Watch, Practice, Improve: Towards In-the-wild Manipulation

Shikhar Bahl

CMU-RI-TR-24-04

February 2024



The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee

Deepak Pathak	Carnegie Mellon University (*Co-chair*)
Abhinav Gupta	Carnegie Mellon University (*Co-chair*)
Katerina Fragkiadaki	Carnegie Mellon University
Russ Tedrake	Massachusetts Institute of Technology
Shuran Song	Stanford University

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

© Shikhar Bahl, 2024. All rights reserved.

To my parents, Rajiv Bahl and Tarun Dua

Abstract

The longstanding dream of many roboticists is to see robots perform diverse tasks in diverse environments. To build such a robot that can operate anywhere, many methods train on robotic interaction data. While these approaches have led to significant advances, they rely on heavily engineered setups or high amounts of supervision, neither of which is scalable. How can we move towards training robots that operate autonomously, in the wild? Unlike computer vision and natural language in which a staggering amount of data is available on the internet, robotics faces a chicken-and-egg problem: to train robots to work in diverse scenarios, we need a large amount of robot data from diverse environments but to collect this kind of data, we need robots to be deployed widely - which is feasible only if they are already proficient. How can we break this deadlock?

The proposed solution, and the goal of my thesis, is to use an omnipresent source of rich interaction data – humans. Fortunately, there are plenty of real-world human interaction videos on the internet, which can help bootstrap robot learning by side-stepping the expensive aspects of the data collection-training loop. To this end, we aim to learn manipulation from watching humans perform various tasks. We circumvent the embodiment gap by imitating the effect the human has on the environment, instead of the exact actions. We obtain interaction priors, and subsequently practice directly in the real world to improve. To move beyond explicit human supervision, the second work in the thesis aims to predict robot-centric visual affordances: where to interact and how to move post interaction, directly from offline human video datasets. We show that this model can be seamlessly integrated into any robot learning paradigm. The third part of the thesis focuses on how to build general-purpose policies by leveraging human data. We show that world models are strong mechanisms to share representations across human and robot data coming from many different environments. We use a structured affordance-based action space to train multitask policies and show that this greatly boosts performance. In the fourth work of the thesis, we investigate how to use human data to build actionable representations for control. Our key insight is to move beyond traditional training of visual encoder and use human actions and affordances to improve the model. We find that this approach can improve real-world imitation learning performance for almost any pre-trained model, across multiple challenging tasks. Finally, visual affordances may struggle to capture complex action spaces, especially in high-degree-of-freedom robots such as dexterous hands. Thus, in the final works of the thesis, we explore how to learn more explicit, physically grounded action priors from human videos, mainly in the context of dexterous manipulation.

Acknowledgments

It takes a village. This thesis is the combined effort of the very many people in my life, who have encouraged and supported me over the years. No one has had a greater influence over my growth as a researcher, roboticist, and scientist than my advisors: Deepak Pathak and Abhinav Gupta. I am very grateful for their mentorship, guidance, and support.

I have been extremely fortunate to be able to work with Deepak. The moment I heard he was joining CMU, I knew I wanted to work with him. My decision was only reaffirmed after our first meeting - which I left feeling extremely excited about what was ahead in robotics research. I could ramble for pages about how much Deepak has taught me and how much his mentorship shaped my research. I feel very fortunate to have been his first student and to have spent so much time discussing and trying new research directions with him. I am very grateful for all of the patience and trust that Deepak has had over the years, and for allowing me to both learn and grow as a researcher, as well as explore different areas of AI. From running experiments together during my first year to the hours-long ideas meetings, the late-night discussions in the lab or during conferences, and paper writing sessions - I will miss all these interactions. As an advisor, Deepak has always had a great eye for detail, and yet never manages to lose sight of the larger picture. He has a great knack for both knowing what problem will be important, and what will work at a more granular level. His passion for solving big research problems, and his creativity have been truly inspiring.

Working with Abhinav has truly been a pleasure. I have left every meeting that I have had with him inspired and ready to attack the problem at hand. He has a unique ability to cut through the extraneous details and understand the essence of a problem. He brings so much energy and makes every discussion fun, which I appreciate a lot. Abhinav's creativity has always been very inspiring to me: no matter what problem is presented, he can quickly think of a solution. He has taught me to focus on the right problems and to attack them with creative and clever solutions. I strongly believe that Abhinav is a true visionary, and I will always think long and hard before betting against his ideas.

I would like to thank Katerina Fragkiadaki, Russ Tedrake, and Shuran Song for being part of my thesis committee. I have always admired how passionate Katerina is about embodied AI and I have enjoyed our research discussions: both low and high level. Shuran Song's research has been instrumental in guiding my own research directions. There is no roboticist whose work I admire more than Shuran, and I am excited whenever she releases new work. She can identify important problems, find creative ways to solve them and execute at a very high level. The first time I heard Russ give a talk, I was truly inspired. The way he can blend techniques and a big-picture vision is unique. His knowledge of robotics and ability to dive into mathematical and technical details amazes me.

I want to thank all my amazing labmates and collaborators, without whom none of the work presented in this thesis would have been possible. I would like to thank Russell Mendonca, Kenny Shaw, Murtaza Dalal Sudeep Dasari, Aditya Kannan, Lili Chen, Unnat Jain, Mustafa Mukaddam, Kevin Gmelin, Kehlani Fay, Alexandre Kirchmayer, Mohan Kumar Srirama, Haoyu Xiong, Jiahui Yang, Raunaq Bhirangi, Alexander Li, Ananye Agarwal, Aravind Sivakumar, Mihir Prabhudesai, Ellis Brown, Shagun Uppal, Shivam Duggal, Xuxin Chen, Zipeng Fu, Helen Jiang, Victoria Dean, Kenny Marino Adithya Murali, Yufei Ye, Sam Powers, Homanga Bharadhwaj, Jared Mejia, Justin Wasserman, Senthil Purushwalkam, Jianren Wang, Joyce Hong, Pragna Mannam, Ankit Ramchandani, Vikash Kumar, Simone Parisi, and many more. I would like to especially thank Russell Mendonca and Kenny Shaw. Some of my most (and sometimes least) productive research discussions have been with Russell. I learnt so much from working with Russell over the years. I want to thank him for being a great sounding board, co-author, and friend. He brings a very synergistic energy to mine as a collaborator, and together we are able to break down problems and find solutions very efficiently. I have been able to learn so much about RL and deep learning over the years from him. I hope to continue our fruitful collaboration as time goes on. I will miss all of our insightful discussions, our late-night snack walks, and all of the Zoom games! I want to thank Kenny for being a great co-author and teaching me so much about hardware and robot systems. I have worked closely with Kenny over the years, and many of these works would not have been possible without his input. Whenever I have needed help, Kenny has always been willing to help, and I admire his selflessness, generosity, passion, and knowledge of robotics. I want to thank Aditya, Kehlani, Alexandre, Lili, Kevin, and Haoyu for their patience – I have been very lucky to work with and mentor them!

CMU is a wonderful institution, and I am very grateful to have done my PhD here. I also have been lucky to interact with all of the amazing faculty at CMU, who have helped guide my research vision or who have helped shape my ideas. I would like to especially thank David Held, Oliver Kroemer, Shubham Tulsiani, Chris Atkeson, Howie Choset, Deva Rammanan, and Kris Kitani. I want to thank everybody in RoboOrg, who made my time at RI very fun and exciting. I want to thank Suzanne Muth, Brian Hutchinson, Christine Downey, and everyone in RI who has made the PhD experience hassle-free.

I had a great time interning NVIDIA. I want to thank Stan Birchfield and Jon Tremblay for being great hosts and mentors. I spent a wonderful summer in Seattle, and I learnt a tremendous amount from both Stan and Jon. This experience really helped shape my vision going forward, and even though we were not able to publish together, in the grander scheme of things my time with Stan and Jon was crucial to my research career. I will cherish all of our research discussions and the time we spent working on the Baxter! I had a great time as a visiting researcher at FAIR and would like to thank Aravind Rajeswaran for hosting me. Aravind was a great mentor and helped guide me towards important problems in Embodied AI. I also would like

to thank fellow interns and members of FAIR Pittsburgh, especially Vikash Kumar, Chris Paxton, Sudharshan Suresh, and Jay Vakil, for the great discussions. I want to thank some of my early mentors at Berkeley: Ashvin Nair and Sergey Levine. I am very grateful to them for giving me my start in robot learning, for inspiring me to continue on this path, and for their patience with me as a young, naive researcher.

I want to thank my friends who have always been such a source of optimism and have helped me find a home in a new country. I would like to thank my friends in Pittsburgh: Sudeep Dasari, Jason Zhang, Helen Jiang, Russell Mendonca, Kenny Shaw, Murtaza Dalal, Yufei Ye, Alexander Li, Peter Manohar, Magdalen Dobson, Raunaq Bhirangi, Jianren Wang, Kenny Marino, Adithya Murali, Unnat Jain, Suvansh Sanjeev, Wenxuan Zhou, and Dinesh Reddy, for all of the amazing times in the last few years. I will cherish all of our times together, including the whiskey Wednesdays, the potluck dinners, the karaoke nights, the Redzone Sundays, and thirsty Thursdays. A special thanks to Sudeep, Jason, and Helen for being so kind and for making Pittsburgh feel like a home. Our office chats, dinners, and hangouts were the highlight of my time here! I would also like to thank my friends from both undergrad and high school: Shikhar Sachdev, Aditya Gandhi, Hasan Muhammad, Geetanjali Jain, Robert Hamwey, Anish Haris, Jessica Chillingerian, Tanji Saraf-Chavez, Aleha Valika, Trey Lowerison, Payas Parab, and Arthur Zargaryan, whose support and encouragement were extremely important to me.

I have been lucky to have so many great mentors around me. I want to thank the Ecolint community for getting me started on my scientific journey, especially Mr. Shirley, Mr. Bawden, Mrs. Juniper, Mr. Barr, Mrs. Raghwan, Mr. Ronn, and Mr. Bottorf. Your guidance was crucial to me and I will always appreciate your encouragement.

I want to thank Niharika, who has been an amazing partner throughout the years. I cannot express in words how grateful I am for her support, encouragement and help. She has been a real source of joy and has been my rock throughout my PhD. Niharika has pushed me to go outside my comfort zone, and she is the reason why I have enjoyed my life outside of research. She has been my partner in crime, my partner throughout all the amazing adventures we have had, and just a wonderful partner overall. NJ, I want to thank you for your endless support, and for always being there for me.

I am very grateful to my family for being so supportive. From an early age, I have really been inspired by my grandparents and their love for education. I want to thank my parents, Rajiv Bahl and Tarun Dua for being amazing role models, for supporting my education, and for finding ways to encourage me. It is truly thanks to their sacrifices and their efforts that I have been able to complete this dissertation. Their love for education, science, and public service has always inspired me, and it is impossible to overstate their influence on me. I owe them everything.

To everyone I listed above, and to anyone who I may have forgotten to thank: *None of this would have been possible without you!*

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Contributions . . . 2
- 1.3 List of Publications . . . 5

I Addressing the Human-Robot Morphology Gap **6**

2 Human-to-Robot Imitation in the Wild **7**
- 2.1 Motivation . . . 7
- 2.2 Related Work . . . 9
- 2.3 Human-to-Robot Visual Imitation In the Wild . . . 11
- 2.4 Experimental Setup . . . 18
- 2.5 Results . . . 19
- 2.6 Conclusion . . . 24

II Behavior Priors from Manipulation **25**

3 Affordances from Human Videos as a Versatile Representation for Robotics **26**
- 3.1 Motivation . . . 26
- 3.2 Related Work . . . 28
- 3.3 Affordances from Human Videos (VRB) . . . 29
- 3.4 Experimental Setup and Results . . . 35
- 3.5 Conclusion . . . 39

4 Learning Dexterous Affordances **40**
- 4.1 Motivation . . . 40
- 4.2 Related Work . . . 41
- 4.3 Fine-Tuning Affordance for Dexterity . . . 43
- 4.4 Experiment Setup . . . 47
- 4.5 Results . . . 48
- 4.6 Conclusion . . . 50

III Learning General Purpose Policies from Human Videos 52

5 Strucured World Models from Human Videos 53
5.1 Motivation . . . 53
5.2 Related Work . . . 55
5.3 Background and Preliminaries . . . 57
5.4 World Models from Human Videos . . . 58
5.5 Experimental Setup . . . 62
5.6 Results . . . 64
5.7 Conclusion . . . 67

6 Actionable Representations Via Affordances 68
6.1 Motivation . . . 68
6.2 Related Work . . . 70
6.3 Preliminaries . . . 72
6.4 Introducing HRP . . . 73
6.5 Experimental Details . . . 75
6.6 Results . . . 77
6.7 Conclusion . . . 81

7 Learning Dexterity from Internet Videos 82
7.1 Motivation . . . 82
7.2 Related Work . . . 83
7.3 Background . . . 84
7.4 Learning Dexterity from Human Videos . . . 85
7.5 Experimental Setup . . . 89
7.6 Results . . . 91
7.7 Conclusion . . . 93

IV Action Priors for Robotics 95

8 Neural Dynamic Policies 96
8.1 Motivation . . . 96
8.2 Related Work . . . 98
8.3 Modeling Trajectories with Dynamical Systems . . . 101
8.4 Neural Dynamic Policies (NDPs) . . . 102
8.5 Hierarchical Neural Dynamical Policies (H-NDPs) . . . 104
8.6 Experimental and Implementation Setup . . . 109
8.7 Results: H-NDPs for Imitation Learning . . . 110
8.8 Results: H-NDPs for Reinforcement Learning . . . 115

9 PlayFusion: Language Conditioned Skills **117**
9.1 Motivation 117
9.2 Related Work 119
9.3 Background 120
9.4 PlayFusion: Discrete Diffusion for Language-Annotated Play 121
9.5 Experiments 124
9.6 Conclusion 127

10 Conclusions, Discussion, and Future Work **129**

List of Figures

2.1 We present WHIRL [23], an efficient real-world algorithm for one-shot visual imitation in the wild. WHIRL is able to directly learn from unstructured human videos and generalize to new tasks as well. Videos and webpage at: https://human2robot.github.io 7

2.2 Our method (WHIRL [23]) provides an efficient way to learn from human videos. We have three core components: we first "watch" and obtain human priors such as hand movement and object interactions. We "repeat" these priors by interacting in the real world, by both trying to achieve task success and explore around the prior. We "improve" our task policy by leveraging our agent-agnostic objective function which aligns human and robot videos. 8

2.3 We perform various experiments in the wild. We select a subsample of tasks, as shown above, to perform a thorough study of our WHIRL as well as baselines and ablations. These tasks are: drawer, door and dishwasher opening and closing. 12

2.4 We show the different components of the human prior. First we extract the position of the hand and possible object interactions (a). This indicates a possible area of interaction (b) and direction for moving the robot hand (c). We project these to the robot's action space and execute the trajectory. . . . 13

2.5 We show a sample of our agent agnostic representation, using the video inpainting method by Lee et al. [202]. An image of a human performing the task is shown in (c) and (d) shows the human inpainted from the image. Similarly for the robot, (a) and (b) show the original and inpainted images. We train segmentation models [422, 331] to obtain human and robot masks. 16

2.6 We present results of our thorough investigation of WHIRL on various kitchen tasks such as drawer (a), door (b) and dishwasher (c) opening and closing, a task involving picking and placing different objects into shelves (d). We test multi-task policies (e) trained on a subset of the tasks and generalization between tasks (f). We report training and testing success rates (out of 1) for 3 iterations of training. 19

2.7 We ablate different aspects of WHIRL. We analyze the need for our task-agnostic exploration policy, as well as our agent-agnostic representations. (a) is the drawer task, (b) is the door task. In (c) we analyze on the drawer task how our cost functions compare for different levels of success obtained by a trajectory. 22

3.1 We leverage human videos to learn visual affordances that can be deployed on multiple real robot, in the wild, spanning several tasks and learning paradigms. 26
3.2 **VRB Overview**. First, we learn an actionable representation of visual affordances from human videos: the model predicts contact points and trajectory waypoints with supervision from future frames. For robot deployment, we query the affordance model and convert its outputs to 3D actions to execute. 27
3.3 **Robot Learning Paradigms** : (a) Offline Data Collection – Used to investigate the quality of the collected data. (b) Exploration – The robot needs to use intrinsic rewards to improve (c) Goal-Conditioned Learning – A desired task is specified via a goal image, used to provide reward. (d) Action Spaces – Reduced action spaces are easier to search and allow for discrete control. 30
3.4 Qualitative affordance model outputs for `VRB`, `HOI` [216], `Hotspots` [123] and `HAP` [123], showing the predicted contact point region, and post-grasp trajectory (green arrow for `VRB`, red for `HOI` [216]). We can see that `VRB` produces the most meaningful affordances. 33
3.5 **Exploration**: Coincidental success of VRB in comparison to random exploration or the exploration based on HAP [123]. 35
3.6 **Goal-conditioned Learning**: Success rate for reaching goal configuration for six different tasks. Sampling via VRB leads to faster learning and better final performance. 38
3.7 **Action Space**: Success using DQN with the discretized action space, for reaching a specified goal image. 38
3.8 **Feature space distance**: Distance to goal in feature space for VRB decreases monotonically for door opening. 39
3.9 Failure mode analysis 39

4.1 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. 40
4.2 **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. 42
4.3 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. 44

4.4 **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. 46

4.5 Qualitative results showing the finetuning procedure for DEFT. The model learns to hold the spatula and flip the bagel after 30 CEM iterations. 47

4.6 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. 48

4.7 We evaluate on three difficult manipulation tasks. 49

5.1 We present SWIM, an approach for learning manipulation tasks in the real world with only a handful of trajectories. 53

5.2 Overview of SWIM. We first pre-train the world model on a large set of human videos. We finetune this on many robot tasks, in an unsupervised manner, and deploy at test-time in the real world to achieve a given goal. Videos can be found at https://human-world-model.github.io 54

5.3 World Model Training: Images and actions are encoded into a learned feature space that has temporal structure, following the approach from [140] . . . 56

5.4 We evaluate SWIM on six different real-world manipulation tasks on two different robot systems (shown on the left). On the right, we show a sample of the visual affordances from the visual affordance model $\mathcal{G}_\psi$. 57

5.5 a) World Model pre-training reconstructions on Epic-Kitchens dataset [73]. b) Model imagination rollouts for high-reward trajectories. We can see that SWIM can imagine plausible and successful trajectories, for both human and robot data. The first image (highlighted in red) is the original observation by the robot. 59

5.6 Comparison of SWIM and `MBRL-Affordance` for both the single task and jointly trained model. We see a large drop in success when removing pre-training on human videos, especially when dealing with diverse robot tasks. 63

5.7 Continuous improvement (a-b): We see that SWIM continues to improve, achieving high success. (c) Ablating the need for external feature space goal distance at test time. 64

5.8 Image reconstruction using world model features in early training stages for SWIM and `MBRL-Affordance` (which has no pre-training), showing that SWIM can effectively transfer representations from human videos. Note that for our experiments we use models trained to convergence. 65

6.1 We present **HRP**, a method that mines affordances from human videos and uses them to improve self-supervised visual encoders. 68

6.2 HRP fine-tunes a pre-trained encoder to predict three classes of human affordance labels via L2 regression. Specifically, the network must predict future contact points, human hand poses, and the target object given an input frame from the video stream. These affordance labels are mined autonomously from a human video dataset [126] using off-the-shelf vision detectors [362]. Representations produced by HRP are then fine-tuned to solve downstream manipulation tasks via behavior cloning. 69

6.3 We show a summary of our affordances. From human videos, we extract contact heatmaps, hand poses, and active object bounding boxes. 71

6.4 Our experiments consider 5 unique manipulation tasks, ranging from classic block-stacking to a multi-stage toasting scenario. These tasks are implemented on 3 unique robot setups, including a high Degree-of-Freedom dexterous hand (right). The 3 camera views shown – front, ego, and side views (for xArm/dexterous hand) – are the same views ingested by the policy during test-time. Note that 3 of the tasks consider 2 unique camera views in order to test for robustness! 74

6.5 We apply HRP to 6 different baseline representations and plot performance on average across *toasting*, *pouring*, and *stacking* tasks, across two distinct views. We find that HRP representations consistently and substantially outperform SOTA baselines. 76

6.6 (a) applies an ablated HRP method (full fine-tuning) to the 6 baseline representations and compares their average performance v.s. standard HRP representations on the *toasting*, *pouring*, and *stacking* tasks (front cam). We find that LayerNorm only fine-tuning is almost always superior. In (b), we drop each of the 3 losses in HRP, and compare the ablated method's average performance against full HRP representations. 79

7.1 We re-target human videos as an action prior, use pretrainined embeddings as a visual prior, and use Neural Dynamical Policies (NDPs) [25] as a physical prior to complete many different tasks on a robotic hand. 83

7.2 The collection of train objects (left) and test objects (right) used for experimentation. 85

7.3 To use internet videos as pseudo-robot experience, we re-target human hand detections from the 3D MANO model [336] to 16 DoF robotic hand (LEAP) embodiment and we retarget the wrist from the moving camera to the xArm6 [3] embodiment. Videos at https://video-dex.github.io 87

7.4 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. 89

7.5 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. 90
7.6 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. 90

8.1 We present H-NDPs, an efficient real-world robot learning algorithm able to generalize across a high amount of diversity. 96
8.2 We train local Neural Dynamic Policies (NDPs) $\pi_l^{(i)}$ on each region i of the task space, from state observations. A global NDP π_g (usually taking in image input I_t) learns to imitate the local experts. We use the global NDP to retrain local NDPs which keeps the NDPs from diverging. These local-to-gloabl interactions happen in an iterative manner. NDPs make a good candidate for capturing such local-to-global interactions due to their shared structure and the fact that they operate over a smooth trajectory space. 101
8.3 Visualizations of the original demonstrations and the trained local NDP on selected joints, for the real-world scooping task. The x-axis is the timestep and y is the joint value. Each curve representations a different demonstration. We can see that local NDPs can efficiently capture the desired motion in a smooth manner. 108
8.4 Sample trajectories for Scooping (top) and Pouring (bottom) tasks, on the Franka Panda robot 109
8.5 Reinforcement Learning environments in MuJoCo [403]. 110
8.6 Images showing the writing task setup. (a) shows the robot setup we used, a 7 DoF robot with a marker placed inside its end-effector, controlled via joint angles. (b) shows the input image (unseen at training time). (c) shows the output of our method, (d) shows the output of the local controller and (e) shows the final output of GPS [203]. We can see that our method produces a smooth and correct-looking 4. 111
8.7 Success rate for the three real-world tasks across iterations. Note that more iterations of the H-NDP method in fact does help in learning, both in the train and test (held-out/unseen) scenarios. 113
8.8 Success rate for the three simulated RL tasks: throwing, catching and picking. Note that all these tasks are stochastic. Our method (red) outperforms all the baselines. 114

9.1 Across multiple real-world and simulated robotic settings, we show that our model can extract semantically meaningful skills from language-annotated play data. Such data is highly multimodal and offers no optimality guarantees. Video results of PlayFusion are available at https://play-fusion.github.io. 118

9.2 Overview of how PlayFusion extracts useful skills from language-annotated play by leveraging discrete bottlenecks in both the language embedding and diffusion model U-Net. We generate robot trajectories via an iterative denoising process conditioned on language and current state. 120
9.3 Simulated (top row) and real-world (bottom row) environments used for our evaluations. In each real-world setup, the robot is tasked with picking up one of the objects (e.g., plate, cup, carrot, bread, corn) and relocating it to a specified location (e.g., drying rack, plate, toaster, grill, pot). 122
9.4 Visualization of the codebook embeddings for various real-world skills. . . 126

List of Tables

2.1 We present a set of evaluations on two real world tasks: drawer and door. The task is to imitate the third person demonstration of the human. The results presented our averaged over 30 trials. 21

3.1 **Imitation Learning**: Success rate for k-NN and Behavior Cloning on collected offline data using various affordance models. We find that VRB vastly outperforms prior approaches, indicating better quality of data. 36

3.2 Imitation with VRB vs. R3M [271] representation. 37

4.1 Parameters that are fine-tuned in the real world. 43

4.2 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. 48

4.3 Ablations for (1) reward function type, (2) model architecture, and (3) parameter estimation. 50

5.1 Success rates of SWIM and baselines on six different manipulation tasks, over 25 trials. 60

6.1 This table compares 3 representations trained w/ `HRP` against the teacher ResNet [362] that generated our human affordance dataset (see Sec. 6.3.2). We find that the ResNet teacher under-performs even the worst `HRP` representation (fine-tuned from `CLIP`), *even after excluding the stacking task, which it failed on.* 78

6.2 We present results of `Ego4D + HRP` and `ImageNet + HRP`, as well as the respective baselines on the x-Arm (Pot on Stove) and a dexterous hand task (Lift Cup). We see that `HRP` can even boost performance in multiple morphologies, including a high-degree of freedom dexterous hand [368]. . . . 79

6.3 This table compares `Ego4D + HRP` and `ImageNet + HRP` representations against their respective baselines on a *stacking w/ distractors*. We find that HRP helps in these settings, especially for `Ego4D`. 81

7.1 We present the results of train objects and test objects for Videodex and baselines as described above. 91
7.2 1-DOF xArm gripper performance using Videodex [3]. 92
7.3 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. 92
7.4 We present the results of the ablations discussed in Section 7.5. These are all performed on the `place` task. 93

8.1 Final results on the three real world tasks. We average the test success rate normalized to $[0-1]$ over 10 trials on held-out testing images/locations. We compare against vanilla NDP [25], vanilla NN imitation, and we replace NDPs in our method with vanilla neural networks (a similar method to GPS [203]). We can see that our method outperforms all the baselines substantially. 112

9.1 Success rates for PlayFusion and the baselines on simulation and real-world settings. PlayFusion consistently outperforms all of the baselines. 125
9.2 Average sequence length on Long Horizon CALVIN and success rate for the n-th instructions. 125
9.3 Effect of discrete bottlenecks. 126
9.4 Effects of conditioning, language model, and loss weights. 127

Chapter 1

Introduction

1.1 Motivation

General-purpose agents are the biggest goal of robot learning research. How can we build such AI agents that can solve hundreds of tasks in diverse complex visual environments, such as rearranging objects or completing chores in a kitchen? In recent years, there have been significant advances in various areas of building autonomous agents: manipulation tasks such as grasping to pushing and pick-n-place tasks [12, 234, 171], from manipulating a Rubik's cube [9] to opening cabinet doors or makeshift doors [359, 309], or locomotion tasks such as humanoid control to quadrupeds that walk run or jump [190, 201]. While there has been substantial progress, most experiments in this area have still been restricted to simulation [236, 35, 403] or table-top experiments [207, 85] in lab settings. This invokes the pressing question: why hasn't this progress transferred to manipulation in the real world and why do we still see most experiments in lab setups or simulations? Although there have been efforts to perform grasping in home setups [130, 385], general manipulation is still studied in lab-like settings. Through this proposal, we delve into how could robot learning move from lab experiments to in-the-wild setups across complex environments.

The biggest bottleneck for learning manipulation *in the wild* is the lack of scalable and safe frameworks. Traditionally, designing a controller or policy for manipulation tasks requires learning via reinforcement (RL) [393, 300, 395, 203, 355, 136], which can be data-hungry and unsafe, especially in the real world. While RL has had success in simulated tasks, real-world tasks do not have structured rewards, thus making the problem that of sparse search. A popular alternative is to use imitation learning (IL) based approaches [348, 326, 278], but common IL approaches rely on lots of kinesthetic or teleoperated demonstrations per task. However, this data can be expensive to obtain in the real world and may not be generalizable to new settings or different robots. There have been attempts at one-shot imitation at inference, but these methods require thousands of demonstrations or interactions during

training [96, 106, 290]. If the robots were to work in realistic scenarios in the wild, we argue that they must be trained in the real world itself but how to do so in a safe and efficient manner remains an open question.

