.4.2

[191] Lynette A Jones and Susan J Lederman. *Human hand function*. Oxford university press, 2006. 1.1

[192] 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. 4.3, 6.3

[193] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric

Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018. 9.2

[194] 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. 6.3

[195] 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. 2.3, 3.3, 4.3, 6.3

[196] Aditya Kannan, Kenneth Shaw, Shikhar Bahl, Pragna Mannam, and Deepak Pathak. Deft: Dexterous fine-tuning for real-world hand policies. *arXiv preprint arXiv:2310.19797*, 2023. 8.3, 11.3.2

[197] A Kapandji. Clinical test of apposition and counter-apposition of the thumb. *Annales de chirurgie de la main: organe officiel des societes de chirurgie de la main*, 5(1): 67–73, 1986. 7.6.2

[198] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *ICRA*, 2011. 6.3

[199] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. Egomimic: Scaling imitation learning via egocentric video, 2024. 10.3.3, 10.6.1

[200] 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. 6.3

[201] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024. 9.3, 10.2, 10.3.1, 10.6.1

[202] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 10.3.1

[203] 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. 2.3

[204] 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. 1.2, 2.3, 8.4.1

[205] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. C.4

[206] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 11.3.1

[207] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 6.4.2

[208] 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. (document), 6.4.1, 6.6, 11.2, 11.4.2

[209] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Advances in neural information processing systems*, 21, 2008. 6.3

[210] 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. 6.3

[211] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *RSS*, 2021. 2.3, 8.3, 9.2

[212] 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. 4.7, 4.7, **??**

[213] 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. 4.3

[214] 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. 2.3, 7.3, 8.3

[215] 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. 2.2, 2.4, 2.4.1, 2.4.2, 5.2, 5.6, 8.4.2

[216] 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, 2017. doi: 10.1109/TMECH.2016.

2634602. (document), 7.2, 7.3, 8.3

[217] 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. 4.3, 11.3.2

[218] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML*, 2013. 6.3

[219] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. 9.2

[220] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016. 4.3

[221] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *ISER*, 2016. 6.3

[222] Puhao Li, Tengyu Liu, Yuyang Li, Yiran Geng, Yixin Zhu, Yaodong Yang, and Siyuan Huang. Gendexgrasp: Generalizable dexterous grasping. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8068–8074. IEEE, 2023. 11.3.1

[223] 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. 5.3

[224] 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. 3.3, 4.3

[225] Zexiang Li and S. Shankar Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal on Robotics and Automation*, 4(1):32–44, 1988. 8.3

[226] Jacky Liang, Jeffrey Mahler, Michael Laskey, Pusong Li, and Ken Goldberg. Using dvrk teleoperation to facilitate deep learning of automation tasks for an industrial robot. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1–8, 2017. doi: 10.1109/COASE.2017.8256067. 9.3

[227] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022. 11.3.3

[228] 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. 5.3

[229] 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. 1.1, 2.2

[230] 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. 3.3

[231] 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. 6.3

[232] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2024. 10.3.3, 10.4.3

[233] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. *arXiv:2404.16823*, 2024. 9.3, 9.4.1

[234] 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. 6.3

[235] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 10.2

[236] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *arXiv preprint arXiv:2211.08416*, 2022. 9.3

[237] 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. 7.7.1

[238] 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. 3.3

[239] 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. (document), 1.1, 2.4, **??**, **??**, 7.2, 7.7.2

[240] 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. 5.4.4

[241] Minghua Liu, Yinhao Zhu, Hong Cai, Shizhong Han, Zhan Ling, Fatih Porikli, and

Hao Su. Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models, 2023. URL https://arxiv.org/abs/2212.01558. 11.3.1

[242] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *CVPR*, 2022. 6.2, 6.4.1

[243] Yibo Liu, Huaping Xiao, Tianze Hao, Dezhi Pang, Fagang Wang, and Shuhai Liu. Dexterous all-soft hand (dash) with active palm: multi-functional soft hand beyond grasping. *Smart Materials and Structures*, 32(12):125012, 2023. 8.3

[244] 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. 6.3

[245] 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. 6.2, 6.4.1, C.4

[246] 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. 3.3, 4.3, 4.4.2

[247] Tyler Ga Wei Lum, Albert H Li, Preston Culbertson, Krishnan Srinivasan, Aaron D Ames, Mac Schwager, and Jeannette Bohg. Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer. *arXiv preprint arXiv:2410.23701*, 2024. (document), 11.3.1, 11.1, 11.6.1

[248] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017. 6.3

[249] 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. 2.2

[250] Raymond R. Ma, Lael U. Odhner, and Aaron M. Dollar. A modular, open-source 3d printed underactuated hand. In *2013 IEEE International Conference on Robotics and Automation*, pages 2737–2743, 2013. doi: 10.1109/ICRA.2013.6630954. 8.3, 8.4.1

[251] 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. 6.3, 11.3.2

[252] Carmel Majidi. Soft-matter engineering for soft robotics. *Advanced Materials*

Technologies, 4(2):1800477, 2019. 8.4.1

[253] 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. 2.9, 2.7.4, 4.3, 11.5

[254] Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning (CoRL)*, 2021. 2.3, 6.3, 8.3

[255] 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. 4.3

[256] 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. 6.3

[257] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018. 8.5.1

[258] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021. 9.3, 10.3.1

[259] Pragna Mannam, Kenneth Shaw, Dominik Bauer, Jean Oh, Deepak Pathak, and Nancy Pollard. Designing anthropomorphic soft hands through interaction. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2023. doi: 10.1109/Humanoids57100.2023.10375195. 9.2, 9.5.1

[260] Pragna Mannam, Kenneth Shaw, Dominik Bauer, Jean Oh, Deepak Pathak, and Nancy Pollard. Designing anthropomorphic soft hands through interaction. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2023. 7.3, **??**, 7.7.2, 8.3, 8.4.3, **??**

[261] Pragna Mannam, Kenneth Shaw, Dominik Bauer, Jean Oh, Deepak Pathak, and Nancy Pollard. A framework for designing anthropomorphic soft hands through interaction, 2023. 6.2, C.2

[262] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022. 6.3

[263] 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. 6.3

[264] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Alan: Autonomously exploring robotic agents in the real world. *arXiv preprint arXiv:2302.06604*, 2023. 6.3

[265] 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. 2.2

[266] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. 11.3.1

[267] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, June 2023. ISSN 2377-3774. doi: 10.1109/lra.2023.3270034. URL http://dx.doi.org/10.1109/LRA.2023.3270034. E.5

[268] Andrew S Morgan, Kaiyu Hang, Bowen Wen, Kostas Bekris, and Aaron M Dollar. Complex in-hand manipulation via compliance-enabled finger gaiting and multi-modal planning. *IEEE Robotics and Automation Letters*, 7(2):4821–4828, 2022. 2.2

[269] Movella. https://www.movella.com/products/xsens#overview. [Motion capture technology]. 9.3

[270] 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. 6.3

[271] 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. 2.3, 5.3, 8.3

[272] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *ICCV*, 2019. 6.2, 6.4.1

[273] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. 2.3, 8.3

[274] 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. 4.2, 4.5.1, 6.3

[275] Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2022. 11.3.3

[276] 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. 6.3, 10.3.1, 10.4.3

[277] 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. 4.2, 4.3, 4.5.1, 4.5.3, 4.5.3, 4.7, 4.7, 6.4.1, 6.6, B.2, B.4, B.6, C.4

[278] Lael U Odhner, Leif P Jentoft, Mark R Claffee, Nicholas Corson, Yaroslav Tenzer, Raymond R Ma, Martin Buehler, Robert Kohout, Robert D Howe, and Aaron M Dollar. A compliant, underactuated hand for robust manipulation. *The International Journal of Robotics Research*, 33(5):736–752, 2014. 7.4.1, 8.4.1

[279] Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000. 8.3

[280] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. 9.3

[281] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation, 2021. 2.3, 4.3, 8.3

[282] Younghyo Park and Pulkit Agrawal. Using apple vision pro to train and control robots, 2024. URL https://github.com/Improbable-AI/VisionProTeleop. 9.2, 9.3, 9.5.1, D.7