Unlike computer vision and natural language where a staggering amount of data is available on the internet [453, 316, 323, 93, 38], robotics faces a chicken-and-egg problem: to train robots to work in diverse scenarios, we need a large amount of robot data from diverse environments but to collect such kind of data, we need robots to be deployed widely which is feasible only if they already work at the first place. How do we get around this deadlock? Our proposed solution to this problem is to learn by watching humans act either in the real world or through passive human interaction videos which are abundantly available on the internet. This data can help bootstrap robot learning by side-stepping the expensive aspects of the data collection-training loop.

However, there are major challenges when aiming to leverage human data [381, 356, 367, 364, 50]: 1) there is a large embodiment gap, 2) it is unclear how to represent action information of humans in a useful manner, 3) there is no concrete way to obtain actions from human videos and train generalizable policies, and 4) it is difficult to incorporate knowledge of the physical world into the policy learning setup. This thesis presents works attempting to address the above challenges, in order to enable in-the-wild manipulation.

1.2 Contributions

1.2.1 Bridging the Embodiment Gap

In order to overcome the fact that humans and robots have different morphologies, in Chapter 2, we propose that the robot mimic the *effect* the human had on the environment and not the human directly. We leverage advances in perception to obtain mid-level priors from a human video: hand-object interaction and hand motion information. We project these to the robot's action space and execute the primitives. Due to possible inaccuracies in the pipeline, the robot must practice. We leverage a novel sampling-based optimization framework to improve our policy in an iterative manner. In order to compare robot and human videos, we introduce an *agent-agnostic* alignment objective function, which relies on in-painting the agent out of the scene. To not be restricted by the prior, we employ a novel task-agnostic exploration policy which allows the agent to sample interesting actions leading to more efficient discovery. We demonstrate our framework on 20 different tasks and environments. We show in-the-wild generalization and success in various real-world settings and day-to-day tasks. This approach is able to scale to many tasks in the wild but is largely limited to the human being in the same scene as the robot. How can the robot infer *what to do when the human is not present*?

1.2.2 Behavior Priors from Humans

In order to move beyond explicit human supervision, it is crucial to obtain action information from offline human videos. When placed in new settings, humans already have a good understanding of how most objects should be manipulated. How can we represent this innate concept and how can we learn it from human video data? In Chapter 3, inspired by the fact humans heavily rely on visual *affordances* of objects to efficiently perform day-to-day tasks [118, 119], we propose formulating **robot-centric visual affordances**. We argue that contact points and post-contact are excellent robot-centric representations of visual affordances, since these are omnipresent in manipulation tasks, and can be easily extracted from human videos. We reformulate human videos to focus on frames without humans for predicting affordances. We utilize off-the-shelf tools for estimating ego motion, human pose, and hand-object interaction. In Chapter 3, we show how to seamlessly integrate this affordance model with different kinds of robot learning paradigms – imitation and offline learning, exploration, visual goal-reaching, and using affordances as a parameterization for action spaces. We perform real-world experiments on 10 *in-the-wild* tasks and 2 robot hardware platforms. Our affordance model outperforms state-of-the-art affordance learning approaches [123, 216, 265], and enables high-performance robot learning in the wild without requiring any simulation.

In Chapter 4, we scale our affordance model to complex dexterous tasks. We propose a novel approach, DEFT (DExterous Fine-Tuning 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. While affordances can help learn general (high-level) action strategies, is it possible to even go further and learn exact control from watching humans, and obtain more explicit supervision?

1.2.3 Learning Generalizable Policies

Structured World Models from Human Videos An efficient way for us to operationalize affordances, human data, and robot is to build a *model of the world*. This can allow us to train policies that can operate in many different environments and do multiple tasks. The key question is what actions will we model? Low-level predictive models do not transfer well between humans and robots, or to new tasks. In Chapter 5, we thus leverage knowledge from our *visual affordance model* and create a high-level, human-centric action space, trained on human videos, and finetuned with real-world practice. World models can work in an unsupervised manner, across many tasks, allowing for large-scale generalization. We find that our pre-trained world model achieves higher success ($\sim$ 2X) than prior approaches while being very

sample efficient, requiring less than 30 minutes of real-world interaction data on multiple robotic platforms and tasks.

Actionable Pre-trained Representations We are also able to scale these ideas to more flexible settings, as our structured action space may not be useful in every setup. In Chapter 6, we show that affordances can be used to improve existing *pretrained representations* for control. We present a simple framework for pre-training representations on hand, object, and contact affordances that highlight relevant objects in images and how to interact with them. Our approach can efficiently fine-tune *any* existing representation, and results in models that are more actionable across the board. We experimentally demonstrate (using 3000+ robot trials) that this affordance pre-training scheme boosts performance by a minimum of 15% on 5 real-world tasks, which consider three diverse robot morphologies (including a dexterous hand). These representations improve performance across 3 different camera views. Quantitatively, we find that our approach leads to higher levels of generalization in out-of-distribution settings.

Pre-training Policies with Human Videos Instead of only training representations with affordances, can we also extend these ideas to models that predict explicit actions that can be deployed on the robot? Learning more explicit action priors can lead to more grounded, safer, and more efficient policies. For continuous control, the action space is exponential in the number of actions and timesteps, and even more difficult for high degree-of-freedom robots, such as dexterous hands. Thus learning actions from humans can greatly empower robotic systems to learn much faster. In Chapter 7, we aim to build a system that can guide robot motions using human videos. This requires understanding the scene in 3D, figuring out human intent, and transferring from human to robot embodiment. First, we propose using 3D human estimation, which works decently well in general human videos. We can leverage such approaches to gather 3D understanding. Second, there have been large-scale datasets that break down the human intent via crowdsourcing labels [73, 127], which can be used to train the policy. Finally, to handle the embodiment transfer, we use human hand to robot hand retargeting as an energy function to obtain explicit action priors from human video data. We use actions from the respective videos to train a behavior cloning policy and finetune it with a few real-world demonstrations. We deploy these in complex real-world dexterous manipulation tasks.

1.2.4 Action Priors for Robot Learning

Neural Dynamical Policies Inferring physical knowledge from passive human data is very challenging, as we only have access to a single sense, vision. Nevertheless, it is possible to design learning algorithms that are aware of the physical world. Thus, in Chapter 8, we propose embedding the structure of dynamical systems [349] into deep neural network-based policies so that the agent can directly learn in the space of physically plausible trajectory distributions. Our key insight is to reparameterize the policy network with nonlinear differential equations corresponding to a dynamical

system and train it end-to-end over time. Our approach is able to perform dynamic tasks such as throwing or catching (Chapter 8). We are also able to scale to dynamic, real-world tasks with raw high-dimensional images as inputs, with large variations in object positions and goal locations.

Learning Skills from Play In Chapter 9, we take the idea of structured actions further into the field of skill learning. Learning from unstructured and uncurated data has become the dominant paradigm for generative approaches in language or vision. Such unstructured and unguided behavior data, commonly known as play, is also easier to collect in robotics but much more difficult to learn from due to its inherently multimodal, noisy, and suboptimal nature. We leverage advances in diffusion models to learn a multi-task diffusion model to extract robotic skills from play data. Using a conditional denoising diffusion process in the space of states and actions, we can gracefully handle the complexity and multimodality of play data and generate diverse and interesting robot behaviors. To make diffusion models more useful for skill learning, we encourage robotic agents to acquire a vocabulary of skills by introducing discrete bottlenecks into the conditional behavior generation process. In our experiments, we demonstrate the effectiveness of our approach across a wide variety of environments in both simulation and the real world.

1.3 List of Publications

List of publications included in the thesis:

Chapter 2: Human-to-Robot Imitation in the Wild. RSS 2022 [23].

Chapter 3: Affordances from Human Videos as a Versatile Representation for Robotics. CVPR 2023 [24].

Chapter 4: DEFT: Dexterous Fine-tuning for Real-world Hand Policies. CoRL 2023 [173].

Chapter 5: Structured World Models from Human Videos. RSS 2023 [250].

Chapter 6: Human Affordances for Robotic Pre-Training. In Submission [15].

Chapter 7: Learning Dexterity from Internet Videos. CoRL 2022 [369].

Chapter 8: Neural Dynamic Policies for End-to-end Sensorimotor Learning. NeurIPS 2020 [25], and Hierarchical Neural Dynamic Policies. RSS 2021 [22].

Chapter 9: Skill Acquisition via Diffusion from Language-annotated Play. CoRL 2023 [53].

Part I

Addressing the Human-Robot Morphology Gap

Chapter 2

Human-to-Robot Imitation in the Wild

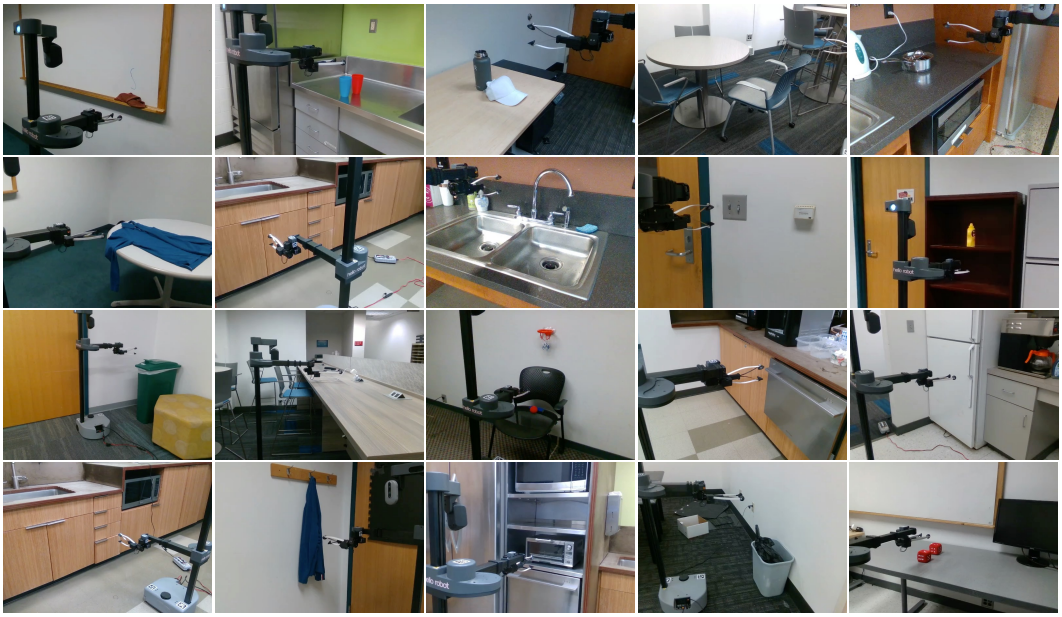


Figure 2.1: We present WHIRL [23], an efficient real-world algorithm for one-shot visual imitation in the wild. WHIRL is able to directly learn from unstructured human videos and generalize to new tasks as well. Videos and webpage at: https://human2robot.github.io

2.1 Motivation

In recent years, there has been significant advances in robot manipulation: from grasping to pushing and pick/place tasks [12, 234, 171]; from manipulating a rubik's cube [9] to opening cabinet doors or makeshift doors [359, 309]. While there has been substantial progress, most experiments in this area have still been restricted

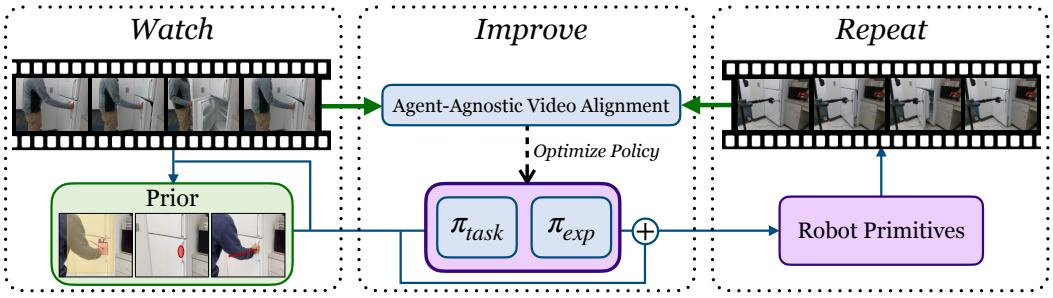


Figure 2.2: Our method (WHIRL [23]) provides an efficient way to learn from human videos. We have three core components: we first "watch" and obtain human priors such as hand movement and object interactions. We "repeat" these priors by interacting in the real world, by both trying to achieve task success and explore around the prior. We "improve" our task policy by leveraging our agent-agnostic objective function which aligns human and robot videos.

to simulation [236, 35, 403] or table-top experiments in the lab [207, 85]. We ask a basic question as to why hasn't this progress transferred to manipulation in the real-world setup and why do we still see most experiments in lab setups or simulations? Although there have been efforts to perform grasping in home setups [130, 385], general manipulation is still studied in either simulation or lab-like settings. This work delves the question of how we could move from from lab experiments to more in-the-wild setups.

We believe the biggest bottleneck for learning manipulation *in the wild* is the lack of scalable and safe frameworks. Traditionally, designing a controller or policy for manipulation tasks requires learning via reinforcement (RL), which can be data-hungry and unsafe especially in the real world. While RL has had success in simulated tasks, real world tasks do not have structured rewards, thus making the problem that of sparse search. A popular alternative is to use imitation learning (IL) based approaches, but common IL approaches rely on lots of kinesthetic or teleoperated demonstrations per task. However, this data can be expensive to obtain in the real world and may not be generalizable to new settings or different robots. There have been attempts at one shot imitation at inference, but these methods requires thousands of demonstrations or interactions during training [96, 106, 290]. To move towards general robot manipulation out of the lab and tabletop settings, we believe visually imitating humans provide a safe and scalable alternative. Rather than asking humans to teleoperate, robot should observe humans to learn as they interact in the world. Humans provide a rich source of data as they often act in interesting and near optimal ways given a task or an environment. In the visual imitation framework, the agent observes other agents perform action without access to actions (just the pixels). This data is then used to guide their own exploration and learning. However, there are several challenges in making visual imitation learning work: first, there is the issue of embodiment mismatch (robots and humans are different agents

and have different bodies). Second, there is no access to actions from humans, which have to be inferred. Third, there is no access to task information including rewards beyond pixels. Current approaches use end-to-end learning [367, 436, 381] which requires a lot of samples during training and are hence restricted to lab/simulation settings.

In this chapter, we propose to revive this visual human-imitation framework to move robot manipulation out of the lab and into the wild. Instead of learning end-to-end from scratch, we propose leveraging advances in computer vision and computational photography to (a) infer a trajectory and interaction information from the human, thus obtaining a prior; (b) learning an improvement policy via interactions in the real world; and (c) bridging the embodiment gap between human demonstrations and robot videos. But above all, we only use imitation data as **priors** for our policy. Naively using priors will not result in success due to a host of issues, e.g. varying morphologies or inaccuracies in detections. It is crucial to *interact* with the real world to learn generalizable manipulation. We introduce a sampling based optimization framework, similar to the Cross Entropy Method (CEM), in order to iteratively improve the interaction policy. To make WHIRL operate without supervision, we introduce an *agent-agnostic* alignment objective function for the described optimization approach. In order to not be too restricted by the prior, we employ a novel task-agnostic exploration policy which allows the agent to sample new and interesting actions. This all leads to an efficient framework for manipulation tasks in real world.

We demonstrate our framework on 20 different tasks in 3 different environments. We show one-shot, in the wild generalization and success in various real world settings, including manipulation tasks such as opening and closing doors or fridges, putting objects in shelves, folding shirts, cleaning white boards, opening taps and a variety of other tasks. We analyze our approach thoroughly in terms of task success, generalization, and performance compared to state-of-the-art baselines. To the best of our knowledge, this is the first effort that takes robot manipulation out of the lab and into the real world at this scale.

2.2 Related Work

2.2.1 Detecting Humans

The field of computer vision has studied the problem of detecting humans in a wide variety of approaches. Most such applications are contained in the domain of graphics, but many have applications in real world robotics as well. There are many possible uses of such, for example modeling human bodies, detecting poses, inferring dynamics or understanding interactions between humans and the world. From a modeling perspective, works such as MANO [336] and SMPL [224] have proposed analytical models of human hands and bodies respectively. Hand and

body pose estimation [412, 172, 337] can be useful in the context of robotics, as it can allow for spatially grounding a demonstration, which is something that we leverage in our approach. Estimation and detection of humans, while useful, does not help in understanding *what* the human is doing. For this, large annotated video datasets can help detect and infer human actions, such as the Something-Something [124], YouCook [80], ActivityNet datasets [103] or the 100 Days of Hands [362] (100DOH) dataset. 100DOH [362] is particularly useful as it contains object level interaction annotations. WHIRL aims to remain as general as possible in terms of the human prior used, using only object interaction data. We employ both the models from [337] and the hand-object detector from Shan et al. [362] for estimating hand position and interaction information. It is possible to combine WHIRL with stronger priors for human hands, for example, building a knowledge graph of objects and functional grasps [259] or using heat sensing [34] to understand interactions.

2.2.2 Imitation and Reinforcement Learning from Videos

Learning From Human Videos A large field of robot learning (Learning from Demonstrations: LfD) is focused on learning from expert demonstrators [287, 326, 308, 339]. However, most of the work in this area tackles the problem of learning from demonstrations that humans provide directly to the robot via kinesthetic teaching or teleoperation. This is an expensive way to gather data for teaching robots. On the other hands, videos of humans perfoming daily activities are widely available on the internet and can provide good semantic supervision for robotics tasks. However, extracting the right knowledge, for example aligning human videos with robot videos, is challenging. One solution is to learn a direct correspondence. The use of paired human and robot data [367, 366, 222] is a common approach in this line of work. For example, Sharma et al. [367] aim to learn to produce subgoals in the robot's perspective, conditioned on a human video. Liu et al. [222] seeks to learn a translation model based on the paired demonstrations directly. Collecting paired demonstrations is challenging, and only a limited amount of data can be collected. Thus, previous work [381, 440] has employed cycle-consistency [456, 97] to learn an unsupervised pairing. Similarly, Sermanet et al. [358] uses a contrastive loss between frames close to and far away from the anchor point in the video, in order to obtain a representation. Sermanet et al. [359] trains a classifier using human demonstrations, which is then used to build a reward function. Unsupervised methods can learn a translation model for single tasks, however they have to be trained in every new setting, which is time consuming. Most such approaches require many random interactions to learn representations, and this process often yields unstable models [381]. WHIRL, on the other hand, does not need any random data to learn representations, and can work with even a single demonstration, in a variety of in-the-wild settings.

Offline Videos and Datasets Instead of using human videos, recent approaches have attempted to employ a reacher-grabber tool as the demonstration collection device [385, 435, 283]. These approaches have the advantage of having a smaller domain gap between robot and human actions, since the videos are in first person view. However, such a setup limits the number of tasks that are achievable, and adds considerable effort in collecting the data, since the approaches are not able to use large-scale human datasets, for example Youtube videos. On the other hand, advances in many computer vision tasks such as action recognition [449, 124, 148, 49, 414, 104], video understanding [104, 211, 223] or self-supervised representation learning [151, 282, 59, 276] have leveraged videos collected offline. These video datasets include the Something-Something [124], Epic Kitchens [73] or ActivityNet datasets [103]. These can provide important semantic information as well a high amount of visual and task diversity, which can aid in generalization. Similarly, works such as Chen et al. [50] and Shao et al. [364] find that using a large-scale human datasets, augmented with a few demonstrations from the robot as well as task labels, can help learn a semantic action classifier which generalizes to new tasks. Unlike these approaches, we do not use any task labels or robot specific fine-tuning for the feedback module. Embedding task specific knowledge into reward classifiers does not scale to in-the-wild settings, contrary to our approach.

Learning Action Policies from Priors While learning reward functions and representations from offline videos can be useful in robotics, videos of humans contain stronger priors. Learning keypoints [51, 189, 425] or object-level [305, 347] from videos, and using these as input to a control policy has been shown to be useful for certain tasks, but requires knowledge of the task and careful design, for example knowing the number of objects or keypoints. This can be a limiting factor when trying to scale to a general robot setup. Previous approaches have also used hand [200, 275] and object tracking [429] to learn action policies, however, these have been limited to simple settings and require very structured planning algorithms that are task specific. Our approach on the other hand is flexible and works for almost any manipulation task. Previous approaches do not perform any iterative improvement, contrary to WHIRL.

2.3 Human-to-Robot Visual Imitation In the Wild

We address the challenge of learning from humans by extracting priors from observing their actions, leveraging the priors to learn an interaction policy in the real world, and exploring around the prior in an efficient manner. We build a general robot learning algorithm that can work in many in-the-wild settings. We call this approach WHIRL: In-the-Wild Human Imitating Robot Learning. In this section, we describe how WHIRL works.



(a) Drawer (b) Dishwasher (c) Door

Figure 2.3: We perform various experiments in the wild. We select a subsample of tasks, as shown above, to perform a thorough study of our WHIRL as well as baselines and ablations. These tasks are: drawer, door and dishwasher opening and closing.

2.3.1 Human Priors

2.3.1.1 Extracting Human Priors

Most trajectories (τ) of interest for manipulation tasks can be broken down into smaller sub-trajectories: $\tau_{\text{pre-interaction}}$, $\tau_{\text{interaction}}$ and $\tau_{\text{post-interaction}}$. Throughout the chapter, we refer to these as primitives. A more complex task can be thought of as a composition of such primitives. Once we are able to use human videos to estimate these primitives, we can try to deploy these on a robot, despite any differences in morphology. Videos of the desired task (V), such as door opening, are used to obtain this trajectory parameterization. The key components of a video of a human performing a task include how the target is moving as well as where and when the interactions happen. We describe how we infer this information from third person videos below.

Extracting Hand Information We process each individual frame V_t of the video (V) at timestep t to obtain an estimate of the position of the hand: x_t, y_t, z_t. We obtain this pose using the 100DOH detection model [362], built on top of Faster-RCNN [331] and trained to output hand bounding box (b_t). This is a continuous vector of coordinates in image space. The hand position (in the camera frame) is referred to as h_t. In order for the robot to grasp and interact with an object, the orientation of the wrist and the force applied on the gripper are important as well. We use the MANO [336] parameterization of hands in order to obtain these. Specifically, we use the part of the parameterization that describes the rotation of the wrist, $\theta^{(t)}_{\text{hand}}$.

Extracting Interaction Information Inferring the position of the hand can give useful information, but we also need to understand when the hand interacts with an object. Detecting contact is important in determining $\tau_{pre-interaction}$: it determines where the interaction occurs. Thus, we employ the 100DOH [362] model to detect when this interaction occurs. We use this information and previously computed hand poses to extract waypoints for the robot. Specifically, we use the 100DOH model



(a) Detection (b) Interaction (c) Direction

Figure 2.4: We show the different components of the human prior. First we extract the position of the hand and possible object interactions (a). This indicates a possible area of interaction (b) and direction for moving the robot hand (c). We project these to the robot's action space and execute the trajectory.

to obtain a discrete valued contact variable: c_t. This represents a possible contact that might be occurring at frame t of the video. The possible options are: no contact, contact with portable or fixed object, and self contact. However, since out-of-the-box detections in unstructured settings can be noisy, we employ the Savitzky–Golay [345] filter for smoothing c_t across timesteps. Using smoothed detection $\hat{c}_t$ we determine the time-step where the interaction started in the video: $t_{\text{interaction}}$ and when it ended: t_{end}. We denote the hand position at these timesteps as $h_{\text{interaction}}$ and h_{end}. In order to not overfit to the detections, we in fact sample from a distribution centered around the start and end points. We also sample intermediate trajectory waypoints, h_{mid}. We additionally use a simple binary representation of a grasp, determined from the contact variable $\hat{c}_t$.
Overall, our extracted prior from a video demonstration from a human can be described as a set of interaction waypoints: $h_{\text{interaction}}$, h_{mid}, and h_{end}, a grasp or interaction orientation measure θ_{hand}, and commands to close or open the hand: $o_{1:T}$ (where T is the length of the video). Figure 2.4 shows the different parts of the human prior we use. Note that some tasks may require a more densely sampled set of waypoints. For simplicity we think of h_{mid} as a single point, but it can be also a set of midpoints in the hand trajectory.

2.3.1.2 Converting Human Priors to Robot Priors

Once we obtain the desired trajectories from human videos, we can convert them into the robots frame and obtain desired poses, using depth image (d_t) from the external camera. Our setup uses a depth image, but is compatible with any 3D pose estimation approach. Given a video V_k of length T, we then project the obtained priors, $h_{\text{interaction}}$, h_{mid}, h_{end}, θ_{hand}, $o_{1:T}$ to the robot's frame via 3D pose estimation from depth data. For both the gripper open and wrist orientation parameters, we use a robot-specific heuristic function, as every robot's coordinate axis is different. Let

the detected waypoints be $\boldsymbol{h} = h_{\text{interaction}}, h_{\text{mid}}, h_{\text{end}}$. This process can be described as:

$$f_{\text{map}}(\boldsymbol{h}, \theta_{\text{hand}}, o_{1:T}) = w_{\text{interaction}}, w_{\text{mid}}, w_{\text{end}}, \theta_{\text{YPR}}, g_{1:T} \triangleq \Psi_k \tag{2.1}$$

where $\boldsymbol{w}$ are waypoints in the robot's frame, θ_{YPR} is a wrist rotation (yaw-pitch-roll format) in the robots frame, and $g_{1:T}$ are robot gripper open/close continuous parameters. We refer to this vector by Ψ_k.

2.3.2 Policy Learning via Interaction

Human priors from videos can give a rough guideline on how to perform the task. They are useful because they can be distilled into a neural network policy, which can possibly generalize beyond the training data. However, directly executing the prior on the task will not generally lead to success, due to differences in morphologies between human and robot hands, inaccuracies in detections, or errors in the calibration process. Thus, we need to learn a policy via real world *interaction* in order to succeed at this task. Such a learning procedure must have 3 important properties. (1) The real world interactions must be safe. (2) While safe, the interactions must not be too restrictive. (3) This process must be sample efficient.
The safety of the interactions can be ensured by the human prior. Following the prior, even one that has errors, will lead to somewhat reasonable behavior, and is very likely to be safe. However, being too close to the prior will restrict the reach of the policy, and thus it will be unable able to solve the task. In order to address this challenge, we employ a *task policy* which aims to solve the task and a task agnostic *exploration policy* that explores around the human prior so that we do not fall into a local minimum. We describe the objective functions of these policies in the following sections. Finally, in order to ensure the learning process is sample efficient, we introduce a simple and easy to use zeroth order real world optimization procedure (similar to CEM). Since our goal is to efficiently perform many manipulation tasks in the wild, traditional RL methods are infeasible. A summary is in Algorithm 1.

Algorithm 1 Training Procedure for WHIRL

Require: Task videos: $V_{1:K}$, f_{map}: video to robot actions function, prior task and exploration policies: π, π_{exp}. Video-level (Φ) and frame-level (Φ_f) agent agnostic representation. M real world interactions per task.

```
while not converged do
    for k = 1...K do
        Ψ_k = f_map(V_k)
        for m = 1...M do
            Sample ΔΨ_{k,m} = π_exp(V_k, Ψ_k) (prob: p)
            Sample ΔΨ_{k,m} = π(V_k, Ψ_k) (prob: 1 − p)
            a_{j,m} = Ψ_k + ΔΨ_{k,m}
            Execute a_{k,m}, collect video: R_{k,m}
        end for
    end for
    for j = 1...K do
        rank Cost(Φ(R_{k,m}), Φ(V_k)) for every m
        pick E = {elite examples}
        fit π(.) as a VAE to Ψ_{k,m} ∈ E
        pick E_exp = {Φ_f(R_{k,m}) with highest "change"}
        fit π_exp(.) as a VAE to Ψ_{k,m} ∈ E_exp
    end for
end while
```

2.3.2.1 Policy Structure

When trying to achieve the desired task via interaction, it is easy to simply get stuck in the local minimum around the prior. Thus we not only need to train a task policy, but an exploration policy as well.

Task policy We would like to learn an interaction policy which will allow the agent to achieve the task. Given prior Ψ_k extracted from video V_k, we propose learning a (task and exploration) policy $\pi(\Psi_k, V_k) = \Delta\Psi_k$, outputting the residual to the prior. Residual learning is common in robot learning [168] as it allow for the policy to search around the prior in order to avoid unsafe behavior. Using this residual structure also allows the policy to initialize close to the human prior, and then explore from the prior as a starting point. Both of our policies are neural network based. The task policy π will try to maximize the robot's performance with respect to the demonstration video.

To be able to sample around the prior, the policy needs to learn a distribution and not just a mean prediction. For a single human video, there are multiple different ways a robot can perform the task. A naive stochastic neural network policy would not be able capture this multi-modal distribution and hence has difficulty in generalizing to new videos. We leverage Variational Auto-Encoders (VAEs) [179, 332] which are popularly used to capture multi-modal distributions. In particular, we fit a



(a) Robot Image (b) Robot Inpainted (c) Human Image (d) Human Inpainted

Figure 2.5: We show a sample of our agent agnostic representation, using the video inpainting method by Lee et al. [202]. An image of a human performing the task is shown in (c) and (d) shows the human inpainted from the image. Similarly for the robot, (a) and (b) show the original and inpainted images. We train segmentation models [422, 331] to obtain human and robot masks.

Conditional VAE [383] to learn a mapping from a set of samples $\Delta\Psi_{k,m}$ and an embedding of the input demonstration video $\phi(V_k)$. We condition the distribution on the video, V_k and learn to encode the prior residuals. The encoder, $q(z|c,x)$ takes input $x = \Delta\Psi_{k,m}$, and c is an embedding of the human video $\phi(V_k)$. The decoder $p(x|z,c)$ takes a latent sample from $p(z)$ as well as the human video embedding. At inference time, we can use the policy, π to output $\Delta\hat{\Psi}_k$ (the residual), conditioned on video $\phi(V_k)$ and latent $z \sim \mathcal{N}(0,1)$, where $\pi = p(x|z,c)$. Since this policy is conditioned on an input video, with enough collected data it is able to generalize to new human demonstrations as well.

Exploration Policy On the other hand, the exploration policy will try to explore around the prior. Many methods of exploration have been studied in literature, for example intrinsic motivation [289], or maximizing state coverage [110]. Instead, our exploration policy, π_{exp}, aims to maximize the change that the agent causes in the environment. Since our actions are close to the prior, it is likely that any changes caused by the agent in the environment will be meaningful and not destructive. Mathematically, for a given video R_k this can be described as:

$$c_k = \max_{i,j} ||\Phi_f(R_{k,i}) - \Phi_f(R_{k,j})||_2 \tag{2.2}$$

where i and j are different frames of the video and Φ_f is a frame-by-frame embedding of the video. This exploration policy uses the same exact inputs and setup as the task policy, as well as the CVAE architecture.

As the robot interacts with the environment, we want the task policy to improve to achieve more success. Thus, we need to have a notion of how good the robot is doing compared to the target human video. Given that human and robots have different morphologies, and perform tasks differently, how do we create a representation space for our objective function?

2.3.2.2 Representations for Human-to-Robot Video Alignment

Trying to learn correspondences between human and robot videos can be a challenging task. Prior works [367, 356, 381, 359] have attempted to achieve this via learning paired or unpaired videos from a single scene to learn a joint embedding. This would not scale to large set of in-the-wild manipulation tasks described in Figure 2.1. Instead of trying to learn a tight coupling between human and robots, we aim to get a comparison between human and robot video at a *high level*. We postulate that the effect that the agent had on the environment is more important than how the agent moved, since that can vary with different morphologies. We embed both robot and human videos into a space that is agnostic to the agent. We find that inpainting both human and robot from the video allows us to do so. We employ Copy-Paste Networks [202], which, given a mask of an agent, trains a network to copy information from other video frames and inpaint the area masked out. An example of this procedure can be seen in Figure 2.5.
Only inpainting the human out of the video is not enough to compare the two semantically, since the videos might be of different lengths, different speeds or may have other minor differences. Advances in action recognition [449, 124, 148, 49, 414, 104] have allowed models to determine if two videos which may look different are performing the same task. We use such an action recognition model from Monfort et al. [256], trained on large-scale passive data. This model takes in an entire video, and outputs an embedding. We call the composition of the action recognition model and the inpainting model our *agent-agnostic representation*. We denote this with Φ. Using this agent-agnostic representation, human video V and robot video R, our objective function is the distance:

$$|\Phi(V) - \Phi(R_k)||_2 \tag{2.3}$$

We use this function to train the task policy. The exploration policy objective (Equation 2.2) which is to maximize "change" leverages frame-wise version of Φ, denoted by Φ_f. In order to increase robustness, we sample costs over multiple embeddings with different video-level augmentations. Given a good way to align robot and human videos, how do we optimize our policies in a sample efficient way?

2.3.2.3 Sampling-based Optimization Procedure

RL methods have shown promise in learning by interaction. However, they remain too sample inefficient for large scale robot learning in unstructured settings. Instead, we propose a simple zeroth order sampling based alternative, in a similar fashion to CEM [342]. In our sampling procedure, we initially extract the prior from a third person human video V_k, and samples residuals:

$$\Delta\Psi_k \sim \mathcal{N}(0, \sigma^2)$$

We execute these M samples $\Psi_{k,m} + \Delta\Psi_{k,m}$ in the real world, and capture resulting videos $R_{k,m}$. We aim to fit π to the best performing samples. We repeat this process till convergence. In the following iterations, instead of sampling only from $\mathcal{N}(0, \sigma^2)$, we sample from π as well. Using the objective (agent-agnostic) functions described above, we rank trajectories based on these costs and fit the policy to the 10 highest ranking $\Delta\Psi_{k,m}$. This set, E, is the set of "elites" in CEM. The result of this procedure are trained exploration and task policies. We provide an overview in Algorithm 1.

2.4 Experimental Setup

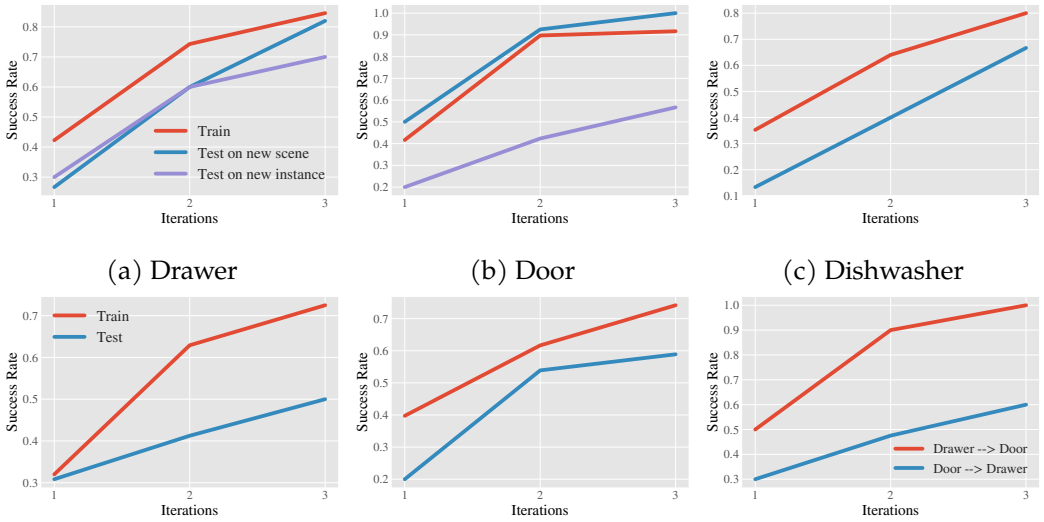
2.4.1 Experimental Details

Hardware In order to perform manipulation in the wild, a mobile robot is needed. Thus, we use the Stretch Robot [176]. This is a mobile base with a 6 dof arm and gripper. Pictures of the hardware setup can be seen in Figure 2.3. We use a Cartesian position control to command the translation of the wrist, and an in-built orientation controller for the rotation. The default gripper comes with suction cups as fingertips. Our setup uses an Intel Realsense D415 depth camera.

Environment and Data Collection We perform experiments on every-day objects and settings, for example drawers, dishwashers, fridges in different kitchens, doors to various cabinets. Data collection is performed in various in the wild settings. Each of the tasks presented in Figure 2.6 was trained over three demonstrations. Our setup involves 20 tasks seen in Figure 2.1.

2.4.2 Baselines and Ablations

Our method relies on several key ideas, such as policy learning from interactions, the high-level video alignment to compare robot trajectories and human demonstrations, and the task-agnostic policy we use. We test both the performance and sensitivity of WHIRL to design choices. We compare against several state-of-the-art baselines. Performing RL in the real world is infeasible [455] for our large set of tasks and in diverse settings. Thus we compare against offline RL, which learn interaction policies from data, and do not interact with the environment at training time. We modify a SOTA method in offline RL, Conservative Q-learning (CQL) [193] to work with our extracted human priors. The policy predicts the residual to the prior, just like ours, even using the same inputs as well. The reward function used is the negative L2 error between the 3D ResNet [104] embedding of target and robot videos. This baseline is trained with same number of samples as WHIRL (30 samples x 2 training iterations). We call this approach `CQL`. We also train `CQL` with the same objective function as WHIRL (agent-agnostic). We call this baseline `CQL-ours`. We compare against competing SOTA approaches for learning joint human and robot embedding spaces. We train a Time Contrastive Network (TCN) [356] to extract a representation



(a) Drawer (b) Door (c) Dishwasher

(d) Object in Shelf (e) Multi-task (f) Task-level Generalization

Figure 2.6: We present results of our thorough investigation of WHIRL on various kitchen tasks such as drawer (a), door (b) and dishwasher (c) opening and closing, a task involving picking and placing different objects into shelves (d). We test multi-task policies (e) trained on a subset of the tasks and generalization between tasks (f). We report training and testing success rates (out of 1) for 3 iterations of training.

from human and robot videos. We call this baseline `CQL-TCN`. The reward used for CQL is the the distance (between human and robot videos) in TCN embedding space. Similarly, Cycle-GAN [456] has been shown to be useful translating robot and human demonstrations [381]. In the same fashion as Smith et al. [381], we employ Cycle-GAN representations (trained on both human and robot videos) as an embedding space for the reward function. We refer to this baseline as `CQL-CycleGAN`. Finally, we compare to an off-the-shelf implementation of Behavior Cloning (`BC`). This approach is similar to WHIRL, however is without iterative refinement. A key difference is that the policy is trained with a standard L2 loss on the output actions, unlike our VAE-based policy.

2.5 Results

We evaluate WHIRL in various real world, in-the-wild settings in order to answer the following questions:

- Can WHIRL work for a large set of in-the-wild robot manipulation tasks?
- How can we use WHIRL to generalize to new scenes, objects and settings?
- How much do the individual components (i.e. policy learning and agent agnostic cost function) of WHIRL help?

- How does WHIRL compare agents SOTA approaches?

We attempt to answer these via various experiments on in-the-wild tasks, analyzing generalization capabilities of WHIRL on a few tasks, comparing to competing methods, and addressing the question of the importance of the individual components of our method, such as iterative improvement, our agent-agnostic objective and the exploration policy. In Figure 2.6 we present the results of our experiments on four tasks: opening and closing a drawer, door and dishwasher, as well as placing different objects in shelves. The setup is as described in Section 2.4.

2.5.1 Robot Learning in the Wild

We provide results on a large set of 20 tasks, ranging from turning on a water tap to folding a shirt. Images of these tasks can be seen in Figure 2.1. We show that WHIRL is able to to scale to a wide variety of tasks that involve large fixed objects such as fridges, or smaller rigid objects such as our ball-in-hoop task, or handling soft objects such as our shirt folding and whiteboard cleaning tasks. We are able to train these in only a few hours, in diverse locations and settings. We provide videos of these tasks in the supplementary material and at `https://human2robot.github.io`.

2.5.2 Evaluation and Comparison to Baselines

Tasks We perform a drawer opening and closing task in a kitchen. We report results averaged across three human demonstrations on two drawers (Figure 2.3a), and 3 iterations of WHIRL. Similarly to the drawer task, we perform a door opening and closing task. Figure 2.3b shows the task setup. We train on two doors and test on the third one. We perform 3 iterations of WHIRL. The third task involves opening and closing a dishwasher, as shown in Figure 2.3c. Due to a lack of dishwashers in the kitchens, we only train on one dishwasher, and test on a held-out dishwasher in a different scene. For the fourth task we train a policy for picking and placing four objects in three different shelves. These objects vary in size and shape (bottles, cans, etc.). We provide a single demonstration for each object. We test on two held-out objects and a held-out shelf placement (starting point is top shelf and goal is middle shelf).

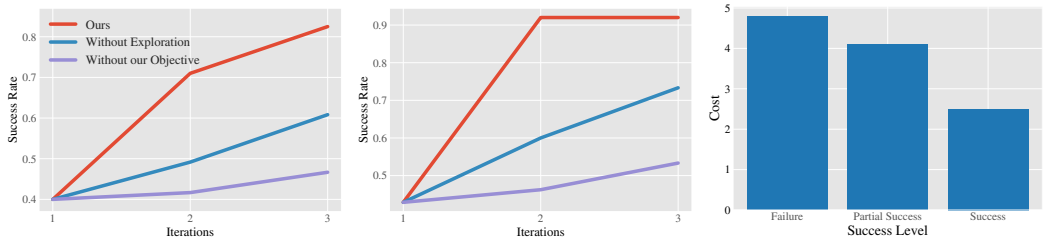
Training Evaluation We look into the (training) results running WHIRL on the various tasks described above. In Figure 2.6a, we show a learning curves of success rates over 3 iterations of WHIRL for the drawer task. The training curve (red) shows an increase from about a 43% success rate to 83% success rate after two iterations. The initial success rate is the success rate of the prior. We see a clear improvement during training with WHIRL. For the door task (Figure 2.6b, red), the training curve shows an increase in performance from about 40% success to 92 % success, indicating that WHIRL is able to learn to improve iteratively. The training curve for

	Drawer	Door
No iterative improvement:		
Behavior Cloning	0.53	0.30
Offline RL (`CQL-ours`)	0.47	0.30
Offline RL (`CQL-CycleGAN`) [381]	0.23	0.30
Offline RL (`CQL-TCN`) [356]	0.27	0.20
Offline RL (`CQL`) [193]	0.33	0.13
No agent-agnotic objective:		
WHIRL (ours)	0.47	0.53
No Exploration Policy:		
WHIRL (ours)	0.60	0.73
WHIRL (ours)	**0.83**	**0.92**

Table 2.1: We present a set of evaluations on two real world tasks: drawer and door. The task is to imitate the third person demonstration of the human. The results presented our averaged over 30 trials.

the dishwasher task (Figure 2.6c, red) shows that WHIRL also improves for this task, similarly to the drawer and door tasks. Figure 2.6d provides learning curves for the task of picking and placing objects from shelves. We can see that the the train (red) curves show improvement. However, we did note that this task required a lot more precision than the kitchen tasks presented above, which is why we see the test success rate to be much lower than the train one. We found that precision mattered immensely: very small prediction errors would lead to failure.

Comparison to Baselines We compare WHIRL to both offline RL and Behavior Cloning. We present results of multiple instances of offline RL in Table 2.1 (we report and compare training results). All methods had the same amount of data to train on. We report success rates out of 1. Our method strongly outperforms all the baselines. Offline RL, especially with smaller datasets, has difficulty in learning without any online interaction. Learning both actor and critic require data that covers more of the action space than is likely available in the wild. Interestingly, `CQL-ours` tends to outperform other approaches such as `CQL-CycleGan` (similar to that presented by Smith et al. [381]) `CQL-TCN` (similar to Sermanet et al. [356]), and `CQL` [193]. Similarly, Behavior Cloning, which uses our agent-agnostic cost function to filter the top trajectories, mostly outperforms the offline RL approaches. This indicates that our objective function is able to differentiate between good and bad trajectories in many algorithmic settings.



(a) Drawer (b) Door (c) Cost Function

Figure 2.7: We ablate different aspects of WHIRL. We analyze the need for our task-agnostic exploration policy, as well as our agent-agnostic representations. (a) is the drawer task, (b) is the door task. In (c) we analyze on the drawer task how our cost functions compare for different levels of success obtained by a trajectory.

2.5.3 Generalization to New Instances

We also evaluate how well policies trained with WHIRL are able to perform on new instances of the same task that they were trained on. In the drawer task, we test the policy on a held-out drawer. The learning curve for the held-out drawer (Figure 2.6a, blue) shows that the achieved success is lower than that on the training drawers. The detected prior for each demonstration may have different biases and errors. A policy may not necessarily transfer as well to a new demonstration without any training, however, as we see here, we expect improvement as there are commonalities in the structure of the task. Another common aspect is the camera and geometry information for both train and test demonstrations (since the view is the same). For the door task (Figure 2.6b, blue), interestingly, the success rate is higher here than the train one. As discussed previously, we expect that this is due to the strength of the prior for this specific door. For the object task, The policy was tested on two held objects and held out shelf placements. The test (blue) curves in Figure 2.6d shows a definite improvement in success rate over multiple iterations. From these experiments, we see that WHIRL does have the ability to generalize to new instances of the same task it was trained on.

2.5.4 Generalization to New Scenes:

To analyze how well WHIRL performs in a new scene, where the calibration and geometry are different, we try the trained policy on a drawer in a different part of the kitchen. Note that the camera angle and view also change. The resulting curve is shown in Figure 2.6a (purple). We see that similar to the test curve on the held-out drawer, there is definitely an increase in the success rate, however, the performance is still worse than the train drawers. In the case of the door task (Figure 2.6b, purple) the performance of this policy in a new setting is significantly worse, unlike the drawer task,. This is likely due to mismatches in the train and test demonstrations. Although the performance is not as strong, there is still an improvement in success

rate from 20% to about 57%. For the dishwasher task (Figure 2.6c, blue) we see a strong improvement in the success rate, similarly to the other two tasks. As expected, the policy does not perform as well as it does on the train dishwasher. We see that WHIRL allows for generalization to new scenes, however, in most cases the performance is worse than running WHIRL on new instances, most likely due to large visual changes, as well as different geometry and calibrations.

2.5.5 Generalization to New Settings:

In order to test the generalization between tasks, we test the trained policy on a held-out door (Figure 2.3b), in order to test the task level generalization. We see a strong performance on the door, as shown in Figure 2.6f (red). We see generalization from the drawer policy to the door task. We suspect that the final high success rate is due to the fact that the prior for this specific door was much more accurate than for other instances or tasks. Nevertheless, we see an improvement on the held-out door as the policy is trained more, indicating that WHIRL is able to improve the performance for not only the task at hand, but also allows for some degree of generalization across tasks. For transferring the policy trained on the door task to the drawer task, we see in Figure 2.6f (blue) an improvement in performance on the drawer task, similarly to the policy trained on the drawer task. This definitely indicates that there is some degree of task-level generalization in WHIRL.

2.5.6 Multi-task Generalization

In the previously described experiments, we trained policies for one task only, even if it was tested on another one. With this experiment, we aim to answer how training a joint policy would work. Using a similar approach as described above for each of the 3 tasks (drawer, door and dishwasher) we train the same policy π. We test this policy on the same test scenes as the previous experiments. We present results in Figure 2.6e. We can see clear improvement over all the iterations, both in the training and testing instances. However, the success rates are lower than policies trained for individual tasks which makes sense. We see a big increase from the first iteration. This is likely because there is more data thus generalization becomes a little easier as compared to individual policies. However, as more training happens this does not remain the case.

2.5.7 Sensitivity of WHIRL

We test the sensitivity of our method to both the inpainting process and the exploration policy. Firstly, We train WHIRL without the agent-agnostic representations. Secondly, We also compare WHIRL against a version which does not use the exploration policy. In Figure 2.7 we perform ablations to test the sensitivity of WHIRL to various components. In previous experiments, we have seen that the iterative

improvement provides a lot of benefit, as evidence by the increase in performance over iterations in almost all of the tasks and scenarios shown in Figure 2.6, as well as the boost in performance over offline RL methods, as shown in Table 2.1.

Agent-Agnostic Objective We train our policy without our agent-agnostic objective function, on both the drawer and door tasks, as shown in Figure 2.7a and 2.7b (purple). We see almost no gain in success rate from the initial samples from the prior in either task. This is likely due to the fact that video alignment models focus too much on the agents, thus the true top trajectories might not be selected. We conclude that an agent-agnostic objective is crucial to the success of WHIRL.

Exploration We test a version of WHIRL without our exploration policy. We see this in Figure 2.7a and 2.7b, in the blue curve. While the performance of this version of WHIRL outperforms the ablation that does not use agent-agnostic objective, we still see a drop in success rate from the WHIRL with the exploration policy. This shows that the method can learn without biasing exploration around maximizing "change" in the environment, but it will be slower.

Analyzing our Objective Function In Figure 2.7c, we show a bar plot of distances in our agent-agnostic embedding space for the drawer task. We present the cost for three types of trajectory. Note that each category is averaged over multiple trajectories. "Failure" trajectories completely fail, and the robot never touches the drawer. "Partial Success" trajectories open the wrong drawer or do not open/close the drawer fully. "Success" trajectories match the human demonstrations. We see that there the cost decreases between these three and that there is big drop between "Partial Success" and "Success". Since the cost is not close to 0, we expect there to be noise in the measurement of video alignment. However, empirically we found that our embedding space was robust enough to differentiate between successful and unsuccessful trials. In future work, we hope to train such an embedding space.

2.6 Conclusion

We propose WHIRL, an efficient real-world robot learning algorithm that can learn manipulation policies in-the-wild from human videos. We leverage advances in computer vision to understand human videos and obtain priors such as hand interactions, movement and direction. WHIRL is able to efficiently improve in the real world by using our sampling-based policy optimization strategy and agent-agnostic representations, as well as our proposed exploration strategy that maximizes the changes seen in the environment. We perform a thorough evaluation in terms of absolute performance, comparison to SOTA baselines, and generalization to new tasks and scenes on multiple tasks on real kitchens and see strong results. We show that our method is able to work on 20 different tasks in the wild (outside lab settings). While WHIRL can enable many tasks, how can the robot learn without online interactions and human demonstrations?

Part II

Behavior Priors from Manipulation

Chapter 3

Affordances from Human Videos as a Versatile Representation for Robotics



Figure 3.1: We leverage human videos to learn visual affordances that can be deployed on multiple real robot, in the wild, spanning several tasks and learning paradigms.

> *The meaning or value of a thing consists of what it affords... what we perceive when we look at objects are their affordances, not their qualities*
>
> J.J. Gibson (1979)

3.1 Motivation

Imagine standing in a brand-new kitchen. Before taking even a single action, we already have a good understanding of how most objects should be manipulated. This understanding goes beyond semantics as we have a belief of where to hold objects and which direction to move them in, allowing us to interact with it. For instance, the oven is opened by pulling the handle downwards, the tap should be

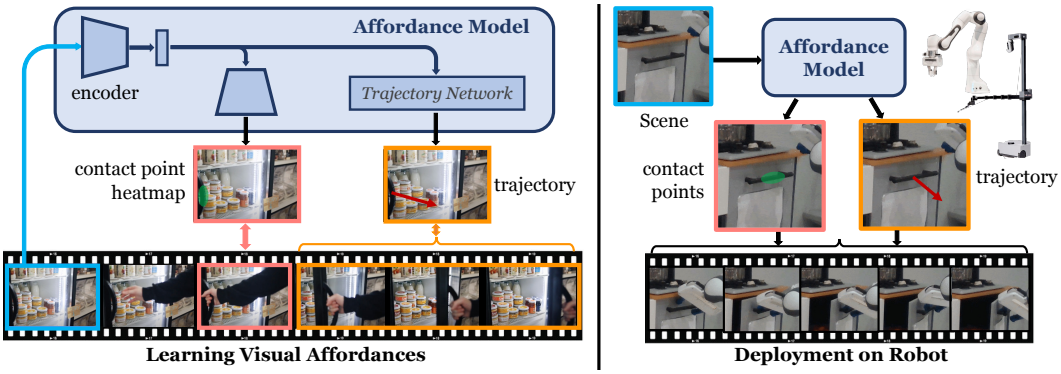


Figure 3.2: **VRB Overview**. First, we learn an actionable representation of visual affordances from human videos: the model predicts contact points and trajectory waypoints with supervision from future frames. For robot deployment, we query the affordance model and convert its outputs to 3D actions to execute.

turned sideways, drawers are to be pulled outwards, and light switches are turned on with a flick. While things don't always work as imagined and some exploration might be needed, but humans heavily rely on such visual *affordances* of objects to efficiently perform day-to-day tasks across environments [118, 119]. Extracting such actionable knowledge from videos has long inspired the vision community.

More recently, with improving performance on static datasets, the field is increasingly adopting a broader 'active' definition of vision through research in egocentric visual understanding and visual affordances from videos of human interaction. With deep learning, methods can now predict heatmaps of where a human would interact [265, 123] or segmentation of the object being interacted with [362]. Despite being motivated by the goal of enabling downstream robotic tasks, prior methods for affordance learning are tested primarily on human video datasets with no physical robot or in-the-wild experiments. Without integration with a robotic system, even the most basic question of how the affordance should be defined or represented remains unanswered, let alone evaluating its performance.

On the contrary, most robot learning approaches, whether imitation or reinforcement learning, approach a new task or a new environment *tabula rasa*. At best, the visual representation might be pretrained on some dataset [361, 271, 231, 423, 317, 426]. However, visual representations are only a small part of the larger problem. In robotics, especially in continuous control, the state space complexity grows exponentially with actions. Thus, even with perfect perception, knowing what to do is difficult. Given an image, current computer vision approaches can label most of the objects, and even tell us approximately where they are but this is not sufficient for the robot to perform the task. It also needs to know *where* and how to manipulate the object, and figuring this out from scratch in every new environment is virtually impossible for all but the simplest of tasks. How do we alleviate this clear gap between visual learning and robotics?

In this chapter, we propose to rethink visual affordances as a means to bridge vision and robotics. We argue that rich video datasets of humans interacting can offer a lot more actionable information beyond just replacing ImageNet as a pretrained visual encoder for robot learning. Particularly, human interactions are a rich source of how a wide range of objects can be held and what are useful ways to manipulate their state. However, several challenges hinder the smooth integration of vision and robotics. We group them into three parts. *First*, what is an actionable way to represent affordances? *Second*, how to learn this representation in a data-driven and scalable manner? *Third*, how to adapt visual affordances for deployment across robot learning paradigms? To answer the first question, we find that contact points and interaction directions are excellent robot-centric representations of visual affordances, as well as modeling the inherent multi-modality of possible interactions. We make effective use of egocentric datasets in order to tackle the second question. In particular, we reformulate the data to focus on frames without humans for predicting contact points and the interaction directions. To extract free supervision for this prediction, we utilize off-the-shelf tools for estimating egomotion, human pose, and hand-object interaction. Finally, we show how to seamlessly integrate these affordance priors with different kinds of robot learning paradigms. We thus call our approach **V**ision-**R**obotics **B**ridge (VRB) due to its core goal of bridging vision and robotics.

We evaluate both the quality of our affordances and their usefulness for 4 different robotic paradigms – imitation and offline learning, exploration, visual goal-reaching, and using the affordance model as a parameterization for action spaces. These are studied via extensive and rigorous real-world experiments on physical robots which span across 10 real-world tasks, 4 environments, and 2 robot hardware platforms. Many of these tasks are performed *in-the-wild* outside of lab environments (see Figure 3.1). We find that VRB outperforms other state-of-the-art human hand-object affordance models, and enables high-performance robot learning in the wild without requiring any simulation. Finally, we also observe that our affordance model learns a good visual representation for robotics as a byproduct. We highlight that all the evaluations are **performed in the real world spanning several hundred hours of robot running time** which is a very large-scale evaluation in robotics.