[283] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009. 4.4.1

[284] Austin Patel, Andrew Wang, Ilija Radosavovic, and Jitendra Malik. Learning to imitate object interactions from internet videos. *arXiv preprint arXiv:2211.13225*, 2022. 11.3.2

[285] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 6.3

[286] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Hamer: Hand mesh recovery for the egoexo4d hand pose challenge. 10.3.3

[287] 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. 3.3, 4.3, 4.4.2, A.2

[288] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *CVPR*, 2024. 9.4.1, 9.5.1, 10.3.3

[289] 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. 6.3

[290] 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. 3.3

[291] 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. 6.3

[292] Alexandra Pfister, Alexandre M West, Shaw Bronner, and Jack Adam Noah. Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis. *Journal of medical engineering & technology*, 38(5):274–280, 2014. 10.3.3

[293] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016. 4.2

[294] 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. 4.3

[295] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 10.4.3

[296] 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. 6.3

[297] 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. 4.4.1

[298] Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient learning on point clouds with basis point sets. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4332–4341, 2019. 11.4.5, 11.6.3

[299] F Pugin, P Bucher, and Philippe Morel. History of robotic surgery: from aesop® and zeus® to da vinci®. *Journal of visceral surgery*, 148(5):e3–e8, 2011. 8.5.1

[300] Steffen Puhlmann, Jason Harris, and Oliver Brock. RBO hand 3: A platform for soft dexterous manipulation. *IEEE Transactions on Robotics*, 38(6):3434–3449, December 2022. doi: 10.1109/TRO.2022.3156806. URL https://doi.org/10.1109/TRO.2022.3156806. 5.2, 5.3, 5.7

[301] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 11.3.1

[302] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 11.3.1

[303] 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. 2.7.4

[304] 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. 2.3, 2.4, 4.3, 8.3

[305] 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. 4.3

[306] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dietor Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023. 7.3, 8.3, 9.3

[307] Ri-Zhao Qiu, Shiqi Yang, Xuxin Cheng, Chaitanya Chawla, Jialong Li, Tairan He, Ge Yan, David J. Yoon, Ryan Hoque, Lars Paulsen, Ge Yang, Jian Zhang, Sha Yi, Guanya Shi, and Xiaolong Wang. Humanoid policy ˜ human policy. *arXiv preprint arXiv:2503.13441*, 2025. 10.3.3, 10.4.1, 10.6.1

[308] 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. 6.4.1, C.4

[309] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. *arXiv preprint arXiv:2004.04650*, 2020. 3.2

[310] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and

Trevor Darrell. Real-world robot learning with masked visual pre-training. *CoRL*, 2022. 10.4.3

[311] 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. 2.3, 3.2, 6.3, 8.3

[312] Nathan D. Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies, 2018. URL https://arxiv.org/abs/1801.02854. E.5

[313] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020. 9.3

[314] 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. 6.4.2

[315] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 6.4.2

[316] Rokoko. https://www.rokoko.com/. [Motion capture hardware and software]. 9.3, 9.4.1

[317] 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. 2.3, 3.3, 3.4.1, 4.3, 4.4.2, 4.3, 4.5.2, 6.3, 6.4.1, 8.3, B.3

[318] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022. 9.3

[319] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: Fast monocular 3d hand and body motion capture by regression and integration. *arXiv preprint arXiv:2008.08324*, 2020. 11.3.2

[320] 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. 2.3, 3.3, 3.4.1, 3.4.1, 3.4.2, 4.3, 4.4.2, 4.5.2, 1, 5.2, 6.3, 6.4.1, 6.4.1, 8.5.2, A.1, A.2, A.3.6, A.4.1, A.4.2, A.7.1, B.3, B.6, C.4

[321] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *ICCV Workshops*,

2021. 9.3, 9.4.1, 10.3.3

[322] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011. 10.4.3

[323] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 1999. 10.4.3

[324] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*. Springer, 2006. 4.4.1

[325] 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. 4.3, 11.3.2

[326] 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. 4.5.2, B.3, B.3, B.6

[327] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2.7.4

[328] Robert J Schwarz and C Taylor. The anatomy and mechanics of the human hand. *Artificial limbs*, 2(2):22–35, 1955. 1.1