3.2 Related Work

Affordance and Interaction Learning from Videos. Given a scene, one can predict interactions using geometry-based rules for objects via 3D scene understanding [149, 451, 260], estimating 3D physical attributes [100, 26, 131, 459] or through segmentation models trained on semantic interactions [340, 346], and thus require specialized datasets. More general interaction information can be learned from large human datasets [209, 74, 72, 219, 76, 128], to predict object information [460, 112] (RGB & 3D) [29], graphs [92] or environment information [266, 108] such as heatmaps [123, 265]. Approaches also track human poses, especially hands

[362, 215, 338, 72, 216, 47, 431]. Similarly, in action anticipation and human motion forecasting, high-level semantic or low level actions are predicted using visual history [186, 334, 115, 160, 156, 410, 4, 74, 78, 161, 39, 113, 195, 409, 128, 114, 253, 245, 120]. Since our observations only have robot arms and no human hands, we adopt a robot-first formulation, only modeling the contact point and post-contact phase of interaction.

Visual Robot Learning. Learning control from visual inputs directly is an important challenge. Previous works have leveraged spatial structures of convolutional networks to directly output locations for grasping and pushing from just an image of the scene [303, 443, 442], which can limit the type of tasks possible. It is also possible to directly learn control end-to-end [203, 170] which while general, is quite sample inefficient in the real world. It has been common to introduce some form of prior derived from human knowledge, which could take the form of corrective interactions [133, 225, 86], structured policy spaces [273, 71, 327, 311, 11, 162, 25, 25, 364, 427], offline robotics data [99, 192, 193, 240, 319], using pretrained visual representations [361, 286, 271, 424, 426] or human demonstrations [356, 50, 23, 381, 367, 364].

Learning Manipulation from Humans. Extensive work has been done on Learning from Demonstrations (LfD) where human supervision is usually provided through teleoperation (of a joystick or VR interface) [387, 448, 255] or kinesthetic teaching, where a user physically moves the robot arm [311, 43, 62, 101, 233].With both these approaches, collecting demonstrations is tedious and slow. Recently, works have shown alternate ways to provide human demonstrations, via hand pose estimation and retargeting [378, 17, 430, 369, 313] in robot hands, but are mostly restricted to tabletop setups. First and third person human demonstrations have been used to train policies directly, transferred either via a handheld gripper [385, 435, 283] or using online adaptation [23]. In contrast to directly mimicking a demonstration, we learn robot-centric *affordances* from passive human videos that provide a great initialization for downstream robot tasks, unlike previous work which require in-domain demonstrations.

3.3 Affordances from Human Videos (VRB)

Our goal is to learn affordance priors from large-scale egocentric videos of human interaction, and then use them to expedite robot learning in the wild. This requires addressing the three questions discussed in Sec. 3.1 about how to best represent affordances, how to extract them and how to use them across robot learning paradigms.

3.3.1 Actionable Representation for Affordances

Affordances are only meaningful if there is an actor to execute them. For example, a chair has a sitting affordance only if it is possible for some person to sit on it. This

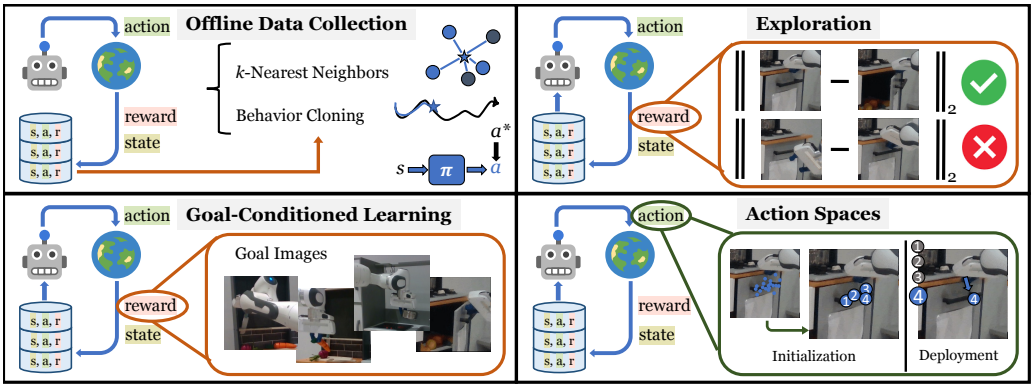


Figure 3.3: **Robot Learning Paradigms** : (a) Offline Data Collection – Used to investigate the quality of the collected data. (b) Exploration – The robot needs to use intrinsic rewards to improve (c) Goal-Conditioned Learning – A desired task is specified via a goal image, used to provide reward. (d) Action Spaces – Reduced action spaces are easier to search and allow for discrete control.

property makes it clear that the most natural way to extract human affordances is by watching how people interact with the world. However, what is the right object-centric representation for affordances: is it a heatmap of where the human makes contact? Is it the pre and postcondition of the object? Is it a description of the human interaction? All of these are correct answers and have been studied in prior works [265, 216, 149]. However, the affordance parameterization should be amenable to deployment on robots.

If we want the robot to *a priori* understand how to manipulate a pan (Fig. 3.1, 3.4) without any interaction, then a seemingly simple solution is to exactly model human movement from videos [216], but this leads to a human-centric model and will not generalize well because human morphology is starkly different from that of robots. Instead, we take a first-principles approach driven by the needs of robot learning. Knowledge of a robot body is often known, hence reaching a point in the 3D space is feasible using motion planning [196, 197, 174]. The difficulty is in figuring out where to interact (e.g. the handle of the lid) and then how to move after the contact is made (e.g., move the lid upwards).

Inspired by this, we adopt contact points and interaction directions as a simple actionable representation of visual affordance that can be easily transferred to robots. We use the notation c for a contact point and τ for interaction direction, both in the pixel space. Specifically, $\tau = f(I_t, h_t)$, where I_t is the image at timestep t, h_t is the human hand location in pixel space, and f is a learned model. We find that our affordance representation outperforms prior formulations across robots. Notably, the c and τ abstraction makes the affordance prior agnostic to the morphological differences across robots.

3.3.2 Learning Affordances from Egocentric Videos

The next question is how to extract c and τ from human videos in a scalable data-driven manner while dealing with the presence of human body or hand in the visual input. VRB tackles this through a robot-first approach.

3.3.2.1 Extracting Affordances from Human Videos

Consider a video V, say of a person opening a door, consisting of T frames *i.e.* $V = \{I_1, ..., I_T\}$. We have a twofold objective — find *where* and *when* the contact happened, and estimate how the hand moved after contact was made. This is used to supervise the predictive model $f_\theta(I_t)$ that outputs contact points and interaction directions. To do so, we utilize a widely-adopted hand-object detection model trained on human video data [362]. For each image I_t, this produces 2D bounding boxes of the hand h_t, and a discrete contact variable o_t. Using this information, we filter for frames where o_t indicates a contact in each video, and find the first timestep where contact occurs, t_{contact}.

The pixel-space positions of the hand $\{h_t\}_{t_{\text{contact}}}^{t'}$ constitute the interaction direction (τ). To extract contact points c, we use the corresponding hand bounding box, and apply skin color segmentation to find all points at the periphery of the hand segment that intersect with the bounding box of the object in contact. This gives us a set of N contact points $\{c^i\}^N$, where N can differ depending on the image, object, scene and type of interaction. How should the contact points be aggregated to train our affordance model (f_θ)? Some options include predicting the mean of $\{c^i\}^N$, or randomly sampling c^i. However, we seek to encourage multi-modality in the predictions, since a scene likely contains multiple possible interactions. To enable this, we fit a Gaussian mixture model (GMM) to the points. Let us define a distribution over contact points to be $p(c)$. We fit the GMM parameters (μ_k, Σ_k) and weights α_k.

$$p(c) = \underset{\mu_1,...,\mu_K,\Sigma_1,...,\Sigma_K}{\arg\max} \sum_{i=1}^{N} \sum_{k=1}^{K} \alpha_k \mathcal{N}(c^i | \mu_k, \Sigma_k) \tag{3.1}$$

We use these parameters of the above defined GMM with K clusters as targets for f_θ. To summarize, 1) we find the first timestep where contact occurs in the human video, t_{contact} 2) For c, we fit a GMM to the contact points around the hand at frame $I_{t_{\text{contact}}}$, parameterized by μ_k, Σ_k and 3) we find the post-contact trajectory of the 2D hand bounding box $\{h_t\}_{t_{\text{contact}}}^{t'}$ for τ.

Accounting for Camera Motion over Time: Consider a person opening a door. Not only do the person's hands move but their body and hence their head also move closer to the handle and then away from it. Therefore, we need to compensate for this egomotion of the human head/camera from time t_{contact} to t'. We address this by using the homography matrix at timestep t, $\mathcal{H}_t$ to project the points back into the coordinates of the starting frame. We obtain the homography matrix by matching

features between consecutive frames. We then use this to produce the transformed trajectory $\tau = \mathcal{H}_t \circ \{h_t\}_{t_{\text{contact}}}^{t'}$.

Addressing Human-Robot Visual Domain Shift: The training videos contain human body or hand in the frame but the human will not be present in downstream robotics task, generating domain shift. We deal with this issue with a simple yet elegant trick: we extract affordances in the frames with humans but then map those affordances back to the first frame when human was yet to enter the scene. For videos in which a human is always in frame, we either crop out the human in the initial frame if there is no interaction yet or discard the frame if the human is always in contact. We compute the contact points and interaction directions with respect to this human-less frame via the same homography procedure described above. This human-less frame is then used to condition our affordance model.

3.3.2.2 Training Affordance Model

Conditioned on the input image, the affordance model is trained to predict the extracted labels for contact points and interaction directions. However, naive joint prediction does not work well as the learning problem is inherently multi-modal. For instance, one would pick up a cup differently from a table depending on whether the goal is to pour it into the sink or take a sip from it. We handle this by predicting multiple heatmaps for interaction points using the same model, building a spatial probability distribution.

For ease of notation, we use $(\cdot)_\theta$ as a catch-all for all parameterized modules and use f_θ to denote our complete network. Fig. 3.2 shows an overview of our model. Input image I_t is encoded using a ResNet [154] visual encoder g_θ^{conv} to give a spatial latent representation z_t, i.e., $g_\theta^{\text{conv}}(I_t) = z_t$. We then project this latent z_t into K probability distributions or heatmaps using deconvolutional layers; concretely, $H_t = g_\theta^{\text{deconv}}(z_t)$. Using a spatial softmax, σ_{2D}, we get the estimation of the labels for GMM means, *i.e.*, μ_k. We found that keeping the covariance matrices fixed gave better results. Formally, the loss for contact point estimation is:

$$\mathcal{L}_{\text{contact}} = \left\| \mu_i - \sigma_{\text{2D}} \left(g_\theta^{\text{deconv}} \left(g_\theta^{\text{conv}} \left(I_t \right) \right) \right) \right\|_2 \tag{3.2}$$

To estimate interaction direction, we train a trajectory prediction network, $\mathcal{T}_\theta$, based on the latent representation z_t. We find that it is easier to optimize for *relative* shifts, *i.e.*, the direction of movement instead of absolute locations, assuming that the first point $\hat{w}_0$ is 0, since the contact points are already spatially grounded. Based on the success of Transformers for sequential prediction, we employ self-attention blocks [407] and train to optimize $\mathcal{L}_{\text{traj}} = \|\tau - \mathcal{T}_\theta(z_t)\|_2$. In a given scene, there are many objects a human could interact with, which may or may not be present in the training data. We tackle this uncertainty and avoid spurious correlations by sampling local crops of I_t around the contact points. These serve as the effective input to our network f_θ and enables better generalization.

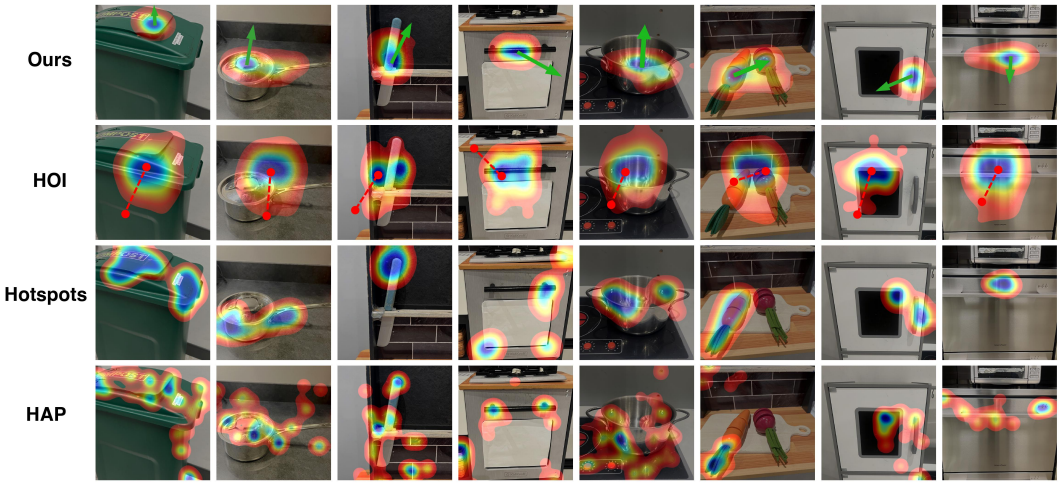


Figure 3.4: Qualitative affordance model outputs for VRB, HOI [216], Hotspots [123] and HAP [123], showing the predicted contact point region, and post-grasp trajectory (green arrow for VRB, red for HOI [216]). We can see that VRB produces the most meaningful affordances.

3.3.3 Robot Learning from Visual Affordances

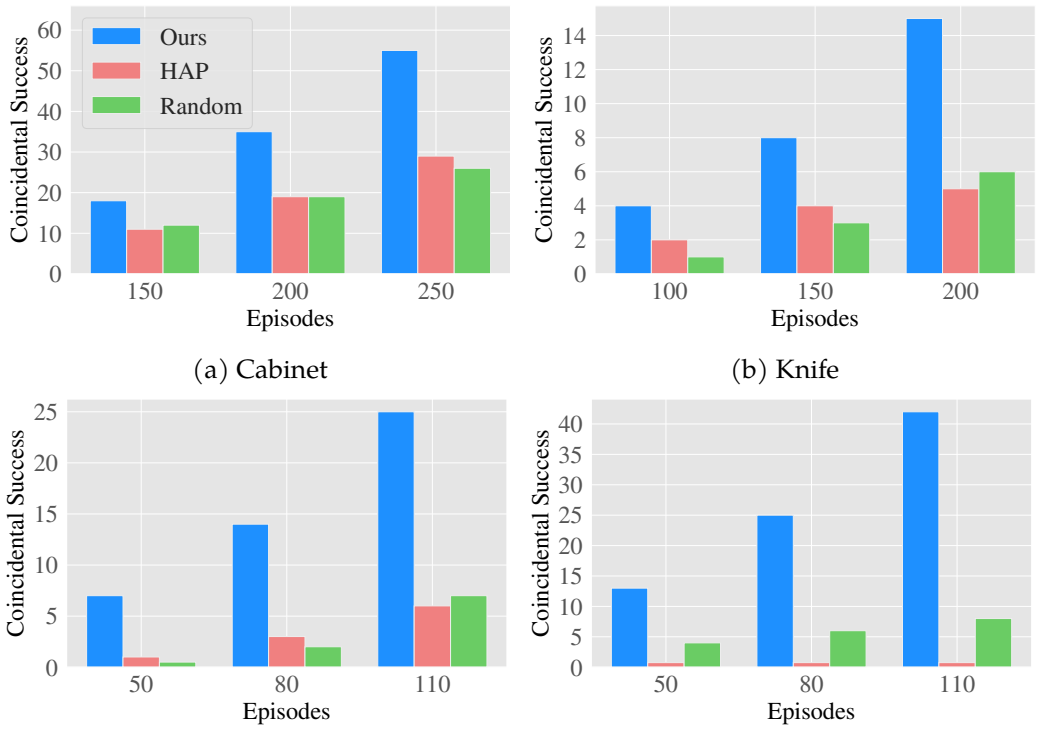
Instead of finding a particular way to use our affordance model for robotics, we show that it can bootstrap existing robot learning methods. In particular, we consider four different robotics paradigms as shown in Fig. 3.3.

A. Imitation Learning from Offline Data Collection Imitation learning is conventionally performed on data collected by human demonstrations, teleoperation, or scripted policies – all of which are expensive and only allow for small-scale data collection[436, 23, 367, 16, 40, 203]. On the other hand, using the affordance model, $f_\theta(\cdot)$ to guide the robot has a high probability of yielding 'interesting' interactions. Given an image input I_t, the affordance model produces $(c, \tau) = f_\theta(I_t)$, and we store $\{(I_t, (c, \tau))\}$ in a dataset $\mathcal{D}$. After sufficient data has been collected, we can use imitation learning to learn control policies, often to complete a specific task. A common approach for task specification is to use *goal images* that show the desired configuration of objects. Given the goal image, the *k-Nearest Neighbors* (k-NN) approach involves filtering trajectories in $\mathcal{D}$ based on their distance to the goal image in feature space. Further, the top (filtered) trajectories can be used for *behavior cloning* (BC) by training a policy, $\pi(c, \tau | I_t)$. We run both k-NN and behavior cloning on datasets collected by different methods in Sec. 3.4.1. Using the same IL approach for different datasets is also useful for comparing the relative quality of the data. This is because higher relative success for a particular dataset implies that the data is qualitatively better, given that the same IL algorithm achieves worse performance on a different dataset. This indicates that the goal (or similar images) were likely seen during data collection.

B. Reward-Free Exploration The goal of exploration is to discover as many diverse skills as possible which can aid the robot in solving downstream tasks. Exploration methods are usually guided by *intrinsic rewards* that are self-generated by the robotic agent, and are not specific to any task [28, 289, 279, 398, 214, 309, 163, 322, 248]. However, starting exploration from scratch is too inefficient in the real world, as the robot can spend an extremely large amount of time trying to explore and still not learn meaningful skills to solve tasks desired by humans. Here our affordance model can be greatly beneficial by bootstrapping the exploration from the predicted affordances allowing the agent to focus on parts of the scene likely to be of interest to humans. To operationalize this, we first use the affordance model $f_\theta(.)$ for data-collection. We then rank all the trajectories collected using a task-agnostic exploration metric, and fit a distribution h to the (c, τ) values of the top trajectories. For subsequent data collection, we sample from h with some probability, and otherwise use the affordance model f. This process can then be repeated, and the elite-fitting scheme will bootstrap from highly exploratory trajectories to improve exploration even further. For the exploration metric in our experiments, we maximize *environment change* $\text{EC}(I_i, I_j) = ||\phi(I_i) - \phi(I_j)||_2$, (similar to previous exploration approaches [23, 285]) between first and last images in the trajectory, where ϕ masks the robot and the loss is only taken on non-masked pixels.

C. Goal-Conditioned Learning While exploring the environment can lead to interesting skills, consider a robot that already knows its goal. Using this knowledge (*e.g.* an image of the opened door), it supervise its policy search. Goal images are frequently used to specify rewards in RL [416, 122, 269, 290, 116, 12, 268, 458, 252]. Using our affordance model can expedite the process of solving goal-specified tasks. Similar to the exploration setting, we rank trajectories and fit a distribution h to the (c, τ) values of the top trajectories, but here the metric is to minimize distance to the goal image I_g. The metric used in our experiments is to minimize $\text{EC}(I_T, I_g)$, where I_T is the last image in the trajectory, or to minimize $||\psi(I_g) - \psi(I_T)||_2^2$, where ψ is a feature space. Akin to exploration, subsequent data collection involves sampling from h and the affordance model f.

D. Affordance as an Action Space Unlike games with discrete spaces like Chess and Go where reinforcement learning is deployed *tabula rasa*, robots need to operate in continuous action spaces that are difficult to optimize over. A pragmatic alternative to continuous action spaces is parameterizing them in a spatial manner and assigning a primitive (e.g. grasping, pushing or placing) to each location [444, 443, 371]. While this generally limits the type of tasks that can be performed, our affordance model already seeks out interesting states, due to the data it is trained on. We first query the affordance model on the scene many times to obtain a large number of predictions. We then fit a GMM to these points to obtain a discrete set of (c, τ) values, and now the robot just needs to search over this space.



(a) Cabinet (b) Knife

(c) Stovepot (d) Shelf

Figure 3.5: **Exploration**: Coincidental success of VRB in comparison to random exploration or the exploration based on HAP [123].

3.4 Experimental Setup and Results

Through the four robot learning paradigms, shown in Fig. 3.3, we seek to answer the following questions: (1) Does our model enable a robot to collect *useful data* (imitation from offline data)?, (2) How much benefit does VRB provide to *exploration* methods?, (3) Can our method enable *goal-conditioned* learning?, and (4) Can our model be used to define a structured *action space* for robots? Finally, we also study whether our model learns meaningful *visual representations* for control as a byproduct and also analyze the *failure modes* and how they differ from prior work.

Robotics Setup We use two different robot platforms - the Franka Emika Panda arm and the Hello Stretch mobile manipulator. We run the Franka on two distinct play kitchen environments and test on tasks that involve interacting with a cabinet, a knife and some vegetables, and manipulation of a a shelf and a pot. The Hello robot is tested on multiple in-the wild tasks outside lab settings, including opening a garbage can, lifting a lid, opening a door, pulling out a drawer, and opening a dishwasher (Fig. 3.1). We also provide support for a simulation environment on the Franka-Kitchen benchmark [111].

Observation and Action space For each task, we estimate a task-space image-crop

	Cabinet	Knife	Veg	Shelf	Pot	Door	Lid	Drawer
k-Nearest Neighbors:								
HOI	0.2	0.1	0.1	0.6	0.0	0.4	0.0	0.6
HAP	0.3	0.0	0.3	0.0	0.1	0.2	0.0	0.1
Hotspots	0.4	0.0	0.1	0.0	**0.5**	0.4	0.3	0.5
Random	0.3	0.0	0.1	0.3	0.4	0.2	0.1	0.2
VRB **(ours)**	**0.6**	**0.3**	**0.6**	**0.8**	0.4	**1.0**	**0.4**	**1.0**
Behavior Cloning:								
HOI	0.3	0.0	0.3	0.0	0.1	0.2	0.0	0.1
HAP	0.5	0.0	**0.4**	0.0	0.3	0.1	0.0	0.1
Hotspots	0.2	0.0	0.0	0.0	**0.8**	0.1	0.0	0.7
Random	0.1	**0.1**	0.1	0.0	0.2	0.1	0.0	0.0
VRB **(ours)**	**0.6**	**0.1**	0.3	**0.3**	**0.8**	**0.9**	**0.2**	**0.9**

Table 3.1: **Imitation Learning**: Success rate for k-NN and Behavior Cloning on collected offline data using various affordance models. We find that VRB vastly outperforms prior approaches, indicating better quality of data.

using bounding boxes [454], and pass random sub-crops to f_θ. The prediction for contact points c and post-contact trajectory τ is in pixel space, which are projected into 3D for robot control using a calibrated robot-camera system (with an Intel RealSense D415i). The robot operates in 6DOF end-effector space – samples a rotation, moves to a contact point, grasps, and then moves to a post-contact position (see Sec. 3.3.1).

Baselines and Ablations: We compare against prior work that has tried to predict heatmaps from human video : 1) Hotspots [265] 2) Hands as Probes (HAP) [123], a modified version for our robot setup of Liu *et al.* [216] that predicts contact region and forecast hand poses: 3) HOI [216] and 4) a baseline that samples affordances at random (Random). HAP and Hotspots only output a contact point, and we randomly select a post-contact direction.

3.4.1 Quality of Collected Data for Imitation

We investigate VRB as a tool for useful data collection. We evaluate this on both our robots across 8 different environments, with results in Tab. 3.1. These are all unseen scenarios (not in train set). Tasks are specified for each environment using goal images (eg - open door, lifted pot etc), and we use the data collected (30-150 episodes) for two established offline learning methods: (1) k-Nearest Neighbors (k-NN) and (2) Behavior Cloning. k-NN [283] finds trajectories in the dataset that are close (via distance in feature space [271]) to the goal image. We run the 10-closest trajectories to the goal image and record whether the robot has achieved the task specified in the goal image. For behavior cloning, we train a network supervised with (image, waypoint) pairs from the collected dataset, and the resulting policy is run 10 times on the real system. With both k-NN and BC, our method outperforms prior tasks on 7 out of 8 tasks, with an average success rate of 57 %, with the runner-up method (Hotspots [265]) only getting 25 %. This shows that VRB leads to much better data

offline data quality, and thus can lead to better imitation learning performance. We additionally test for grasping held-out *rare* objects such as VR remotes or staplers, and find that VRB outperforms baselines.

3.4.2 Reward-Free Exploration

Here we study self-supervised exploration with no external rewards. We utilize environment change, *i.e.*, change in the position of objects as a task-agnostic metric for exploration [23]. For improved exploration, we bias sampling towards trajectories with a higher environment change metric. To evaluate the quality of exploration data, we measure how often does the robot achieves coincidental success *i.e.* reach a goal image configuration without having access to it. As shown in Fig. 3.5, we obtain consistent improvements over HAP [123] and random exploration raising performance multiple fold – from $3\times$ to $10\times$, for every task.

3.4.3 Goal-Conditioned Learning

The previous settings help robots improve their behaviors with data without an external reward or goal. Here we focus on goal-driven robot learning. Goals are often specified through images of the goal configuration. Note that goal images are also used in Sec. 3.4.1 but as part of a static dataset to imitate. Here, the robot policy is updated with new data being added to the buffer. We sample this dataset for trajectories that minimize visual change with respect to the goal image. As shown in Fig. 3.6, VRB learns faster and better HAP [123] and Random on this robot learning paradigm, over six diverse tasks.

3.4.4 Affordance as an Action Space

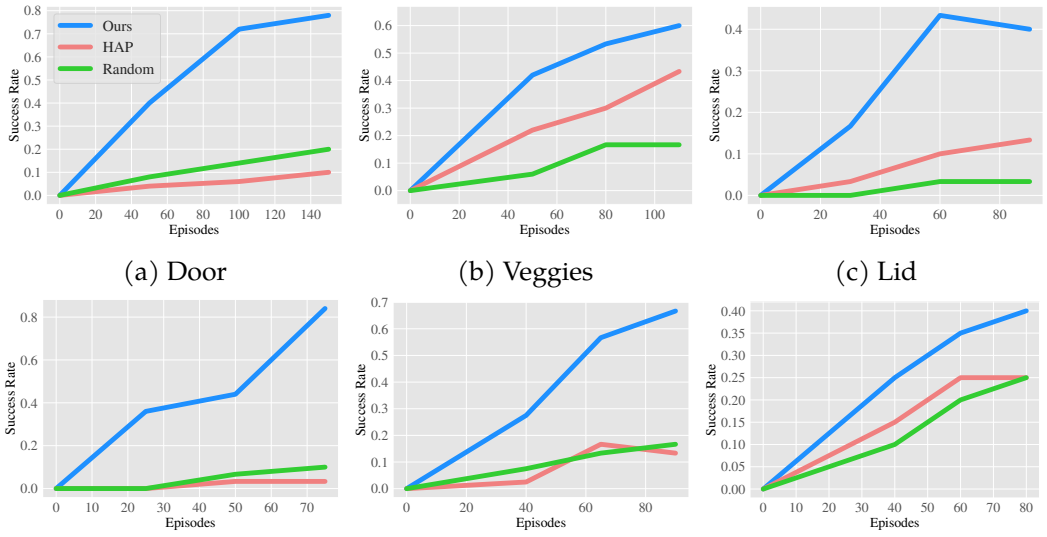
We utilize visual affordances to create a discrete action space using a set of contact points and interaction directions. We then train a Deep Q-Network (DQN) [254] over this action space, for the above goal-conditioned learning problem.In Fig. 3.7, we see that with VRB, the robot experiences more successes showing that a greater percentage of actions in the discretized action space correspond to meaningful object interactions.

	VRB	R3M [271]
`microwave`	**0.16**	0.10
`slide-door`	**0.84**	0.70
`door-open`	**0.13**	0.11

Table 3.2: Imitation with VRB vs. R3M [271] representation.

3.4.5 Analyzing Visual Representations

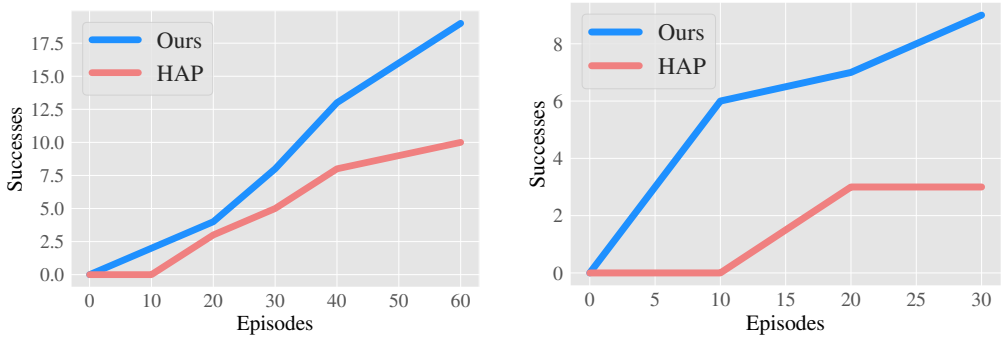
Beyond showing better utility for robot learning paradigms, we analyze the quality of visual representations of the encoder learned in VRB. Two standard evaluations for this are (1) if they can help for downstream tasks and (2) how meaningful distances in their feature spaces are.



(a) Door (b) Veggies (c) Lid

(d) Dishwasher (e) Drawer (f) Garbage Can

Figure 3.6: **Goal-conditioned Learning**: Success rate for reaching goal configuration for six different tasks. Sampling via VRB leads to faster learning and better final performance.



(a) Cabinet (b) Veggies

Figure 3.7: **Action Space**: Success using DQN with the discretized action space, for reaching a specified goal image.

Finetuning To investigate if the visual representations are effective for control, we directly finetune a policy on top of the (frozen) visual encoder. We evaluate on three simulated Franka environments, as shown in Tab. 3.2, and we see that VRB outperforms R3M on all tasks. (We finetuned the policy only for 2K steps, instead of 20K in the R3M paper). This demonstrates that VRB visual representations contain information that is useful for control.

Feature space distance We record the distance in feature space between the current and goal image for every timestep in the episode, for both VRB and R3M [271] on successful cabinet opening trajectories. As shown in Fig. 3.8, the distance for VRB decreases almost monotonically which correlates well with actual task progress.

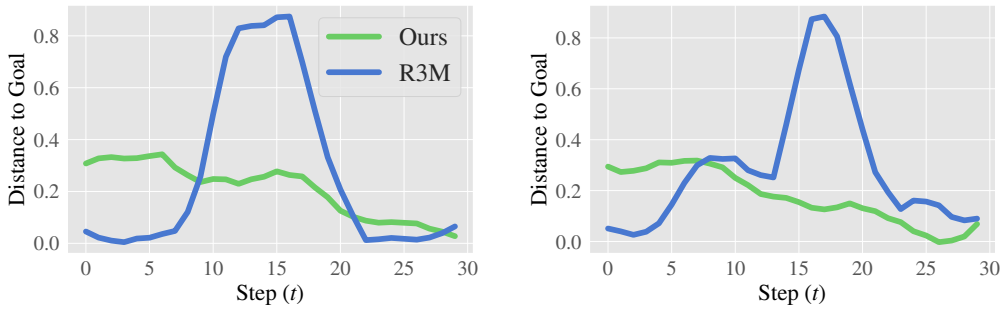


Figure 3.8: **Feature space distance**: Distance to goal in feature space for VRB decreases monotonically for door opening.

3.4.6 Failure Modes

While VRB and the baselines see qualitatively similar successes, VRB in general sees a larger number of them and the *average case* scenario for VRB is much better. For the cabinet opening task, we classify each collected episode into three categories: "Failure", "Partial Success" and "Success". While VRB has a higher number of successful trajectories compared to the baselines (almost $2\times$), the number of partial successes is more than $6\times$ (Fig. 3.9).



Figure 3.9: Failure mode analysis

3.5 Conclusion

We propose Vision-Robotics Bridge (VRB), a scalable approach for learning useful affordances from passive human video data and deploying them on many different robot learning paradigms (such as data collection for imitation, reward-free exploration, goal-conditioned learning, and paramterizing action spaces). Our affordance representation consists of contact points and post-contact trajectories. We demonstrate the effectiveness of this approach on the four paradigms and 10 different real-world robotics tasks, including many that are in the wild. We run thorough experiments, spanning over 200 hours, and show that VRB drastically outperforms prior approaches. However, we do not deploy VRB more complex, multi-stage tasks or higher degree of freedom morphologies such as dexterous hands.

Chapter 4

Learning Dexterous Affordances

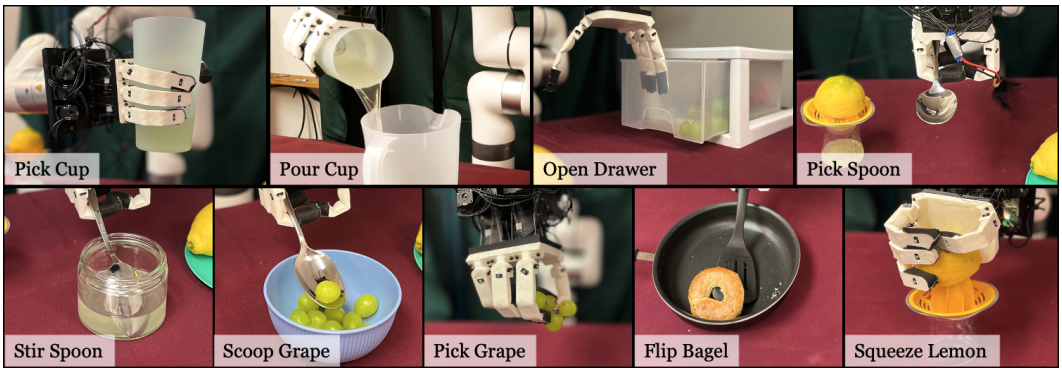


Figure 4.1: We present DEFT, a novel approach that can learn complex, dexterous tasks in the real world in an efficient manner. DEFT manipulates tools and soft objects without any robot demonstrations.

4.1 Motivation

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* [242], 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.

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* [109, 24, 123, 265, 362, 216, 220, 413]. 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 [86, 168], 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 4.1) 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.

4.2 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

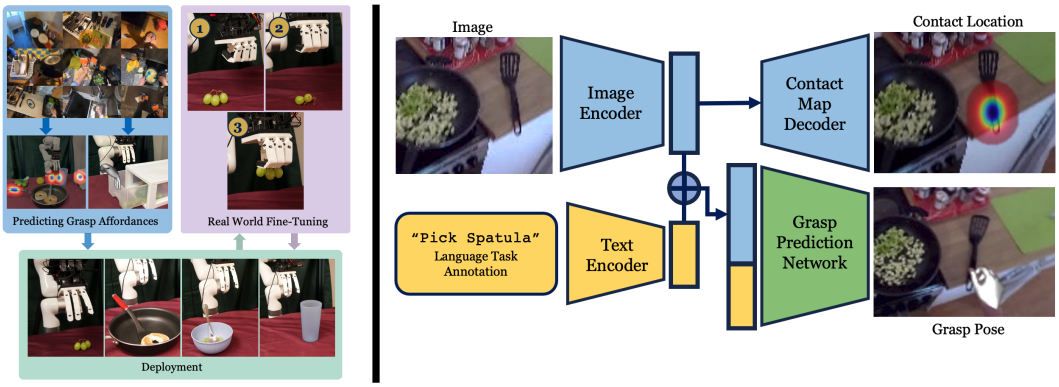


Figure 4.2: **Left:** DEFT consists of two phases: an affordance model that predicts grasp parameters followed by online fine-tuning with CEM. **Right:** Our affordance prediction setup predicts grasp location and pose.

flexibility in unstructured settings [175, 188, 257]. 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 [184, 298, 210, 310, 289]. Efforts have been made to scale end-to-end RL [207, 269, 6, 135, 170, 171] 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 [244], goal relabeling [13], trajectory guidance [205], visual imagined goals [269], or curiosity-driven exploration [248]. 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 [336] of axes aligned with the wrist and a 10-dimensional shape vector. MANOtorch from [428] aligns it with the anatomical joints. Many recent works detect MANO in monocular video [412, 172, 337]. Some also detect objects as well as the hand together [362, 432]. We use FrankMocap to detect the hand for this work. There are many recent datasets including the CMU Mocap Database [2] and Human3.6M [159] for human pose estimation, 100 Days of Hands [362] for hand-object interactions, FreiHand [461] for hand poses, Something-Something [125] for semantic interactions. ActivityNet datasets [103], or YouCook [81] are action-driven datasets that focus on dexterous manipulation. We use these three datasets: [127] is a large-scale dataset with human-object interactions, [221] for curated human-object interactions, and [73] which has many household kitchen tasks. In addition to learning exact human motion, many others focus on learning priors from human motion. [231, 272] 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 [143, 379, 401]. Moving further, human motion in internet datasets can be retargeted and used directly to pre-train robotic policies [369, 238]. Additionally, using human motion as a prior for RL can help with learning skills that are human-like [321, 292, 237]. Without using human data as priors, object reorientation using RL has been recently successful in a variety of settings [14, 57]. 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 [5, 243].

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 [434, 19]. Coupled with dynamics controllers, these robots can learn not to apply too much torque to the environment around them [230, 213, 177]. A variety of touch sensors [374, 437, 31, 391] 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.

4.3 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.

Parameter	Dimensions	Description
μ	3	XYZ grasp loc.
θ_{wrist}	3	Wrist grasp rot.
P	16	Finger joint angles

Table 4.1: Parameters that are fine-tuned in the real world.

4.3.1 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

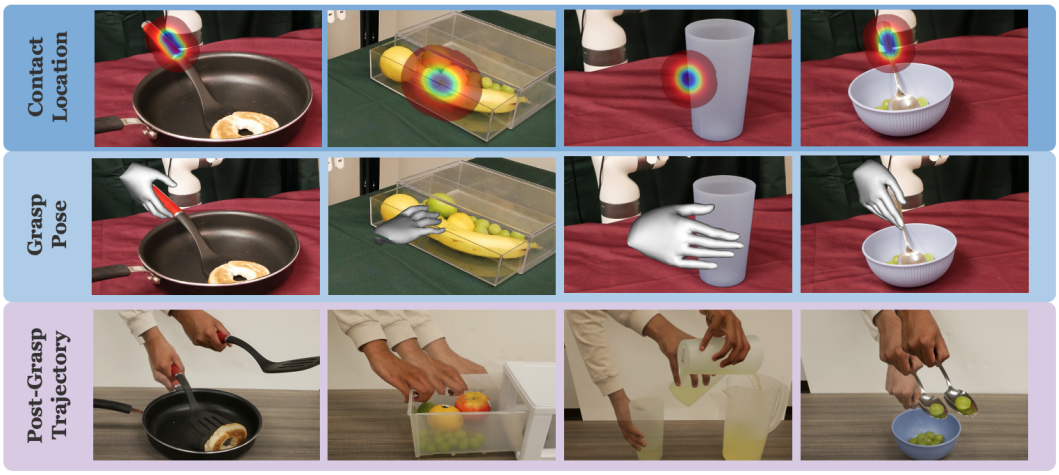


Figure 4.3: We produce three priors from human videos: the contact location (**top row**) and grasp pose (**middle row**) from the affordance prior; the post-grasp trajectory (**bottom row**) from a human demonstration of the task.

internet videos, humans frequently interact with a wide variety of objects. This data is especially useful in learning object affordances. Furthermore, one of the major obstacles in manipulating objects with few samples is accurately grasping the object. A model that can perform a strong grasp must learn *where* and *how* to grasp. Additionally, the task objective is important in determining object affordances– humans often grasp objects in different ways depending on their goal. Therefore, we extract three items from human videos: the grasp location, human grasp pose, and task.

Given a video clip $V = \{v_1, v_2, \ldots, v_T\}$, the first frame v_t where the hand touches the object is found using an off-the-shelf hand-object detection model [362]. Similar to previous approaches [24, 123, 216, 265], 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 [453] 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 [337] 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 [127] for its large scale and the variety of different scenarios depicted, HOI4D [220] for high-quality human-object interactions, and EPIC Kitchens [73] 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 [24]. First, we use a pre-trained visual model [271] 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 [316]. 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

$$\mathcal{L} = \lambda_\mu ||\mu - \hat{\mu}||_2 + \lambda_\theta ||\theta_{\text{wrist}} - \hat{\theta}_{\text{wrist}}||_2 + \lambda_P ||P - \hat{P}||_2 \tag{4.1}$$

At test time, we generate a crop of the object using Segment-Anything [182] 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 [336] 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 [144, 379]. For more details, we refer readers to the appendix.

Algorithm 2 Fine-Tuning Procedure for DEFT

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

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 4.3) separate from the affordance model f. We extract the task-specific wrist trajectory after the grasp using [337]. 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?

4.3.2 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

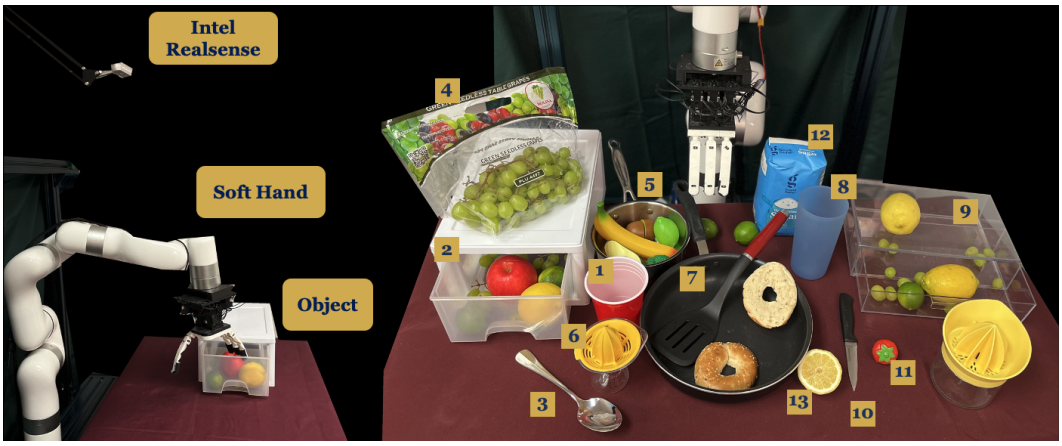


Figure 4.4: **Left**: Workspace Setup. We place an Intel RealSense camera above the robot to maintain an egocentric viewpoint, consistent with the affordance model's training data. **Right**: Thirteen objects used in our experiments.

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 4.1.

Residual policies have been used previously to efficiently explore in the real world [168, 23]. 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 4.1 for more information.

At test time, we could take the mean values of the top N trajectories for the rollout policy. However, this does not account for the appearance of different objects, previously unseen object configurations, or other properties in the environment. To generalize to different initializations, we train a VAE [382, 332, 333, 180] 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

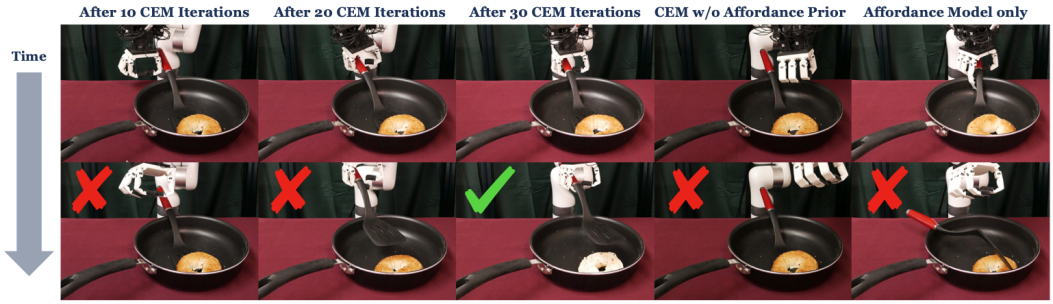


Figure 4.5: Qualitative results showing the finetuning procedure for DEFT. The model learns to hold the spatula and flip the bagel after 30 CEM iterations.

$\xi + \hat{\delta}$. Because the VAE is conditioned on the initial image, we generalize to different locations and configurations of the object.

4.4 Experiment Setup

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

Task Setup We introduce 9 tabletop tasks, *Pick Cup*, *Pour Cup*, *Open Drawer*, *Pick Spoon*, *Scoop Grape*, *Stir Spoon*, *Pick Grape*, *Flip Bagel*, *Squeeze Lemon*. Robotic hands are especially well-suited for these tasks because most of them require holding curved objects or manipulating objects with tools to succeed. For all tasks, we randomize the position of the object on the table, as well as use train and test objects with different shapes and appearances to test for generalization. To achieve real-world learning with the soft robot hand, we pretrain an internet affordance model as a prior for robot behavior. As explained in Section 4.3, 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

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 4.2: 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.

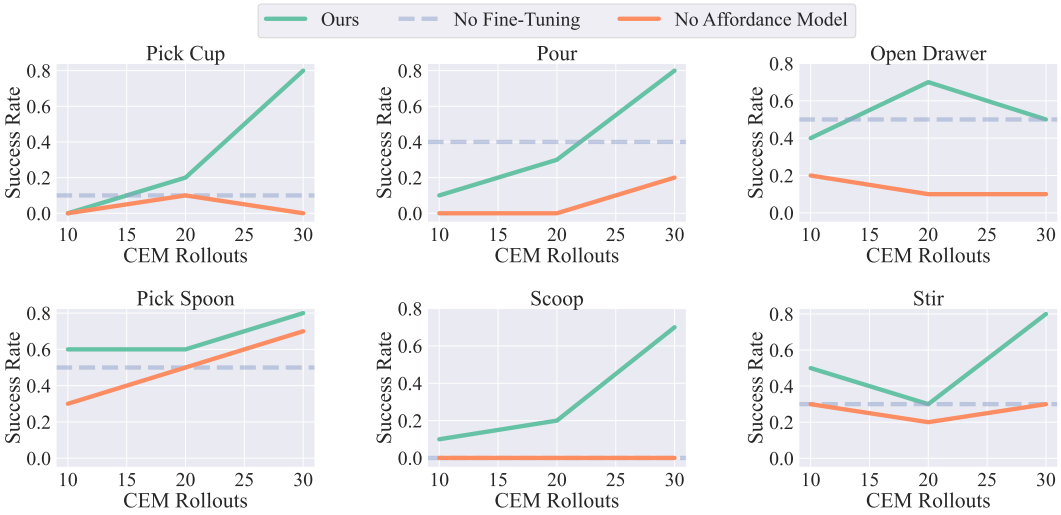


Figure 4.6: Improvement results for 6 tasks: pick cup, pour, open drawer, pick spoon, scoop, and stir. We see a steady improvement in our method as more CEM episodes are collected.

the camera frame. We calibrate the camera so that the predictions of the affordance model can be converted to and executed in the robot frame. The flexibility of the robot hand also makes it robust to collisions with objects or unexpected contact with the environment. For the arm, we ensure that it stays above the tabletop. The job will be terminated if the arm's dynamics controller senses that the arm collided aggressively with the environment.

4.5 Results

Effect of affordance model We investigate the role of the affordance model and real-world fine-tuning (Table 4.2 and Figure 4.6). 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 [182] 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 4.2). 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 4.3 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 [271] or a ResNet [154] 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 4.3). 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 [54] where in addition to the image and affordance model outputs, we condition on the reward outputs and train a transformer to predict the residual. At test time the maximum reward is queried and the output is used in the rollout. While this method performs well, we hypothesize that the transformer needs more data to match DEFT. Finally, we train a VAE to directly estimate ξ instead of the residual. This does not effectively distill the

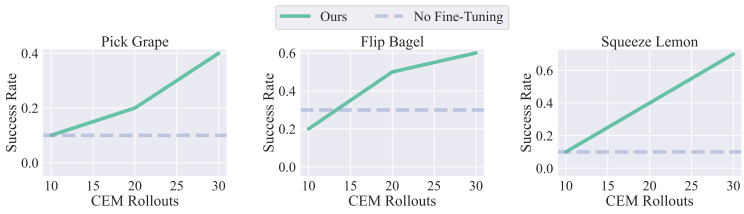


Figure 4.7: We evaluate on three difficult manipulation tasks.

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

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 4.7. 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.

4.6 Conclusion

In this chapter, 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 can 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 can 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

Learning General Purpose Policies from Human Videos

Chapter 5

Strucured World Models from Human Videos

5.1 Motivation

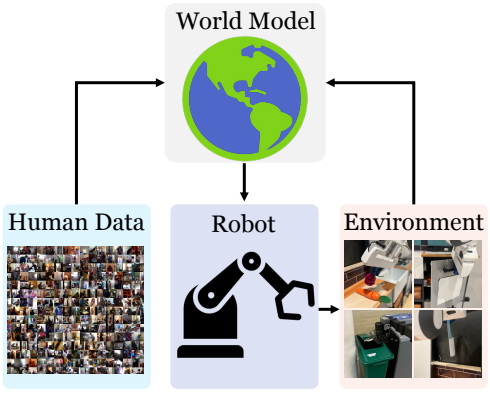


Figure 5.1: We present SWIM, an approach for learning manipulation tasks in the real world with only a handful of trajectories.

A truly useful home robot should be general purpose, able to perform arbitrary manipulation tasks, and *get better* at performing new ones as it obtains more experience. How can we build such generalist agents? The current paradigm in robot learning is to train a policy, in simulation or directly in the real world, with engineered rewards or demonstrations directly constructed for the environment. While this has shown successes in lab-based tasks [170, 303, 203], learning is heavily dependent on the structure of the reward. This is not scalable as it is very challenging to *transfer* to new tasks, with different objectives. Often, it is also difficult to obtain ground truth objectives for a task in the real world. For a robot to succeed in the wild, it must not only learn many tasks at a time but also *get better* as it sees more data. How can we build an agent that can take advantage of large-scale experience and multi-task data?

We aim to build *world models* to tackle this challenge. One key observation is that there is commonality between many tasks performed by humans on a daily basis. Even across diverse settings, the environment dynamics and physics share a similar structure. Learning a single *joint* world model, that predicts the future consequences of actions across diverse tasks can thus enable agents to extract this shared structure. While world models enable learning from inter-task data, they require action information to make predictions about the future. Furthermore, for planning in an

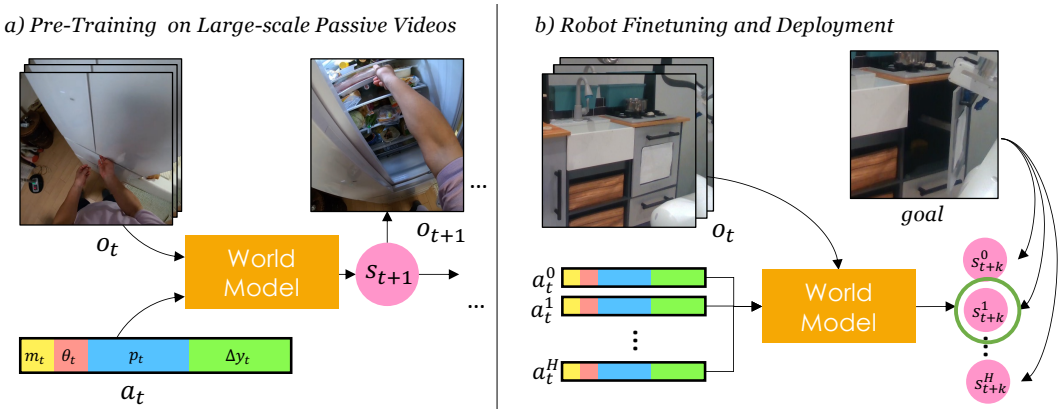


Figure 5.2: Overview of SWIM. We first pre-train the world model on a large set of human videos. We finetune this on many robot tasks, in an unsupervised manner, and deploy at test-time in the real world to achieve a given goal. Videos can be found at https://human-world-model.github.io

environment, the actions need to be relevant to the particular robot. Consequently, world models for robotics have mostly been trained only on data collected directly by a robot [98, 419, 199, 82, 99]. However, the quantity of this data is limited, which is very expensive and cumbersome to collect in the real world. Thus, the benefits of using large datasets as seen in other machine learning areas such as computer vision and language [38, 316] have not been realized for robotics, as no such dataset exists for robotics. However, there is an abundance of *human videos*, performing a very large set of tasks, on the internet. Is there a way to leverage this abundant data to learn world models for robotics, that will enable the robot to predict the consequences of its *actions* in any environment, enabling general-purpose learning? Due to the large morphology gap between robots and humans, it is challenging to obtain actions from human videos. Thus, previous approaches have mostly focused on learning visual representation features [271, 424] from observations alone. Using internet human videos to train robots requires us to define an action space that is applicable both in the human video domain and for robots. Consider the task of picking up a mug. To perform this task, the low-level signals sent to a person's arm compared to that of a robot would be completely different, and so predictive models in low-level joint space will not transfer well. If the action space instead required predicting the target pose and orientation of the mug handle, with low-level control abstracted away, then target poses used by humans could be utilized by robots as well. Thus, we learn *high-level* structured action spaces that are morphology invariant. For manipulation tasks, predicting a grasp location and post-grasp waypoints is an effective action space since it encourages object interaction. We can train visual affordance networks that produce these locations given videos leveraging techniques in computer vision [216, 123, 362, 265, 24].

We propose **S**tructured **W**orld **M**odels for **I**ntentionality (**SWIM**), which utilizes large-scale internet data to train world models for robotics using structured action spaces. Training the world model in the common high-level structured action space allows it to capture how human hands interact with objects when trying to grasp and manipulate them. This model can then be fine-tuned for robotics settings with only a handful of real-world interaction trajectories. This is because the world model can leverage the actionable representations it was pre-trained with due to the similarity in how the human hands from video data and robot grippers interact with the world. Furthermore, these interaction trajectories for fine-tuning do not require any task supervision and can be obtained simply by executing the visual affordance actions. We note that both pre-training on human videos and finetuning the world model on robot data do not make any assumption on rewards, and this *unsupervised* setting allows us to utilize data relevant for different tasks. This allows the robot to train a *single* world model on all the data, thus enabling us to train generalist agents. In our experiments, we show that we can train such joint world models through two distinct robot systems operating in real-world environments. Finally, we can deploy the fine-tuned world model to perform tasks of interest by specifying a goal image. The world model then plans in the affordance action space to find a sequence of actions to manipulate objects as required by the task.

To summarize, SWIM trains world models for robot control and consists of three stages: 1) Leveraging internet videos of human interactions for pre-training the model, 2) Finetuning the model to the robot setting using reward-free data, 3) Planning through the model to achieve goals. We evaluate this framework on two robot systems – a Franka Arm, and a Hello Stretch robot. SWIM is able to learn directly, is trained on data from multiple settings and gets better with data from more tasks. We perform a large-scale study across multiple environments and robots and find that SWIM achieves higher success ($\sim$ 2X) than prior approaches while being very sample efficient, requiring less than 30 minutes of real-world interaction data.