[329] 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. 4.2

[330] Shadow Robot Company. `https://www.shadowrobot.com/`. [Robotic hand]. 9.3, 9.5.1

[331] Dandan Shan, Jiaqi Geng, Michelle Shu, and David Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, pages 9869–9878, 2020. 3.3, 3.4.1, 4.3, 6.2, 6.3, 6.4.1, A.3.6

[332] 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. 4.3, 11.3.2

[333] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019. 4.3, 11.3.2

[334] Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. Correcting robot plans

with natural language feedback. *arXiv preprint arXiv:2204.05186*, 2022. 11.3.3

[335] Kenneth Shaw and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *in Submission, ICRA*, 2023. 4.5.2, 4.7

[336] Kenneth Shaw and Deepak Pathak. LEAP hand v2: Dexterous, low-cost anthropomorphic hybrid rigid soft hand for robot learning. In *2nd Workshop on Dexterous Manipulation: Design, Perception and Control (RSS)*, 2024. URL https://openreview.net/forum?id=eQomRzRZEP. 9.4

[337] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *CoRL*, 2022. 6.3

[338] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. VideoDex: Learning Dexterity from Internet Videos. In *Conference on Robot Learning (CoRL)*, 2022. 2.3, 2.4, 2.8, 2.6, 2.7.3, 5.3, 7.3, 8.3

[339] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023. (document), 7.2, 7.3, 7.4.3, **??**, 8.2, 8.3, 8.6, **??**, 9.1, 9.2, 9.3, 9.4, 9.2, 9.5.2, 9.2, 9.6, 10.1

[340] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *Robotics: Science and Systems (RSS)*, 2023. 10.5.3

[341] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023. 11.4.1, 11.5

[342] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 654–665. PMLR, 14–18 Dec 2023. 10.3.1

[343] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning*, pages 654–665. PMLR, 2023. 10.2, 10.3.2, 10.4, 11.3.2

[344] Kenneth Shaw, Shikhar Bahl, Aravind Sivakumar, Aditya Kannan, and Deepak Pathak. Learning dexterity from human hand motion in internet videos. *The International Journal of Robotics Research*, 43(4):513–532, 2024. doi: 10.1177/02783649241227559. URL https://doi.org/10.1177/02783649241227559. 9.2

[345] Kenneth Shaw, Yulong Li, Jiahui Yang, Mohan Kumar Srirama, Ray Liu, Haoyu Xiong, Russell Mendonca, and Deepak Pathak. Bimanual dexterity for complex tasks. *arXiv preprint arXiv:2411.13677*, 2024. 8.5.1

[346] Kenneth Shaw, Yulong Li, Jiahui Yang, Mohan Kumar Srirama, Ray Liu, Haoyu Xiong, Russell Mendonca, and Deepak Pathak. Bimanual dexterity for complex tasks. In *8th Annual Conference on Robot Learning*, 2024. 10.3.2, 10.3.3, 10.4.1, 10.5.3, 10.6.3, 11.3.2, E.3

[347] 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. 6.3

[348] Leon Sievers, Johannes Pitz, and Berthold Bäuml. Learning purely tactile in-hand manipulation with a torque-controlled hand. *arXiv preprint arXiv:2204.03698*, 2022. 2.2

[349] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4.7, 4.7

[350] Himanshu Gaurav Singh, Antonio Loquercio, Carmelo Sferrazza, Jane Wu, Haozhi Qi, Pieter Abbeel, and Jitendra Malik. Hand-object interaction pretraining from videos, 2024. URL https://arxiv.org/abs/2409.08273. 10.3.2

[351] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024. 10.3.2

[352] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube, 2022. (document), 2.3, 2.4, 2.8, 2.7.2, 2.7.2, 4.4.2, 4.4.2, 4.5.2, 4.5.2, 7.3, 8.3, 8.6, 8.5.2, 10.4.1, B.3, B.3, B.5, B.6

[353] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: learning a robotic hand imitator by watching humans on youtube. *RSS*, 2022. 2.3, 2.5.2, 5.2, 5.4.4, 6.3, 6.4.1, 7.7.2, 9.3, 9.4.1, 11.3.2