5.2 Related Work

Efficient Real World Robot Learning Deploying learning-driven approaches on hardware is challenging and requires either large engineering efforts to collect demonstrations [36, 165], many hours of autonomous interactions [170, 171], or simulations [14, 402, 191]. A major constraint of continuous control is the extremely large action space. Prior methods have focused on reducing this search space by using skills or options in a hierarchical manner [71, 288, 21, 396, 75, 284, 75], physical inductive biases [258, 300, 187, 389, 185, 25, 244]. It is also possible to visually ground the action space, by parameterizing each observed location by a 2D [442, 443, 371, 164] or 3D [372] action. While these can speed up learning, we find that our structured action space, based on human-centric visual affordances allows

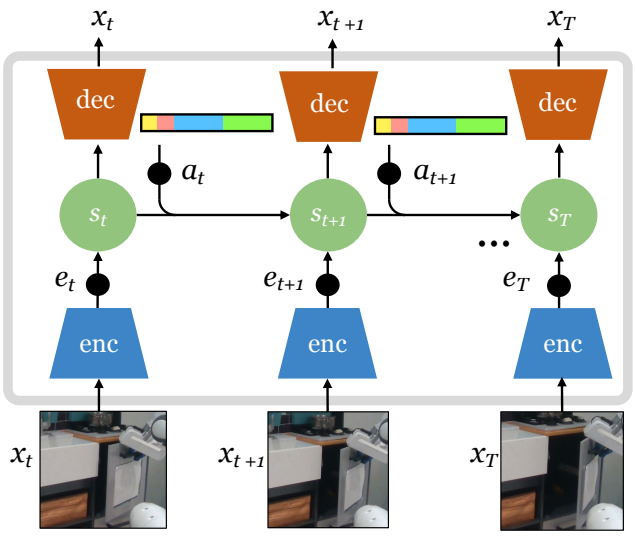


Figure 5.3: World Model Training: Images and actions are encoded into a learned feature space that has temporal structure, following the approach from [140]

us to not only perform manipulation efficiently but also leverage out-of-domain human/internet videos.

Model-based learning To tackle the sample efficiency problem in robot learning, prior methods have proposed learning dynamics models, which can later be used to optimize the policy [87, 89, 138, 64, 263, 264]. Such approaches mostly operate and learn in state space, which tends to be low dimensional. In order to deal with the highly complex visual observations from real-world settings, prior methods have used *World Models* [134], which capture dynamics of the *agent* and its *environment*. Such models can plan in image space [105, 98] or fully in imagination space [417, 137, 139, 198]. Such world models have been shown to be useful on a large set of tasks [329], including on hardware [420]. We argue that world models can be helpful in modeling the real world, especially if they can understand how the environment will behave at a high level and model the intentions of the agent.

Visual and Action Pre-Training for Robotics In order to learn more generalizable and actionable representations, prior methods have learned visual encoders from large-scale human video data, either via video-language contrastive learning [271] or through inpainting masked patches [424, 318]. These representations have been shown to be useful for dynamics models as well [146]. Such approaches focus on the visual complexity of the world but do not encode any behavior information. Some works have incorporated low-level actions from human videos into the learning loop [239, 313, 17, 369], but these are fixed for a specific morphology and use a direct mapping to the robot. In contrast, our approach is able to learn a world model from human videos, incorporating action information, and works in multiple settings.



Figure 5.4: We evaluate SWIM on six different real-world manipulation tasks on two different robot systems (shown on the left). On the right, we show a sample of the visual affordances from the visual affordance model $\mathcal{G}_\psi$.

5.3 Background and Preliminaries

World Models These are used to learn a compact state space for control given high-dimensional observations like images. The learned states preserve temporal information, which enables effective prediction and planning [134, 353, 352]. In this work, we use the model structure and training procedure from Dreamer [138, 137, 140], which has the following components:

$$\begin{aligned} \text{encoder:} &\quad e_t = \text{enc}_\phi(x_t) \\ \text{posterior:} &\quad p(s_t|s_{t-1}, a_{t-1}, e_t) \\ \text{dynamics:} &\quad p(s_t|s_{t-1}, a_{t-1}) \\ \text{decoders:} &\quad p(x_t|s_t), p(r_t|s_t) \end{aligned}$$

Here x_t, a_t, r_t denote the observation, action, and reward at time t, and s_t denotes the learned state space. Note that all these components are parameterized using neural networks. The model is trained by optimizing the ELBO as described in Dreamer, where the learned features are trained to reconstruct images and rewards and are regularized with a dynamics prior. The reward head decoder is not trained if r_t is not provided. For more details, we refer the readers to [140].

Hand-Object Interactions from Human Videos In this work, leverage human videos to learn world models. Throughout the chapter, we will refer to a set of visual affordances. These visual affordances comprise of the hand trajectory h_t in image space (normalized to a 0-1 range), and object locations (o_t). We obtain human

hand-object information (h_t, o_t) for each frame using the 100 Days of Hands [362] detector model, trained on many hours of youtube videos. These can then be used to identify where on the object the hand makes contact p^g, and we sample the hand position from a later frame in the video to obtain p^{pg}. Here p^g and p^{pg} denote the grasp and post-grasp pixel respectively and specify the visual affordance space.

5.4 World Models from Human Videos

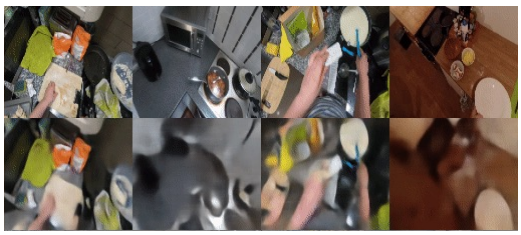
5.4.1 Visual Affordances as Actions

One of the key challenges is defining what the actions should be from human videos, most of which just contain image observations. Action information is essential for world models since they are required to learn dynamics and make predictions about the future. Furthermore, we need to define actions in a manner that is *transferable* from the human video domain to robot deployment settings. Following previous work that studies human-to-robot transfer for manipulation [23, 364, 50, 425, 381, 369, 367, 359, 439], we use the human hand motion in the videos to inform the action space. This is because we are focused on performing manipulation tasks, and how humans interact with objects using their hands contains useful information that can be transferred to robot end-effectors.

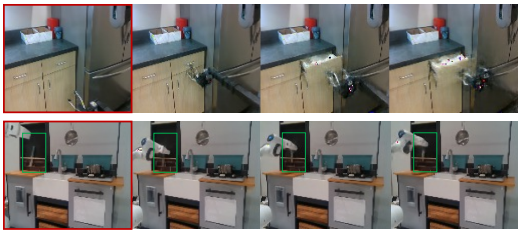
Structured Actions from Videos We note that the videos of humans interacting with objects often consist of the hand moving to a point on the object, performing a grasp, and then manipulating the object. After obtaining the grasp pixel p^g and post-grasp pixels p^{pg}, using computer vision techniques similar to [123, 216, 265] from the video clip, we use these to train $\mathcal{G}_\phi$, which distills these labels into a neural network model conditioned on the first frame of the video clip. This model thus learns affordances associated with objects in the scene, by modeling how humans interact with them. This follows the affordances described in [24], but our work can also be combined with other affordance-learning approaches.

Transfer to Robot Scene When dealing with 2D images, there is an inherent ambiguity regarding depth, which is required to map to a 3D point. To overcome this, we utilize depth camera observations to obtain the depth d_t^g at the image-space point p_t^g, and also sample the post-grasp depth d_t^{pg} within some range of the environment surface. This can then be projected into 3D coordinates in the robot frame, using hand-eye calibration, and the robot can attempt to grasp and manipulate objects by moving its gripper to these locations. The affordance action at time t can thus be expressed as $u_t = [p_t, d_t]$, where d_t is the depth corresponding to pixel p_t.

Hybrid Action Space While visual affordances help structure the action space to increase the likelihood of useful manipulation and allow us to learn from human video, they impose restrictions on the full space of end-effector motion. Hence, we adopt a hybrid action space that has the option to execute both the aforementioned



(a) World Model pre-training reconstructions



(b) Imagination rollouts at deployment

Figure 5.5: a) World Model pre-training reconstructions on Epic-Kitchens dataset [73]. b) Model imagination rollouts for high-reward trajectories. We can see that SWIM can imagine plausible and successful trajectories, for both human and robot data. The first image (highlighted in red) is the original observation by the robot.

visual affordance, as well as arbitrary end-effector Cartesian actions. We append a mode index to denote which type of action should be executed. This enables the robot to benefit both from the structured pixel-space visual affordance actions and the pre-training data in mode (m) 0, and make adjustments using arbitrary end-effector delta actions in mode 1. An action can be described by the following:

$$a_t = [m_t, \theta_t, u_t, \Delta y_t] \tag{5.1}$$

Here m_t denotes the mode, θ_t is the rotation of the gripper, u_t is the image-space action ($u_t = [p_t, d_t]$, where p_t are pixel coordinates in the image and d_t is depth), and Δy_t is the Cartesian end-effector action. At a particular timestep, only one out of the image action and Cartesian actions can be executed. If $m_t = 0$, this corresponds to the affordance mode, and so p_t is executed. If $m_t = 1$, then the robot is operating in the Cartesian control mode, and Δy_t is used. Due to our hybrid action space, we can seamlessly switch between training with the visual affordance and Cartesian end-effector action spaces. This allows the robot to leverage the structure from human video and also make adjustments if required using Cartesian actions which are useful for fine-grain control.

Algorithm 3 Human Video Data Training

Require: Human Video Dataset $\mathcal{D}$
  initialize: World model $\mathcal{W}$, Affordance model $\mathcal{G}$
  Process $\mathcal{D}$ into video clips $C^0, ...C^T$
  Obtain grasp p^g and post grasp p^{pg} pixels for each C^k.
  Create actions a_t using eq. 5.1, with mode $m_t = 0$, and randomly sampling depth d_t and rotation θ_t
  Train $\mathcal{G}_\phi(a^g, a^{pg}|I_0^k)$, where I_0^k is the first frame of C^k
  Train $\mathcal{W}$ on trajectory sequences $\{(I_0^k, a^g, I_{t1}^k, a^{pg}, I_{t2}^k)\}$
  return $\mathcal{W}, \mathcal{G}$

	Cabinet	Veg	Knife	Drawer	Dishwasher	Can	Average
No world model:							
BC-Affordance	0.32	0.48	0.16	0.56	0.20	0.44	0.36
BC-Pix	0.16	0.40	0.00	0.24	0.08	0.12	0.17
No human-centric affordance-based actions:							
MBRL-single [140]	0.00	0.28	0.20	0.00	0.04	0.00	0.09
MBRL-Pix-single	0.52	0.36	0.16	0.00	0.04	0.04	0.19
No pre-training from human videos:							
MBRL-Affordance-single	0.68	0.16	0.40	0.84	0.20	0.36	0.44
MBRL-Affordance-joint	0.12	0.36	0.36	0.08	0.20	0.04	0.19
SWIM	0.84	0.76	**0.72**	0.92	**0.84**	**0.68**	**0.79**
SWIM-single	**0.88**	**0.80**	0.60	**0.96**	0.68	0.56	0.75

Table 5.1: Success rates of SWIM and baselines on six different manipulation tasks, over 25 trials.

5.4.2 Structured Affordance-based World Models for Robotics

The overall approach is outlined in Alg. 4. We now describe each of the three phases - 1) World model pre-training on human videos, 2) Unsupervised finetuning with robot data, and 3) Robot deployment to perform a task given a goal image.

Training from Passive Human Videos We first use a large set of human videos, obtained from Epic-Kitchens [73] to both train the world model $\mathcal{W}$, and obtain the visual affordance model $\mathcal{G}_\phi$. This dataset includes around 50k egocentric videos of people performing various manipulation tasks in kitchens. We first process this dataset into a set of short video clips (around 3 seconds). After obtaining the grasp pixel p^g and post-grasp pixels p^{pg} from the video clip, we convert them to our action space (specified in eq. 5.1), and train $\mathcal{G}_\phi$, as previously described in section 5.4.1. For video clip k, let I_t^k denote an image frame from the clip at time t. We collect images $I_{t_1}^k$ and $I_{t_2}^k$, where t_1 is the time of the grasp, and t_2 is when the hand is at p^{pg}. $\mathcal{W}$ is then trained on the trajectory sequences:

$$\{I_0^k, a^g I_{t_1}^k, a^{pg} I_{t_2}^k\} \tag{5.2}$$

This procedure is outlined in Alg. 3. As described in section 5.4.1, there are two modes for the actions - either in pixel space or end-effector space. In order to train on human videos, we consistently set $m_t = 0$ and thus use the image space actions. Since image depth and robot rotation information are not present in the video, we randomly sample values for these components. We include visualizations of the world model predictions on the passive data in Figure 5.5, and see that the model is able to capture the structure of the data.

Algorithm 4 Overview of SWIM

Get $\mathcal{W}, \mathcal{G}$ = Human Video Data **Pre-Training** (Alg. 3)
Finetuning: Query $\mathcal{G}$ for N_0 iterations to collect robot dataset $\mathcal{R}_\mathcal{D}$ to train $\mathcal{W}$.
Task Deployment: (Given goal I_g)
Rank trajs in $\mathcal{R}_\mathcal{D}$ using I_g. Fit GMM g.
for traj 1:K **do**
  Query N proposals from $\mathcal{G}$, $\{a^g, a^{pg}\}_{1..N}$
  Query M proposals from g
  Select best proposal using CEM through $\mathcal{W}$
  Execute on the robot to reach I_g
end for

Finetuning with Robot Data To use the world model $\mathcal{W}$ for control, we need to collect some in-domain robot data for finetuning. We do so by running the visual affordance model $\mathcal{G}$ to collect a robot dataset R_D, which is then used to train $\mathcal{W}$. We emphasize that this step does not require *any* supervision in the form of task rewards or goals. Hence, we can collect data from diverse tasks in the finetuning step. We see in Fig. 5.8 that SWIM enables the world model to pick up on the salient features of the robot environment very quickly as compared to models that do not use pre-training on human videos.

Task Deployment After the world model has been fine-tuned on robot domain data, it can be used to perform tasks specified through goal images. The procedure for doing so is outlined in the Task Deployment section in Alg. 4. We collect two sets of action proposals. The first set is obtained by querying the visual affordance model $\mathcal{G}$ on the scene. We also want to leverage our knowledge of trajectories in $\mathcal{R}_\mathcal{D}$ that reach states close to the goal. For this, we create a second set of proposals by fitting a Gaussian Mixture Model to the top trajectories in $\mathcal{R}_\mathcal{D}$ and sampling from it. We then use the world model to optimize for an action sequence using the standard CEM approach [341], where the initial set of plans is set to be the combined set of action proposals. Ranking the trajectories in $\mathcal{R}_\mathcal{D}$ and running CEM requires rewards, and we can obtain this by measuring the distance to the goal in the world model feature space:

$$r_t = \text{cosine}(f_\mathcal{W}(I_g), f_t)$$

where f_t is the world model feature, and $f_\mathcal{W}$ is the learned feature space of the model. For ranking trajectories in $\mathcal{R}_\mathcal{D}$, $f_t = f_\mathcal{W}(I_k)$ for image k in the dataset. For planning, f_t corresponds to the predicted feature state. In our experiments, we use cosine distance to goal in the feature space from [271] to provide reward for model-free baselines, since they do not have a model, and so we also add this term to our reward by training a reward prediction head to get feature space [271] distance to goal

from f_t. In our experiments we run an ablation where we use only the world model feature space, and find that performance for our approach is about the same.

5.5 Experimental Setup

5.5.1 Environments

Our real-world system consists of two different robots, evaluated over six tasks. Firstly, we use the Franka Emika arm, with end-effector control. This robot acts in a play kitchen environment with multiple tasks that mimic a real kitchen. Specifically, the robot needs to open a cabinet, pick up one of two toy vegetables from the counter and lift a knife from a holder. Note that the knife task is very challenging as it requires fine-grained control from the robot. In order to test SWIM in the wild we also deploy it on a mobile manipulator, the Stretch RE-1 from Hello-Robot. This is a collaborative robot designed with an axis-aligned set of joints and has suction cups as fingertips. We run this robot in real-world kitchens to perform different tasks, including opening a dishwasher, pulling out a drawer, and opening a garbage can. The garbage can task is challenging as the area for the robot to grasp onto is quite small. We show images of the environments in Figure 5.4.

5.5.2 Baselines and Ablations

In order to compare different aspects of SWIM, we run an extensive of baselines and ablations. All world-model-based approaches directly use code from Dreamer [140].

- `MBRL-Affordance:` An important contribution of SWIM is pre-training on human videos. This baseline is similar to SWIM but does not use any human video pre-training, allowing us to test our hypothesis that using human video is important for learning a generalizable world model.

- `MBRL-Pix:` Secondly, we would like to test how much the affordance action space helps the robot. This approach uses the same world model control procedure as SWIM, but does not sample actions using the visual affordance, $\mathcal{G}_\psi$. Instead grasp and post grasp locations are randomly sampled from an image crop around the object.

- `MBRL:` This baseline further removes structure from the action space, and only uses cartesian end-effector actions, without any pixel-space structure, thus $m_t = 1$ (described in section 5.4.1) for every timestep t. In order to help with sample efficiency, we use a simple heuristic to bootstrap this baseline: we initialize the robot each episode near the center of the detected object, using Detic [453], a state-of-the-art object detector.

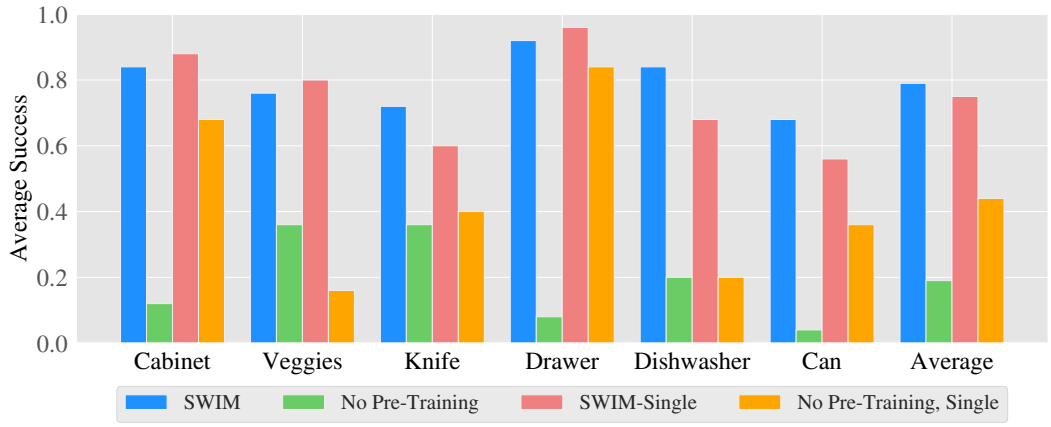


Figure 5.6: Comparison of SWIM and `MBRL-Affordance` for both the single task and jointly trained model. We see a large drop in success when removing pre-training on human videos, especially when dealing with diverse robot tasks.

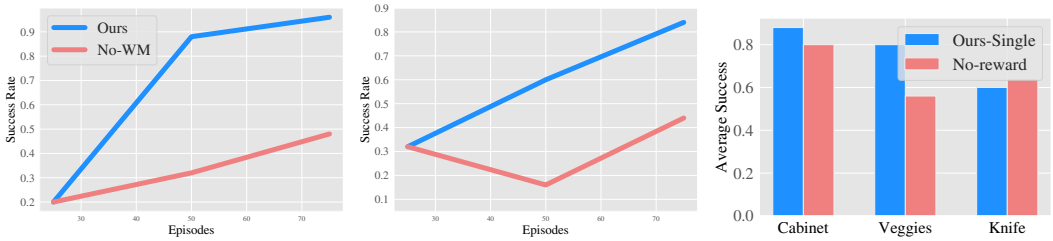
- `BC-Affordance:` We would like to test if using world models is critical to performance, or if a simple behavior cloning approach can be effective. This baseline employs a filtered-behavior cloning [293, 299] strategy, in which the top trajectories based on reward (in our case distance to goal) are selected. Since there is no learned world model, we use distance in the feature space from the R3M model [271]. After selecting the top trajectories, we fit a gaussian mixture model and sample from it to obtain actions. These are in the same visual affordance action space used by our approach.

- `BC-Pix:` Uses behavior cloning in the same way as `BC-Affordance`. The only difference is the action space - this approach randomly samples locations and does not use $\mathcal{G}_\psi$ to obtain actions.

5.5.3 Implementation details

Human Video Data Pre-trainig In order to pre-train the world model on human videos, we use the Epic-Kitchens [73] dataset. The dataset is divided into many small clips of humans performing semantic actions. We use the 100 Days of Hands [362] detector to find when an object has been grasped and find post grasp waypoints. Around 55K such clips are used to train the world model. Since we do not have depth or 3D information available, we randomly sample θ_t and the depth component of the image space action p_t.

Affordance Model We show some qualitative examples of the affordances of the human-affordance model ($\mathcal{G}_\psi$) we use in Figure 5.4. This model has a UNet style encoder-decoder architecture, with a ResNet18 [154] encoder. The final output of



(a) Cabinet (b) Knife (c) No-Reward

Figure 5.7: Continuous improvement (a-b): We see that SWIM continues to improve, achieving high success. (c) Ablating the need for external feature space goal distance at test time.

the model is h_t and g_t, where g_t is a set of keypoints obtained from a spatial softmax over the network's heatmap outputs, representing the grasp point, and h_t is the post-grasp trajectory of the detected hand.

World Model We use the world model from [140]. However, in order to handle high-dimensional image inputs we employ NVAE [405] as a stronger visual encoder. While not necessary to train the reward model q_r when finetuning, we empirically found that it added stability to the filtering setup a test time. We leave distilling the latent features into a neural distance function as future work.

Robot Deployment Setup To capture videos and images we use an Intel Realsense D415, to get RGBD images. For each task we collect either 25 or 50 iterations of randomly sampled actions (in human-affordance, random image or Cartesian space), which takes about 30 minutes, finetuning the model on collected data. We obtain feature distance w.r.t. to image goals using the ResNet18 encoder from [272]. We sample around 2K action proposals and use the output of $\mathcal{W}_\phi$ to prune these. The model outputs are then evaluated (25 times). A human measures success based on a pre-defined metric (i.e. the cabinet should be fully open, etc).

5.6 Results

In our experiments we ask the following questions (i) Can we train a *single world model jointly* with data coming from diverse tasks? (ii) Does training the world model on *human video* data help performance? (iii) How important is our structured action space, based on human visual *affordances*? (iv) Are *world models* beneficial for learning manipulation with a handful of samples? (v) Can our approach *continually improve* performance with iterative finetuning?

We present a detailed quantitative analysis of various approaches in Table 5.1. We see that across environment settings and robots SWIM achieves an average success rate of about 80% when using joint models (trained separately for Franka and Hello robot tasks). We also observe strong performance when SWIM is trained on individual

tasks, getting an average success of 75%, compared to the next best approaches which only get around 40% success.



Figure 5.8: Image reconstruction using world model features in early training stages for SWIM and `MBRL-Affordance` (which has no pre-training), showing that SWIM can effectively transfer representations from human videos. Note that for our experiments we use models trained to convergence.

Joint World Model A big benefit of SWIM is that it can deal with different sources of data. `SWIM-single` employs a model trained individually for each task. We see that overall the performance improves when sharing data, from the last two rows of Table 5.1. This is likely because there are some similarities across tasks that the model is able to capture. We find this encouraging and hope to scale to more tasks in the future. Further, we see that for the best baseline, `MBRL-Affordance-single`, using all the data jointly to train the world model leads to a major drop in performance (from 44% to 19%). We show this visually in the bar chart in Figure 5.6, where the effect of pre-training a model on human videos is amplified when dealing with all of the data from all the tasks. This shows that when dealing with a large set of tasks and diverse data, it is crucial to incorporate human-video pre-training for better performance and generalization. Hence we do not run joint world model experiments for `MBRL` and `MBRL-Pix` since the performance is already quite low (around 10 - 20 %) when the model is trained just on single task data.

Human Video Pre-Training As noted in the previous section, pre-training on human videos is *critical* to being able to effectively train joint world models on multi-task data, as seen in Figure 5.6. For `MBRL-Affordance` we saw that in many cases the model collapses quickly to a sub-optimal control solution when trained on multiple tasks jointly. To investigate this further, we visualize the image reconstructions from $\mathcal{W}$ within the first minute of training and find that the outputs of SWIM were already very realistic, as compared to those of `MBRL-Affordance`. This can be seen

in Figure 5.8, where the outputs of `MBRL-Affordance` are very pixelated while those of SWIM already capture important aspects of the ground truth, indicating the usefulness of pre-training on human videos.

Human-Affordance Action Space How does the choice of action space affect performance? For this we compare the (single task) model based and BC approaches separately. Comparing `MBRL-single` and `MBRL-Affordance-single` in Table 5.1, we can see that there is a clear benefit in using structured action spaces, with over 5X the success compared to cartesian end effector actions. This fits our hypothesis, as it is very difficult for methods that use low-level actions to find successes in a relatively small number of interaction trajectories. The few successes that `MBRL-single` does see are due to the initialization of the robot close to the object using Detic. Furthermore, we note that this benefit is not simply because the affordance actions are in image space. In both the filtered-BC and world model case, the success rate with the affordance action space is roughly *double* than that of acting in pixel space, where target locations are sampled from a random crop around the object. This shows that picking the right action space *and* acting in a meaningful way to collect data can bootstrap learning and lead to efficient control.

Role of World Model How important is using a world model, and can we achieve good performance by just using the affordance action space? From Table 5.1, we see that the average performance of `BC-Affordance` is not too far behind that of `MBRL-Affordance-single`. However, without a world model, the controller cannot leverage *multi-task* data, both for the pre-training stage to use human videos and for learning the shared structure across multiple robot domains by training a joint world model. Due to these critical reasons discussed previously, SWIM outperforms the best filtered-BC approach by more than a factor of 2.

Continual Improvement Next, we investigate if SWIM can keep improving using the data that it collects when planning for the task. Since the world model can learn from all the data, we want to test if it can improve its proficiency on the task. Thus, after evaluating $\mathcal{W}$ once, we retrain on the newly collected data (as well as the old data), and re-evaluate the model. We present the learning curves in Figure 5.7. We see that SWIM is able to effectively improve performance, and achieves success of over 90%, which is far better than the performance of `BC-Affordance` even after continual training. This is an encouraging sign that SWIM can scale well since it can keep improving its performance with more data to continually learn. In the future, we hope to not only continually finetune, but also add new tasks and settings.

Reward Model In Figure 5.7 c) we examine the effect of removing the reward prediction module on planning and find that only using distance in world model feature space is fairly competitive with using both the feature distance and predicted reward. We hypothesize that for the veggies task, it was harder to estimate reward accurately because the free objects tend to move around a lot during training, thus it might take more samples to learn consistent features.

5.7 Conclusion

In this chapter, we present SWIM, a simple and efficient way to perform many different complex robotics tasks with just a handful of trials. We aim to build a single model that can learn many tasks, as it holds the promise of being able to continuously learn and improve. We turn to a scalable source of useful data: human videos, from which we can model useful interactions. In order to overcome the morphology gap between robot and human videos, we create a structured action space based on human-centric affordances. This allows SWIM to pre-train a world model on human videos, after which it is fine-tuned using robot data collected in an unsupervised manner. The world model can then be deployed to solve manipulation tasks in the real world. The total robot interaction samples for the system can be collected in just 30 minutes. Videos of SWIM can be found at https://human-world-model.github.io. While SWIM provides a scalable solution and shows encouraging results, some limitations are in the types of actions and tasks that can be performed, as they currently only include quasi-static setups.

Chapter 6

Actionable Representations Via Affordances

6.1 Motivation

A truly generalist robotic agent will need to acquire diverse manipulation skills (ranging from block stacking to pouring) that work with novel objects and are robust to realistic environmental disturbances (e.g., lighting changes, small camera shifts). Due to the scale of this challenge, the field has trended towards learning these agents directly from data [204, 304] (i.e., robot trajectories), which is either collected by expert demonstrators or (via Reinforcement Learning [394]) autonomously by the agent itself. Unfortunately, there are innumerable objects/environments, so roboticists cannot tractably collect enough real-world demonstration data and/or design a simulator that captures all this diversity.