[354] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *RSS*, 2022. 10.3.3

[355] 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. 4.3, 11.3.2

[356] K. Sohn, X. Yan, and H. Lee. Learning Structured Output Representation using Deep Conditional Generative Models. In *NeurIPS*, 2015. 6.4.2

[357] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020. 9.3, 10.3.3

[358] Mohan Kumar Srirama, Sudeep Dasari, Shikhar Bahl, and Abhinav Gupta. HRP: Human affordances for robotic pre-training. In *Proceedings of Robotics: Science*

and Systems, Delft, Netherlands, 2024. 10.3.1, 10.4.3

[359] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. PaliGemma 2: A Family of Versatile VLMs for Transfer. *arXiv preprint arXiv:2412.03555*, 2024. 10.2

[360] Peize Sun, Shoufa Chen, Chenchen Zhu, Fanyi Xiao, Ping Luo, Saining Xie, and Zhicheng Yan. Going denser with open-vocabulary part segmentation, 2023. URL https://arxiv.org/abs/2305.11173. (document), 11.2, 11.4.2

[361] 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. 2.3

[362] 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. 6.3

[363] 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):698–702, 2019. 8.4.4

[364] 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. 2.2

[365] 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. 3.3

[366] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021. B.6

[367] George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh, 2025. URL https://arxiv.org/abs/2408.13679. 11.3.1

[368] Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *Robotics: Science and Systems (RSS)*, 2025. 11.3.2

[369] OM Team, D Ghosh, H Walke, K Pertsch, K Black, O Mees, S Dasari, J Hejna, C Xu, J Luo, et al. Octo: An open-source generalist robot policy. *Proceedings of Robotics: Science and Systems, Delft, Netherlands*, 2023. 10.3.1, 10.4, 10.6.1

[370] Yushuang Tian, Xiaoli Meng, Dapeng Tao, Dongquan Liu, and Chen Feng. Upper

limb motion tracking with the integration of imu and kinect. *Neurocomputing*, 159: 207–218, 2015. 10.3.3

[371] From One Hand to Multiple Hands: Imitation Learning for Dexterous Manipulation from Single-Camera Teleoperation. Qin, yuzhe and su, hao and wang, xiaolong, 2022. 2.3, 6.3

[372] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012. 4.3

[373] Yasunori Toshimitsu, Benedek Forrai, Barnabas Gavin Cangan, Ulrich Steger, Manuel Knecht, Stefan Weirich, and Robert K Katzschmann. Getting the ball rolling: Learning a dexterous policy for a biomimetic tendon-driven hand with rolling contact joints. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–7. IEEE, 2023. 7.3, 8.3, 8.4.1, 8.6

[374] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 10.2

[375] Dylan Turpin, Liquan Wang, Eric Heiden, Yun-Chun Chen, Miles Macklin, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Grasp'd: Differentiable contact-rich grasp synthesis for multi-fingered hands. In *European Conference on Computer Vision*, pages 201–221. Springer, 2022. 11.3.1

[376] Dylan Turpin, Tao Zhong, Shutong Zhang, Guanglei Zhu, Eric Heiden, Miles Macklin, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Fast-grasp'd: Dexterous multi-finger grasp generation through differentiable simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8082–8089. IEEE, 2023. 11.3.1

[377] Dylan Turpin, Tao Zhong, Shutong Zhang, Guanglei Zhu, Eric Heiden, Miles Macklin, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Fast-grasp'd: Dexterous multi-finger grasp generation through differentiable simulation. In *ICRA*, 2023. 11.3.1

[378] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991. A.3.2

[379] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 10.2

[380] Anand Vazhapilli Sureshbabu, Giorgio Metta, and Alberto Parmiggiani. A systematic approach to evaluating and benchmarking robotic hands—the ffp index. *Robotics*, 8 (1):7, 2019. 7.6.2, 7.7.2

[381] Jean Vertut and Philippe Coiffet. Robot technology. vol. 3a. teleoperation and robotics: evolution and development. 1985. 3.2

[382] Vicon. https://www.vicon.com/. [Motion capture technology]. 9.2

[383] 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. 3.3

[384] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023. 9.3, 10.2, 10.3.1

[385] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024. 8.5.1, 9.2, 9.3, 9.4.1, 9.4.3, 10.2, 10.3.3

[386] Chenxi Wang, Hao-Shu Fang, Minghao Gou, Hongjie Fang, Jin Gao, and Cewu Lu. Graspness discovery in clutters for fast and accurate grasp detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15964–15973, 2021. 11.3.1

[387] 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. 2.3, 3.3, 4.3, 6.3

[388] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11359–11366. IEEE, 2023. 11.3.1

[389] Xiaolong Wang, Rohit Girdhar, and Abhinav Gupta. Binge watching: Scaling affordance learning from sitcoms. In *CVPR*, 2017. 6.2

[390] Zehang Weng, Haofei Lu, Danica Kragic, and Jens Lundell. Dexdiffuser: Generating dexterous grasps with diffusion models. *IEEE Robotics and Automation Letters*, 2024. (document), 11.3.1, 11.3, 11.4.5

[391] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 2.7.4

[392] Frank R Wilson. *The hand: How its use shapes the brain, language, and human culture*. Vintage, 1999. (document), 1.1

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

preprint arXiv:2309.13037, 2023. 8.5.1, 8.6.2, 9.2, 9.3, 9.4, 9.4.2, 10.3.2, 10.6.3

[394] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 4.2, 4.5.1, 4.7, 4.7

[395] Haoyu Xiong, Russell Mendonca, Kenneth Shaw, and Deepak Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024. 9.4.3

[396] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023. 11.3.1

[397] 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. 1.2

[398] 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. 6.3

[399] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024. 10.3.2

[400] 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. 6.3, 11.3.2

[401] Yufei Ye, Poorvi Hebbar, Abhinav Gupta, and Shubham Tulsiani. Diffusion-guided reconstruction of everyday hand-object interaction clips. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 19717–19728, 2023. 11.3.2

[402] Tsuneo Yoshikawa. Dynamic manipulability of robot manipulators. *Transactions of the Society of Instrument and Control Engineers*, 21(9):970–975, 1985. 6.3

[403] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985. 2.4.2

[404] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv preprint arXiv:2008.04899*, 2020. 4.3, 11.3.2

[405] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee,

Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023. 11.3.3

[406] 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. 2.3

[407] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017. 6.3

[408] 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. 3.3

[409] Han Zhang, Songbo Hu, Zhecheng Yuan, and Huazhe Xu. Doglove: Dexterous manipulation with a low-cost open-source haptic force feedback glove. *arXiv preprint arXiv:2502.07730*, 2025. 10.3.3

[410] Jialiang Zhang, Haoran Liu, Danshi Li, XinQiang Yu, Haoran Geng, Yufei Ding, Jiayi Chen, and He Wang. Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes. In *8th Annual Conference on Robot Learning*, 2024. (document), 11.3.1, 11.4, 11.6.2

[411] Shutong Zhang, Yi-Ling Qiao, Guanglei Zhu, Eric Heiden, Dylan Turpin, Jingzhou Liu, Ming Lin, Miles Macklin, and Animesh Garg. Handypriors: Physically consistent perception of hand-object interactions with differentiable priors. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024. doi: 10.1109/ICRA57147.2024.10610748. 11.3.2

[412] Yu Zhang and Robert K Katzschmann. Creation of a modular soft robotic fish testing platform. *arXiv preprint arXiv:2201.04098*, 2022. 5.2, 5.2, 5.3

[413] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. (document), 9.2, 9.4.2, 9.4.3, 9.3, 9.6.2, D.8

[414] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. 8.5.1, 9.3, 10.3.2, 10.4.3, E.4

[415] 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. 3.2

[416] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra.

Detecting twenty-thousand classes using image-level supervision. In *ECCV*, 2022. 4.5.2

[417] 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. 6.4.1

[418] Xuance Zhou, Carmel Majidi, and Oliver M O'Reilly. Soft hands: An analysis of some gripping mechanisms in soft robot design. *International Journal of Solids and Structures*, 64:155–165, 2015. 7.4.1, 8.4.1

[419] 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. 5.4.2

[420] 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. 3.3, 3.4.1, 3.6.2, 4.3, 6.3, A.3.6

[421] 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. 10.2, 10.4.1