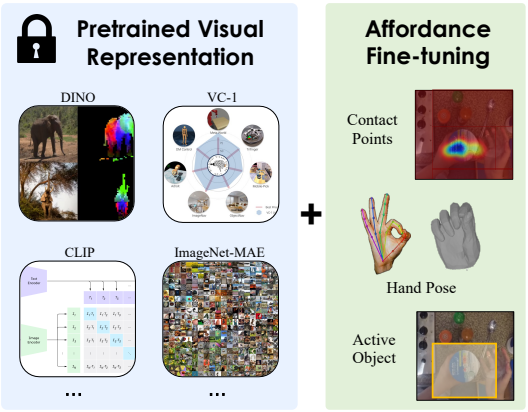


Figure 6.1: We present **HRP**, a method that mines affordances from human videos and uses them to improve self-supervised visual encoders.

One promising solution for this "data challenge" is for the robot to learn a *suitable representation* from out-of-domain data that can be transferred into the robotics domain. For example, prior work [271, 318, 235] trained self-supervised image encoders on large scale datasets of human videos (e.g., Ego4D [126]), using standard reconstruction objectives and contrastive learning [277] objectives – e.g., Masked Auto-Encoders [150] (MAE) and Temporal Contrastive Networks [357] (TCN) re-

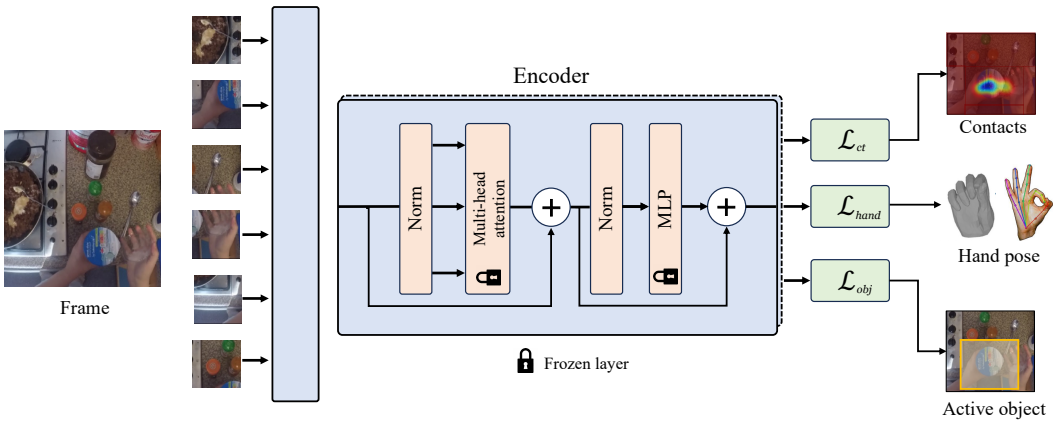


Figure 6.2: HRP fine-tunes a pre-trained encoder to predict three classes of human affordance labels via L2 regression. Specifically, the network must predict future contact points, human hand poses, and the target object given an input frame from the video stream. These affordance labels are mined autonomously from a human video dataset [126] using off-the-shelf vision detectors [362]. Representations produced by HRP are then fine-tuned to solve downstream manipulation tasks via behavior cloning.

spectively – developed by the broader learning community. After pre-training, these representations are used to initialize downstream imitation learning [350] algorithms. This formula is extremely flexible, and can substantially reduce the amount of robot data required for policy learning. However, the representations are often only effective when using specific camera views and robot setups. Furthermore, independent evaluations [84, 42] recently showed that these representations cannot improve (on average) over the most obvious baseline – a self-supervised ImageNet representation [150, 91]!

This result is surprising since robotic trajectories and human video sequences share so much common structure: both modalities contain an agent (e.g., human or robot) using their end-effector (e.g., human hand, robot gripper) to manipulate objects in its environment. Ideally, representation trained on this data would learn useful object attributes (e.g., where to grasp a mug), and spatial relationships between the end-effector and things in the environment. We hypothesize that such "actionable representations" would be much more useful (than ImageNet) for downstream robotic tasks, and that traditional, self-supervised learning objectives are unable to extract the relevant information from human video data. Our key insight is to leverage both agent (human hand motion) and environment-centric (contact and object level information) commonalities across human data.

This chapter proposes Human affordances for Robotic Pre-training (HRP), a semi-supervised pipeline to learn effective robotic representations from human video. HRP works in two stages: first, it extracts hand-object "affordance" information – i.e., which objects in the scene are graspable and how the robot should approach them – from human videos using off-the-shelf tracking models [362, 337]. These

affordances are then distilled into a pre-existing representation network (e.g., ImageNet MAE [150]), *before* the policy fine-tuning stage. This paradigm allows us to inject useful information into the vision encoder, while preserving the flexibility of self-supervised pre-training – i.e., all labels are automatically generated and the network can be easily slotted into downstream robotic policies/controllers via fine-tuning. To summarize, **we enable actionable representations by predicting hand-object interactions and hand motion from human video dataset images** (see Fig. 6.1). Our investigations and experiments reveal:

1. HRP substantially improves downstream robot performance, when applied to 6 pre-existing representations (including ImageNet [91, 150], VC-1 [235], and DINO [48]). These conclusions are backed by **over 3000 robot trials**, and replicate across 3 camera views, 3 distinct robotic setups, and 5 manipulation tasks.

2. Our ablation study reveals that our three affordance objectives (hand, object, and contact-based loss terms) are all critical for effective representation learning.

3. We find that HRP outperforms the base off-the-shelf computer vision models that we used to extract affordances.

4. We study HRP's OOD performance and find that it boosts robustness to distractor objects.

5. Our best representation, which increases performance by 20% over State-of-the-Art, will be fully open-sourced, along with all code and data.

6.2 Related Work

Representation Learning in Robotics: End-to-end policy learning offers a scalable formula for acquiring robotic representations: instead of hand-designing object detectors or image features, a visual encoder is directly optimized to solve a downstream robotic task [204]. Numerous works applied this idea to diverse tasks including bin-picking [170, 206, 304], in-the-wild grasping [130, 385], insertion [85, 204], pick-place [37], and (non-manipulation tasks like) self-driving [32, 306, 52]. Furthermore, secondary learning objectives – e.g., dynamics modeling [141, 418], observation reconstruction [269], inverse modeling [83], etc. – can be easily added to improve data efficiency. While this paradigm can be effective, learning purely from robot data requires an expensive data collection effort (e.g., using an arm farm [206, 170], large-scale tele-operation [37], or multi-institution data collection [82, 65]), which is infeasible for (most) task settings.

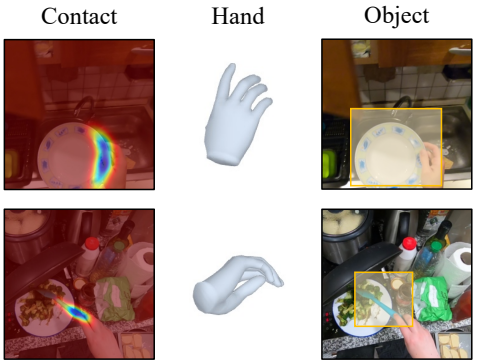


Figure 6.3: We show a summary of our affordances. From human videos, we extract contact heatmaps, hand poses, and active object bounding boxes.

To increase data efficiency, prior work applied self-supervised representation learning algorithms on out-of-domain datasets (like Ego4D [126]), and then fine-tuned the resulting representations to solve downstream tasks with a small amount of robot data – e.g., via behavior cloning on a ≤ 50 expert demonstrations [271, 235, 318], directly using them as a cost/distance function to infer robot actions [232, 411], or directly pre-training robot policies from extracted human actions. [369, 239, 173]. While this transfer learning paradigm can certainly be effective, it is unclear if these robotic representations [235, 271, 318] actually provide a substantial boost over pre-existing vision baselines [84, 42], like ImageNet MAE [150] or DINO [48]. One potential issue is that roboticists often use the same exact pre-training methods from the vision community, but merely apply them to a different data mix (e.g., VC-1 [235] applies MAE [150] to Ego4D [126]). Thus, the resulting representations are never forced to key in on actionable information in the scene. We propose a simple formula for injecting this information into a vision encoder, using a mix of hand and object affordance losses, which empirically boost performance on robotic tasks by 25%.

Affordances from Humans: HRP is heavily inspired by the *affordance learning* literature in computer vision [119, 118]. These works use human data as a probe to learn environmental cues (i.e., affordances) that tell us how humans might interact with different objects. These include physical [100, 26, 131, 459, 149, 451, 260] and/or semantic [340, 346] scene properties, or forecast future poses [186, 334, 115, 160, 156, 410, 4, 195, 409, 126, 114, 245, 120] Affordances can also be learned at object or part levels [460, 112, 123, 265, 216, 433]. Usually such approaches leverage human video datasets [126, 73, 81, 74] or use manually annotated interaction data [219, 77, 362]. In addition to these cues, robotic affordances must consider how to move before and after interaction [24, 172]. A simple, scalable way to capture this information is by detecting these cues from human hand poses in monocular video streams [412, 172, 337, 226], which show robots reaching for and manipulating diverse, target objects. Our method combines these three approaches to create a human affordance dataset automatically from human video streams. The labels generated during this process are distilled into a representation and used to improve downstream robotics task performance.

6.3 Preliminaries

6.3.1 Visual Representation Learning

Our goal is to learn a visual encoder network f_θ that takes an input image I and processes it into a low-dimensional vector $f_\theta(I) \in \mathcal{R}^d$. This resulting "embedding vector" would ideally encode important scene details for robotic policy learning – like the number and type of objects in a scene and their relationship to the robot end-effector. In this chapter, f_θ is a transformer network (specifically ViT-B [445], with patch size 16 and $d = 768$) parameterized with network weights θ. But to be clear, all our methods are network architecture agnostic.

Self-Supervised Learning: The computer vision community has broadly adopted *self-supervised* representation learning algorithms that can pre-train network weights without using *any* task-specific supervision. This can be accomplished using a *generative learning objective* [95], which trains f_θ alongside a decoder network D that reconstructs the original input image input from the representation. Another common approach is *contrastive learning* [277, 152], which optimizes f_θ to maximize the mutual information between the encoding and the input image (i.e., place "similar" images closer in embedding space). In practice, these methods can learn highly useful features for downstream vision tasks [150, 152], but struggle to key in on actionable features required for robot learning [84, 42]. Our goal is to inject these features into an existing self-supervised network, with an affordance-driven fine-tuning stage.

6.3.2 Extracting Affordance Labels from Human Data

Before we can do any fine-tuning, we must first curate a suitable human affordance dataset $\mathcal{D}_H$. Thankfully this task can be done automatically using off-the-shelf vision modules, applied to a set of $150K$ human-object interaction videos from Ego4D (originally sampled by R3M [271]). Each video V contains image frames $V = \{I_1, \ldots, I_T\}$ that depict human hands performing tasks and moving around in the scene. From these images, we obtain **contact locations**, **future hand poses**, and **active object labels** (examples in Fig. 6.3) that capture various agent-centric properties (how to move and interact) and environment centric properties (where to interact) at multiple scales, i.e. contact-level and object-level. The following sections detail how each of these labels were generated.

Contact Locations: To extract contact locations for an image I_t (with no object contact), we find the frame $I_j; j > t$ where contact with a given object will begin, using a hand-object interaction detection model [362]. Then, we use I_j to find the active object O_j and the hand mask M_j. The points intersecting M_j and O_j (acquired via skin segmentation) are our contact affordances (C_j). To account

for motion between I_t and I_j, we compute the homography matrix between the frames and project those points forward. This is done using standard SIFT feature tracking [452]: $C_t = H_{j,t}C_j$. *In other words, the contact locations denote where in I_t the human will contact in the future.* Note that there could be a different number of points for each contact scenario, which is non-ideal for learning. Thus, we fit a Gaussian Mixture Model with $k = 5$ modes on C_t to make a uniform contact descriptor – defined as the means c_t of the mixture model.

Future Hand Poses: This affordance label captures how the human moves next (e.g., to complete a task or reach an object), as the video V progresses. Given a current frame I_t, we detect the human hand's 2d wrist position (h_{t+k}) in a future frame I_{t+k}, where usually $k = 30$ (empirically determined). This is done using the Frank Mocap [337] hand detector. To correctly account for the human's motion, these wrist points are back-projected (again using the camera homography matrix) to I_t to create the final "future wrist label," $h_t = H_{t+k,t}h_{t+k}$.

Active Object Labels: In a similar manner to the contact location extraction, we run a hand-object interaction detection model [362] on V to find the image where contact began I_c. The same detector is used to find the four bounding box coordinates of the object that is being interacted with, which we refer to as the "active object." These coordinates b_c are then projected to every other frame I_t, using the homography matrix (see above). This results in an active bounding box b_t for each image in V.

6.4 Introducing HRP

We now present HRP, a simple and effective representation learning approach that injects actionable priors into a self-supervised network, f_θ, using an automatically generated human affordance dataset, $\mathcal{D}_H$ (see above for definitions and dataset mining approach). HRP is illustrated in Fig. 6.2, and the following sections describe its implementation in detail.

6.4.1 Learning Actionable Representations

The initial network f_θ is fine-tuned using batches sampled from the human dataset: $(I_t, c_t, h_t, b_t) \sim \mathcal{D}_H$, where c_t, h_t, and b_t are contact, hand, and object affordances corresponding to image I_t (see Sec. 6.3.2 for definitions). Some frames may not include all 3 affordances, so we include 3 mask variables – $m_t^{(c)}, m_t^{(h)}, m_t^{(b)}$ – so the missing values can be ignored during training. We add 3 small affordance modules – p_c, p_h, p_b – on top of f_θ that are trained to regress the respective affordances for I_t. This results in the following three loss functions:

$$\mathcal{L}_{\text{ct}} = ||c_t - p_c(f_\theta(I_t))||_2 \tag{6.1}$$

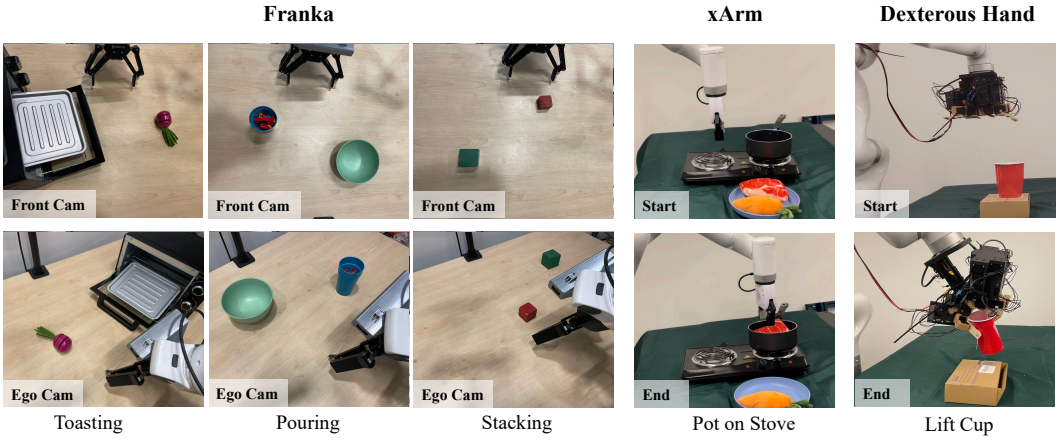


Figure 6.4: Our experiments consider 5 unique manipulation tasks, ranging from classic block-stacking to a multi-stage toasting scenario. These tasks are implemented on 3 unique robot setups, including a high Degree-of-Freedom dexterous hand (right). The 3 camera views shown – front, ego, and side views (for xArm/dexterous hand) – are the same views ingested by the policy during test-time. Note that 3 of the tasks consider 2 unique camera views in order to test for robustness!

$$\mathcal{L}_{\text{hand}} = ||h_t - p_h(f_\theta(I_t))||_2 \tag{6.2}$$

$$\mathcal{L}_{\text{obj}} = ||b_t - p_b(f_\theta(I_t))||_2 \tag{6.3}$$

The full loss is:

$$\mathcal{L} = m_t^{(c)}\lambda_{\text{ct}}\mathcal{L}_{\text{ct}} + m_t^{(h)}\lambda_{\text{hand}}\mathcal{L}_{\text{hand}} + m_t^{(b)}\lambda_{\text{obj}}\mathcal{L}_{\text{obj}} \tag{6.4}$$

Where the λs are hyper-parameters that control the relative weight of each affordance loss. We empirically found $\lambda_{\text{obj}} = 0.05, \lambda_{\text{ct}} = 0.005, \lambda_{\text{hand}} = 0.5$ to be optimal for downstream performance (see Appendix E).

6.4.2 Implementation Details

Our affordance dataset ($\mathcal{D}_H$) is at least an order of magnitude smaller than the pre-training image dataset initially used by the baseline representation (e.g., ImageNet has 1M frames v.s. our 150K). To preserve the useful features learned from the larger pre-training distribution, we keep most of the parameters in θ fixed during HRP fine-tuning. Specifically, we only fine-tune the baseline network's normalization layers and leave the rest fixed. In the case of our ViT-B this amounts to fine-tuning only the LayerNorm parameters γ and β:

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sigma}\gamma + \beta \tag{6.5}$$

These parameters are fine-tuned to minimize $\mathcal{L}$ using standard back-propagation and the ADAM [178] optimizer.

6.5 Experimental Details

Our contributions are validated using a simple empirical formula: first, HRP is applied to each baseline model (listed below). Then, (following standard practice [271, 235, 84]) the resulting representation is fine-tuned into a manipulation policy using behavior cloning. Details for each stage are provided below, and the HRP is illustrated in Fig. 6.2.

Baseline Representations: We chose 6 representative, SOTA baselines from both the vision and robotics communities:

1. **ImageNet MAE** was pre-trained by applying the Masked Auto-Encoders [150] (MAE) algorithm to the ImageNet-1M dataset [91]. It achieved SOTA performance across a suite of vision tasks, and is the first self-supervised representation to beat supervised pre-training.

2. **Ego4D MAE** was trained by applying the MAE algorithm to a set of 1M frames sampled from the Ego4D dataset [126]. For consistency with prior work, we use the same 1M frame-set sampled by the R3M authors [271].

3. **CLIP [316]** is a SOTA representation for internet data. It was learned by applying contrastive learning [277] to a large set natural language - image pairs crawled from internet captions.

4. **DINO [48]** was trained using a self-distillation algorithm that encourages the network to learn local-to-global image correspondences. DINO's emergent segmentation capabilities could be well suited for robotics, and it has already shown SOTA performance in sim [42].

5. **MVP [318]** was trained by applying MAEs to a mix of in-the-wild datasets (100 DoH [362], Ego4D [126], etc.). The authors showed strong performance on various manipulation tasks.

6. **VC-1 [235]** was trained in a similar fashion to MVP, but used a larger dataset mix. It showed strong performance on visual navigation tasks.

Note that each baseline is parameterized with the same ViT-B encoder w/ patch size 16 (see Sec. 6.3.2), to ensure apples-to-apples comparisons.

Policy Learning: Each representation is evaluated on downstream robotic manipulation tasks, by fine-tuning it into a policy (π) using Behavior Cloning [306, 348, 339]. Note that π must predict the expert action (a_t – robot motor command) given the observation (o_t – input image and robot state): $a_t \sim \pi(\cdot|o_t)$. And π is learned using a set of *50 expert demonstrations* $\mathcal{D} = \{\tau_1, \ldots, \tau_{50}\}$, where each demonstration $\tau_i = [(o_0, a_0), \ldots, (o_T, a_T)]$ is a trajectory of expert observation-action tuples. In our

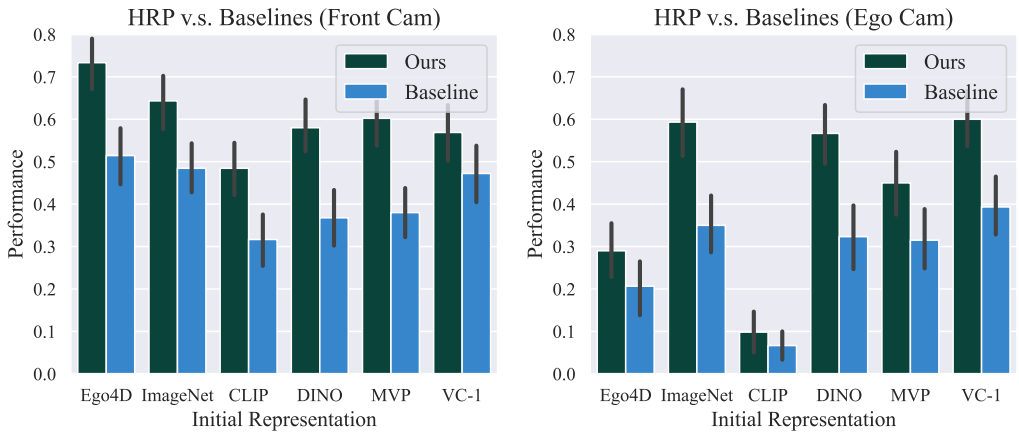


Figure 6.5: We apply HRP to 6 different baseline representations and plot performance on average across *toasting*, *pouring*, and *stacking* tasks, across two distinct views. We find that HRP representations consistently and substantially outperform SOTA baselines.

case, π is parameterized by a small 2-layer MLP (p) placed atop the pre-trained encoder $p(f(o_t))$ that predicts a Gaussian Mixture policy distribution w/ 5 modes. Both the policy network and visual encoder are optimized end-to-end (using ADAM [178] w/ $lr = 0.0001$ for 50K steps) to maximize the log-likelihood of expert actions: $\max_{p,f} log(\pi(a_t|p(f(o_t))))$. During test time actions are sampled from this distribution and executed on the robot: $a_t \sim \pi(\cdot|p(f(o_t)))$. This is a standard evaluation formula that closely follows best practice from prior robotic representation learning work [241, 84].

Real World Tasks: We fine-tune policies for each representation on the 5 diverse tasks listed below, which are implemented on 3 unique robotic setups, including a dexterous hand (illustrated in Fig. 6.4). 50 expert fine-tuning demonstrations were collected for each task via expert tele-operation. Note that the stacking, pouring, and toasting tasks were *evaluated twice using different camera views* to test robustness!

- **Stacking:** The stacking task requires the robot to pick up the red block and place it on the green block. During test time both blocks' starting positions are randomized to novel locations (not seen in training). A trial is marked as successful if the robot correctly picks and stacks the red block, and half successful if the red block is unstably placed on the green block. This task is implemented on a Franka robot, and used both an Ego and Front camera viewpoint.

- **Pouring:** The pouring task requires the robot to pick up the cup and pour the material (5 candies) into the target bowl. During test time we use novel cups and bowls and place each in new test locations. This task's success metric is

the fraction of candies successfully poured (e.g., $2/5$ candies poured $\rightarrow 0.4$ success). This task was also implemented on the Franka using Ego and Front cameras.

- **Toasting:** The toasting task requires the robot to pick up a target object, place it in the toaster oven, and shut the toaster. This is a challenging, multi-stage task. During test time the object type, and object/toaster positions are both varied. A test trial is marked as successful if the whole task is completed, and 0.5 successful if the robot only successfully places the object. This is the final task implemented on Franka w/ Ego and Front camera views.
- **Pot on Stove:** The stove task requires picking up a piece of meat or carrot from a plate and placing it within a pot on a stove. During test time, novel "food" objects are used and the location is randomized. A trial is marked as successful if the food is correctly placed in the pot. This task is implemented on a xArm and uses the side camera view.
- **Hand Lift Cup** This task requires a dexterous hand to reach, grasp, and lift up a deformable red solo up. The hand's high dimensional action space ($\mathcal{R}^{20}$) makes this task especially challenging. A trial is marked successful if the cup is stably grasped and picked. This task is implemented on a custom dexterous hand using a side camera view.

6.6 Results

Our experiments are designed to answer the following:

1. **Can HRP improve the performance of the pre-trained baseline networks (listed above)**? Does the effect hold across different camera views and/or new robots? (see Sec. 6.6.1)
2. Our affordance labels are generated using off-the-shelf vision modules – **does distilling their affordance outputs into a representation (via HRP) work better than simply using those networks as encoders?** (see Sec. 6.6.2)
3. How important are each of the three affordance losses for HRP's final performance? (see Sec. 6.6.3)
4. Is it effective to only fine-tune the LayerNorms and leave the other weights fixed? (see Sec. 6.6.3)
5. Can we qualitatively analyze how HRP improve representations? (see Sec. 6.6.4)

Note that all experiments were conducted on real robot hardware, and the models were all tested back-to-back (i.e., using proper A/B evaluation) using 50+ trials per model to guarantee statistical significance (std. err. listed on all tables/figures).

Front Cam	Teacher ResNet 100DoH [362]	HRP Models w/ Ego4D	w/ ImageNet	w/ CLIP
Toasting	$35\% \pm 15\%$	**83%** $\pm 9\%$	$75\% \pm 10\%$	$50\% \pm 11\%$
Pouring	$34\% \pm 13\%$	**60%** $\pm 11\%$	$48\% \pm 12\%$	$39\% \pm 11\%$
Stacking	0%	**77%** $\pm 10\%$	$70\% \pm 11\%$	$57\% \pm 11\%$
Average	$35\% \pm 10\%$	**73%** $\pm 6\%$	$64\% \pm 7\%$	$48\% \pm 6\%$

Table 6.1: This table compares 3 representations trained w/ `HRP` against the teacher ResNet [362] that generated our human affordance dataset (see Sec. 6.3.2). We find that the ResNet teacher under-performs even the worst `HRP` representation (fine-tuned from `CLIP`), *even after excluding the stacking task, which it failed on.*

6.6.1 Improving Representations w/ HRP

To begin, we evaluate the 6 baseline representations (detailed in Sec. 6.5) on the *toasting*, *pouring*, and *stacking* tasks using the *front camera view*. Then, we apply `HRP` to each of these baselines, and evaluate those 6 new models on the same tasks. Average success rates across all 3 tasks are presented in Fig. 6.5 (left), and the full table is in the Appendix.First, this experiment demonstrates that ImageNet MAE is still highly competitive on real-world manipulation tasks, when compared to other self-supervised representations from the vision [126, 48], machine learning [316], and robotics communities [424, 235]. Second, we show that `HRP` **uniformly boosts performance** on downstream robotics tasks – i.e., `baseline` + `HRP` > `baseline` for every baseline representation considered! Thus, we conclude that the affordance information injected by our method is highly useful for robot learning, and (for now) cannot be learned in a purely self-supervised manner.

Second Camera View: A common critique is that robotic representations perform very differently when the camera view (even slightly) changes. To address this issue, we replicated the first experiment using a radically different *ego view*, where the camera is placed over the robot's shoulder (i.e., on its "head"). While perhaps a more realistic view, it is significantly more challenging due to the increased robot-object occlusion. Average success rates are presented in Fig. 6.5 (right), and a per-task breakdown is in Appendix C.Note that our findings replicate almost exactly from the front camera view. The ImageNet MAE representation is still competitive with the other baselines, and applying `HRP` uniformly improves the baseline performance. In addition, we find that **HRP injects a higher level of robustness to camera view shifts**, when compared to the baselines. For example, we find that `ImageNet + HRP` performs the same on the ego and front camera, even though the `ImageNet` baseline clearly prefers the front cam. This general effect holds (to varying degrees) across all six baselines!

Scaling to More Robots: Finally, we verify that `HRP` representations can provide

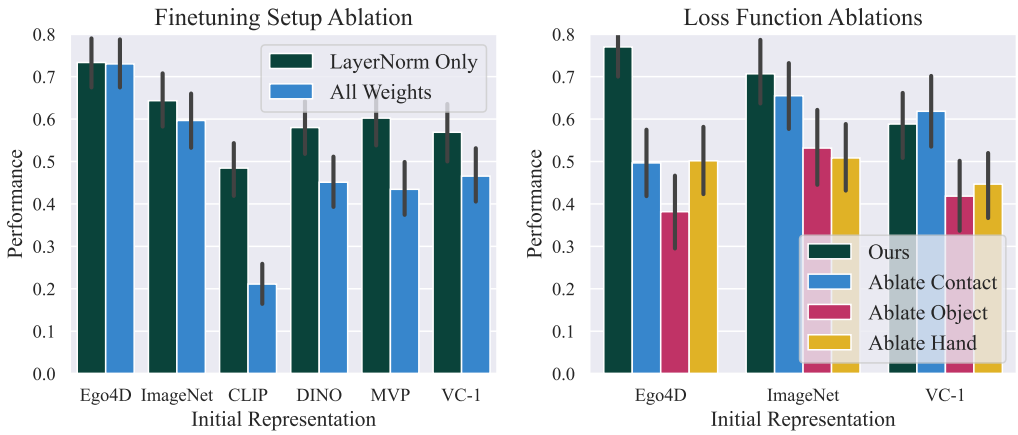


Figure 6.6: (a) applies an ablated HRP method (full fine-tuning) to the 6 baseline representations and compares their average performance v.s. standard HRP representations on the *toasting*, *pouring*, and *stacking* tasks (front cam). We find that LayerNorm only fine-tuning is almost always superior. In (b), we drop each of the 3 losses in HRP, and compare the ablated method's average performance against full HRP representations.

	Ego4D		**ImageNet**	
	w/ HRP	Baseline	w/ HRP	Baseline
Pot on Stove	**50%** $\pm$ 17%	40% $\pm$ 16%	**60%** $\pm$ 16%	40% $\pm$ 16%
Hand Lift Cup	**50%** $\pm$ 17%	40% $\pm$ 16%	**50%** $\pm$ 17%	30% $\pm$ 15%

Table 6.2: We present results of `Ego4D + HRP` and `ImageNet + HRP`, as well as the respective baselines on the x-Arm (Pot on Stove) and a dexterous hand task (Lift Cup). We see that HRP can even boost performance in multiple morphologies, including a high-degree of freedom dexterous hand [368].

benefits on other robotic hardware setups. Specifically, we compare `Ego4D + HRP` and `ImageNet + HRP` versus the respective baselines on the *Pot on Stove* (xARM) and *Hand Lift Cup* (dexterous hand) tasks. Results are presented in Table 6.2. Note that HRP representations provide a consistent and significant performance during policy learning on these radically different robot setups, which both also use a unique side camera view. This gives us further confidence in HRP's view robustness, and demonstrates that these representations are not tied to specific hardware setups, and can scale to complex morphologies like dexterous hands.

6.6.2 Distillation w/ HRP is Improves Over Label Networks

It is clear that applying HRP to self-supervised representations results in a consistent boost. However, the hand, object, and contact affordance labels for HRP themselves come from neural networks (see Sec. 6.3.2) – specifically we use the ResNet-101 [154] detector from 100DoH [362] as a label generator for our active object and contact

affordance. The hand affordance we use comes from FrankMocap [337], which uses 100DoH [362] as a base model. Thus, does distilling labels from this detector via HRP actually provide a benefit over simply using the 100DoH model itself as a pre-trained representation? To test this question, we fine-tune policies on the toasting, pouring, and stacking (front cam) tasks and compare them against HRP applied to ImageNet, Ego4D, and (the weakest model) CLIP (see Table 6.1). In all cases, our representation handily beats the 100DoH policy. So while the affordance labels can dramatically boost policy learning (via HRP), the source/teacher models are not at all competitive on robotics tasks.

6.6.3 What Design Decisions are Important?

The following section ablates the key components of HRP to evaluate their relative importance. First, we apply HRP to each of the 6 baseline representations again, but this time none of the weights are kept fixed (see Sec. 6.4.2). These representations are fine-tuned on the toasting, stacking, and pouring tasks (front cam), and compared against the original HRP representations in Fig. 6.6. Note that fine-tuning all the layers results in a substantial performance hit on average, and this trend is consistent regardless of the base representation! Thus, we conclude fine-tuning only the layer norms when applying HRP is the correct decision.
Next, we ablate each of the affordance losses in Eq. 6.4, by applying HRP three times: once with $\lambda_{\text{ct}} = 0$, then with $\lambda_{\text{hand}} = 0$, and finally $\lambda_{\text{obj}} = 0$. This process is repeated using 3 different base models; ImageNet, Ego4D, and VC-1. This creates 9 ablated models (3 losses x 3 initializations) that are compared versus the full HRP models on the toasting, pouring, and stacking tasks. The average results are presented in Fig. 6.6, and the full, per-task breakdown is presented in the Appendix.We find that removing the object (Eq. 6.3) and hand (Eq. 6.2) losses uniformly results in significant performance degradation. Meanwhile, the contact loss (Eq. 6.1) only provides a significant boost for the Ego4D base model but does not affect the others. Thus, we conclude that object and hand losses are critical for our method, while the contact loss is more marginal.

6.6.4 OOD Analysis

We evaluate the performance of HRP and baseline approaches in OOD settings, by adding extraneous "distractor" objects (an orange carrot and a light green bowl) in the stacking task. The robot must successfully ignore the distractor and complete the task. Results are presented in Table 6.3. We found that both ImageNet + HRP and ImageNet had the same level of robustness to distractors. Meanwhile, Ego4D's performance dropped substantially, while Ego4D + HRP remained robust. Our hypothesis is that human data by itself does not contain enough information to allow for OOD tasks. However, using HRP allows for more focus on task-relevant features, even when the representation is trained on less diverse data.

Initialization	**w/ HRP**	**MAE Initialization**
Ego4D	**40%** $\pm$ 15%	15% $\pm$ 11%
ImageNet	**40%** $\pm$ 15%	**40%** $\pm$ 15%

Table 6.3: This table compares `Ego4D + HRP` and `ImageNet + HRP` representations against their respective baselines on a *stacking w/ distractors*. We find that HRP helps in these settings, especially for `Ego4D`.

6.7 Conclusion

In this work, we investigate human affordances as a strong prior for training visual representations. Our insight is that the best representations for control contain *actionable* information. We thus present HRP, in which we extract contact points, hand poses and activate objects from human videos, and use these affordances for fine-tuning representations. HRP improves base model performance drastically, for five different, downstream behavior cloning tasks, across three robot morphologies and three camera views. All components of our approach, including LayerNorm tuning, our three affordances, and our distillation process (from affordance labels to representations) are important for the model's success. One key limitation of this approach is that it has only been tested on imitation settings. In the future, we hope to not only scale this approach to many more tasks and robot morphologies, but also incorporate HRP in other robot learning paradigms such as reinforcement learning or model based control.

Chapter 7

Learning Dexterity from Internet Videos

7.1 Motivation

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 [127, 73, 125]. However, most of the prior work tends to use human data as a mechanism for pretraining just the visual representation [302, 356, 271, 424, 269], much like how deep learning has been used as a pretraining tool in related areas of computer vision [153, 60] and natural language processing [39, 93]. 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 7.1). 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.

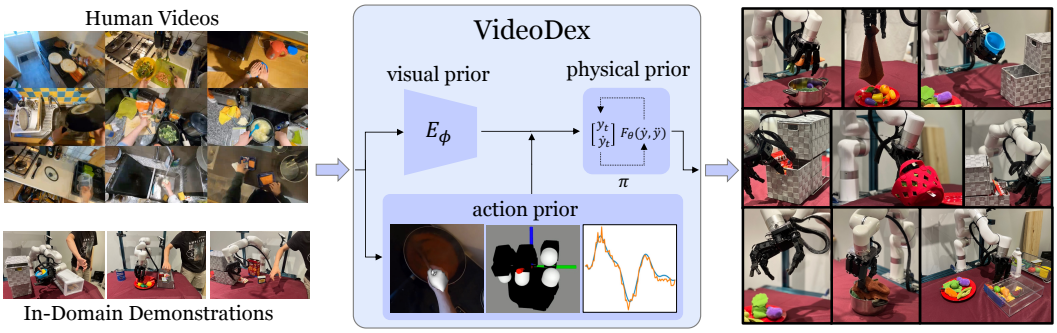


Figure 7.1: We re-target human videos as an action prior, use pretrainined embeddings as a visual prior, and use Neural Dynamical Policies (NDPs) [25] as a physical prior to complete many different tasks on a robotic hand.

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 [73, 127]. 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 [25, 22] (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 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.

7.2 Related Work

Learning for Dexterity Reinforcement learning (RL) with an engineered reward function can show dexterous simulation results [170, 203] but requires lots of data, especially in high DOF dexterous manipulation. This requires simulators [403, 236], which cannot model physics properly, making real-world transfer difficult. Behavior cloning is an approach [307, 33] that can work safely. DIME [18] involves using nearest neighbor matching of image representations with demonstrations to determine actions. Qin et al. [312] (2022) teleoperate and learn policies in simulation,

followed by Sim2Real transfer. DexMV[314] uses collected human hand videos for robot hand imitation learning. DexVIP [239] 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 [461] for hand poses, 100 Days of Hands [362] for hand-object interactions, Something-Something [125] for semantically similar interactions, Human3.6M [159] and the CMU Mocap Database [2] for Human pose estimation. Epic Kitchens [73], ActivityNet datasets [103], or YouCook [81] are action-driven datasets we focus on for dexterous manipulation.

Learning Action from Videos Detecting humans, estimating poses of different body parts, or understanding the dynamics and interactions related to human motion is a commonly studied problem. One can model human hands using the MANO [336] model and the human body using SMPL, SMPL-X [224, 291] models. There are many efforts in human pose estimation such as [412, 172, 337]. We focus on FrankMocap [337] for our project as it is robust for online videos. Traditionally, teleoperation approaches have employed hand markers with gloves for motion capture [142] or VR settings [194]. Without gloves, Li et. al. [208] used depth images and a paired human-robot dataset for teleoperation, and Handa et. al. [145] 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 [364, 50, 23], learn action correspondences [351] both in a paired [367] and unpaired manner [381]. This data can also be used to extract explicit actions by leveraging structure in the collection (such as reacher-grabber tools [435]) or prediction of future hand and object locations [200], as well as key-point detectors [79]. This can also be used to build representations for robot learning [271, 283]. R3M [271] trains on the Ego4D [127] 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.

7.3 Background

7.3.1 Neural Dynamic Policies

Neural Dynamic Policies (NDPs) [25, 22, 85], 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 [158, 311, 349, 287]:

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f_w(x, g), \tag{7.1}$$

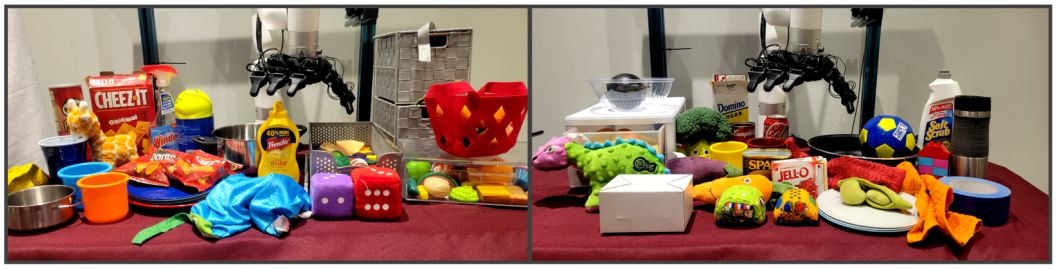


Figure 7.2: The collection of train objects (left) and test objects (right) used for experimentation.

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.

7.3.2 Learning from Watching Humans

Recently, Shaw et al.[380] introduced Robotic Telekinesis, a pipeline that teleoperates the Allegro Hand [1] using a single RGB camera. Leveraging work in monocular human hand and body pose estimation [337], hand and body modeling [336, 224, 291], 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 Shaw et al. [380]'s approach for demonstration collection.

We borrow the human hand to robot hand retargeting method from Robotic Telekinesis [380] 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:

$$E_\pi(\ (\beta_h, \theta_h),\ q\ ) = \sum_{i=1}^{10} ||v_i^h - (c_i \cdot v_i^r)||_2^2 \tag{7.2}$$

Shaw et al. [380] train an MLP $H_R(.)$ to implicitly minimize this energy function in 7.2, conditioned on knowing human poses (β, θ). For more details, we refer the readers to Shaw et al. [380].

7.4 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.

7.4.1 Visual Priors from Human Activity Data

Many previous works [271, 424, 269] 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. [271] as a useful visual initialization for our policy. Nair et al. [271] 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 [154] encoder, E_ϕ from R3M [271]. The output of this network is our visual representation for learning.

7.4.2 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. [145] and Shaw et al. [380] 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 7.4 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 [73] but an action detection network such as [67] can be used. Now, we must detect the hand. VideoDex first computes a crop c around the operator's hand using OpenPose [46] and the result is passed to FrankMocap [337] to obtain hand shape (β) and pose parameters (θ) of the 3D MANO model [336]. 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

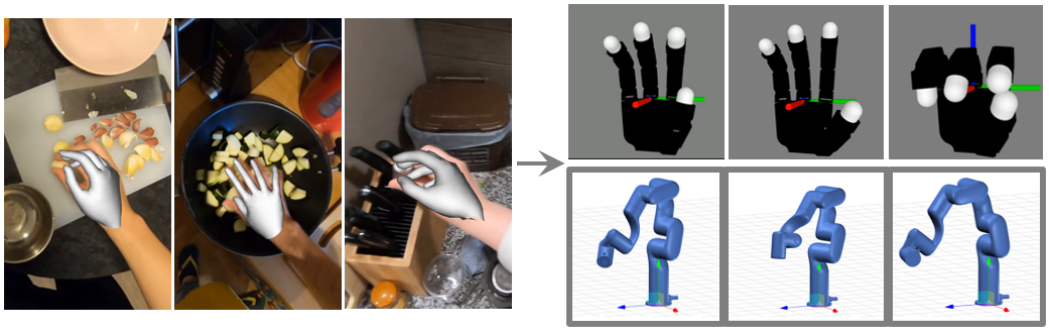


Figure 7.3: To use internet videos as pseudo-robot experience, we re-target human hand detections from the 3D MANO model [336] to 16 DoF robotic hand (LEAP) embodiment and we retarget the wrist from the moving camera to the xArm6 [3] embodiment. Videos at https://video-dex.github.io

Perspective-n-point algorithm [107]. 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 $M_{C_t}^{Wrist}$. To accurately obtain camera intrinsics for PnP, COLMAP is used [354].

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 $M_{C_1}^{C_t}$. To estimate this, we run monocular SLAM, specifically ORBSLAM3 [45].

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 [454]). Then an estimated depth map from RGB frames only using Adabins [30] is computed. This way, the method does not rely on the long-term contiguity of a video like most SLAM approaches. We then use depth map portions that correspond to the relevant objects and calculate a surface normal vector. We estimate α_p using this normal vector and the following equations:

$$\text{pitch} = \tan^{-1}(x_{Acc}/\sqrt{y_{Acc}^2 + z_{Acc}^2}) \tag{7.3}$$

$$\text{roll} = \tan^{-1}(y_{Acc}/\sqrt{x_{Acc}^2 + z_{Acc}^2}) \tag{7.4}$$

Detailed ablations on the parameterization of the initial pitch of the predicted trajectory (α) are provided in Section 7.6. In SLAM, we also remove the dependency on gyroscope data by assuming that the scaling factor is 1.0. This is acceptable because the trajectory is rescaled to the robot frame later. Therefore, this wrist re-targeting approach uses only 2D images from human videos.

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

$$M_{Robot}^{Wrist} = T_{Robot}^{World} \cdot M_{World}^{C_1} \cdot M_{C_1}^{C_t} \cdot M_{C_t}^{wrist} \tag{7.5}$$

Algorithm 5 Procedure for VideoDex

Require: Human videos $V_{1:K}^H$ (length T), policy π_θ, demonstrations $\mathcal{D}_{1:N}$. Human detection f_{human} [337].
  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 7.3, 7.4 and 7.5,
      Hand pose $h_t = H(\theta_t, \beta_t)$
    end for
    Store all h_t, w_t into robot trajectory τ_R^k
    $\hat{\tau}_R^k = \pi_\theta(I_1^k, h_1^k, w_1^k)$
    Optimize $\mathcal{L}_\theta = ||\tau_R^k - \hat{\tau}_R^k||_1$
  end for
  Store policy weights θ_h to initialize π_θ
  while not converged **do**
    for $n = 1...N$ **do**
      $\tau_n, I_{1:T}^n = \mathcal{D}_n$
      $\hat{\tau}_n = \pi_\theta(I_1^n, h_1^n, w_1^n)$
      Optimize $\mathcal{L}_\theta = ||\tau_n - \hat{\tau}_n||_1$
    end for
  end while

Re-targeting Hand Pose Human hands are also in a different *embodiment* compared to that of robot hands, like our 16 DOF LEAP Hand [370] . Similarly, to Shaw et al. [380], 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 Shaw et al. [380], and get human hand trajectories: τ_R^{hand} in the robot's embodiment. We use τ_R to denote the combined hand and wrist trajectories: τ_R^{hand}, τ_R^{wrist}. See Figure 7.3 for a visualization.

7.4.3 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 tele-operation. 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 [154] encoder provided by Nair et al. [271], E_ϕ. Then, we

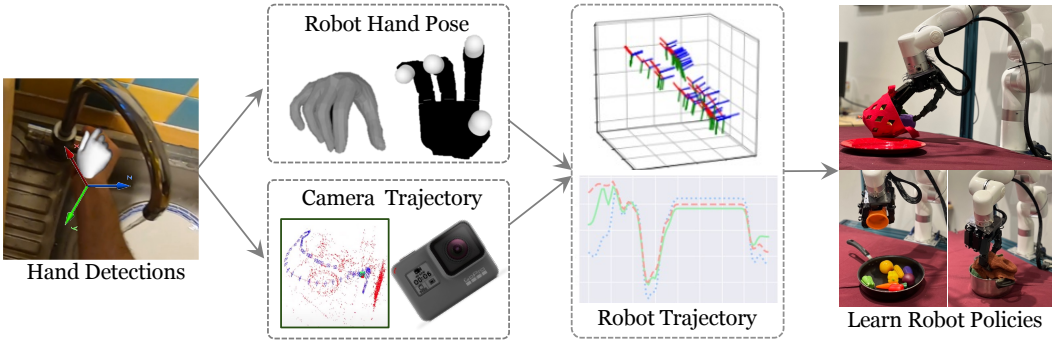


Figure 7.4: To use human videos as an action prior for training policies, we re-target them to the robot embodiment. The detected human fingers are converted to the robot fingers using a learned energy function. The wrist is re-targeted using the detections and camera trajectory and transformed to the robot arm.

introduce a *physical prior* to the network, the physically-inspired Neural Dynamic Policies [25, 22].

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 [73]. 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 5. Our network consists of the R3M [271] initialized ResNet-18 [154]. We process these features with a 3-layer MLP with a hidden layer size of 512, which are then processed by 2 NDP [25] networks.

7.5 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

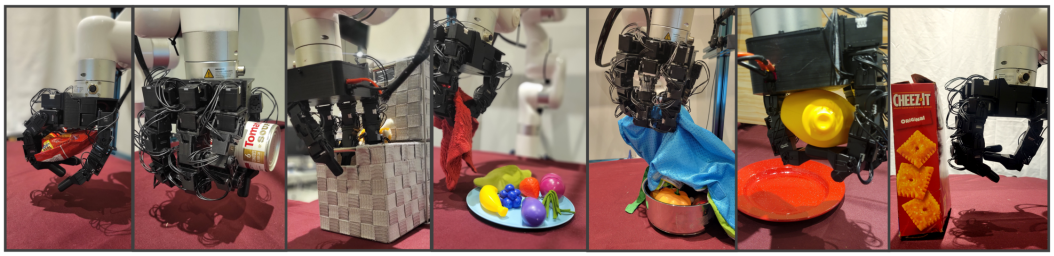


Figure 7.5: Tasks used in experiments. From left to right: pick, rotate, open, cover, uncover, place and push. See https://video-dex.github.io for videos of these tasks.

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)?

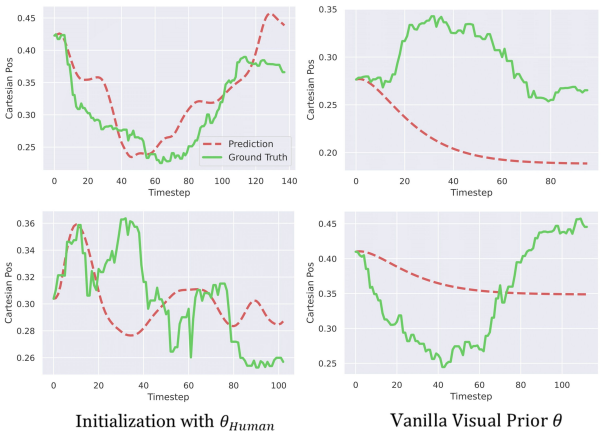


Figure 7.6: Networks initialized using action priors on human data without further training are closer to ground truth robot trajectories than networks only initialized using visual priors.

Task Setup We pretrain action priors on retargeted Epic Kitchens data for seven robot tasks. Then, we collect about 120-175 demonstrations for each of these tasks on our setup to train the policy. In `pick`, the goal is to pickup an object. In `rotate`, the agent grasps and rotates the object in place. In `cover` and `uncover`, the goal is to cover or uncover a pan/plate with a soft cloth object. `Push` involves flicking/poking an object with the fingers. In `place`, the robot has to pick up an object and place it into a plate, pan or pot. In `open` we open three different drawers. Our testing procedure consists of unseen locations and objects.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 [3]. Separate action priors are trained for the 16 DoF LEAP Hand and the 2-finger gripper.

	Pick		Rotate		Open		Cover		Uncover		Place		Push	
	train	test	train	test	train	test	train	test	train	test	train	test	train	test
BC-NDP [22]	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[85]	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 [85]	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 7.1: We present the results of train objects and test objects for Videodex and baselines as described above.

7.6 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 [85]. 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 [193], 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 [377] (VideoDex-VGG) and the MVP network [424, 150] (VideoDex-MVP) based representation trained for robot learning. We ablate the need for a two stream policy, instead training a single NDP for both hand and wrist. (VideoDex-Single) To see if VideoDex works with fewer demonstrations (around 50 demonstrations, 5-7 per variant only), we train a policy called VideoDex-Constrained.
We analyze the results of our experiments and the guiding questions discussed in Section 7.5. We present the results of our findings as a 0-1 success rate in Table 7.1 and the result of the ablations we ran on the place task in Table 7.4.

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 7.5. 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 7.6. 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. [271] (right). From the results, we see that VideoDex-BC-Open with action priors (Table 7.4) 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 7.4, our offline RL baseline, `CQL` [193] does not perform as well as the rest of the approaches, even under-performing the Behavior Cloning setup. Qualitatively, we see a much less smooth and less safe execution with this method, thus we only perform it on one task (`place`). Note that we use the same visual prior for this as well.

	Place	Open	Pick
`1-DOF BC-Open`[85]	0.62	0.69	0.71
`1-DOF VideoDex`	**0.69**	**0.82**	**0.77**

Table 7.2: 1-DOF xArm gripper performance using Videodex [3].

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 7.2, 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 7.3: 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.

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 7.3, 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 7.1 and in Table 7.4. 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 7.4 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 7.1, 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.

	Train	**Test**
Baselines:		
`BC-NDP` [22]	0.70	0.35
`BC-Open` [85]	0.40	0.25
`BC-RNN` [85]	0.70	0.50
`CQL` [193]	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 7.4: We present the results of the ablations discussed in Section 7.5. These are all performed on the `place` task.

Effect of Visual Priors We compared using our approach with MVP (`VideoDex-MVP`) [424] and VGG (`VideoDex-VGG`) [377] and their performance was below VideoDex using Nair et al. [271]. This is likely because both encoders are much larger than the ResNet18 [154] 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. [424] trained the representation in self-supervised fashion on videos and use the embeddings to perform robotics tasks in simulation. We see in Table 7.1, 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 [1]. 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 [370] .

7.7 Conclusion

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 [73], 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.

Part IV

Action Priors for Robotics

Chapter 8

Neural Dynamic Policies

8.1 Motivation

Consider the tasks such as pouring liquid or scooping beans as shown in Figure 8.1. These tasks are dynamic in nature, i.e., they require the robot to continuously apply the right forces and accelerations to act in a reactive manner to changes in the environment. Unlike quasi-static tasks, e.g. pushing or 2D grasping, where it can take arbitrarily long in between each action, the robot needs to reason at the whole trajectory level to execute a swift motion to perform dynamic tasks. For instance, if the robot scoops the beans too slowly they will fall back into the bowl, or if scooped too quickly the beans will be thrown out of the bowl. A common way to address this trajectory-level reasoning is to encode robot movements using nonlinear dynamical systems, like the ones that govern the flow of heat or the movement of planets. This idea is encapsulated by a family of methods known as Dynamic Movement Primitives (DMPs) [349, 158], which can compactly represent basic building block trajectories that are then stitched together to perform complex tasks. DMPs restrict the space of permissible robot movements by constraining the robot's goal and trajectory shape to obey a parametric nonlinear differential equation, consistent with the robot's kinematics and dynamics. This allows DMPs to effectively reason in the space of entire *trajectories*, rather than at the level of individual actions. Consequently, these approaches have led to impressive demos such as pancake flipping [187], dart throwing [183] or playing table tennis [261]. However, this class of approaches

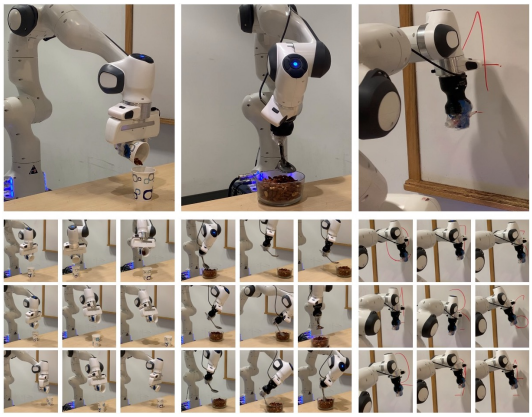


Figure 8.1: We present H-NDPs, an efficient real-world robot learning algorithm able to generalize across a high amount of diversity.

suffers from two drawbacks – (a) Generalization: Because of the constraints they impose on trajectories, DMPs do not have the power to represent general movements, and limit the controller to small variations in the initial or goal states. (b) Image-Observations: DMPs have mostly been built on estimating state vectors and struggle with high-dimension input, such as raw images. These shortcomings are in contrast to the flexibility allowed by deep learning methods in terms of generalization to unseen scenarios and scalability to high-dimensional image inputs [203, 303, 234, 170]. However, most of the real robot results using deep learning methods are still limited to pick and place-style quasi-static tasks, as compared to the dynamic tasks achieved by DMP-based methods.

Our goal is to address the following question: can we build *generalizable* robot policies for dynamic tasks by combining the ability of DMPs to reason in the space of trajectory distributions, with the ability of modern deep robot learning methods to learn from unstructured image data? We address this problem in our work, Bahl et al. [25], by embedding the structure of dynamical systems into deep neural network-based policies such that the agent can directly learn in the space of physically plausible trajectory distributions. **Our key insight is to reparameterize the action space in a deep policy network with nonlinear differential equations corresponding to a dynamical system and train it end-to-end over time in either reinforcement learning or imitation learning setups.** However, this is quite challenging to accomplish, since naively predicting a full arbitrary dynamical system directly from the input, trades one hard problem for another. Instead, we want to prescribe some structure such that the dynamical system itself manifests as a layer in the deep policy that is both, amenable to take arbitrary outputs of previous layers as inputs, and is also fully differentiable to allow for gradients to backpropagate. We address these challenges through our approach, Neural Dynamic Policies (NDPs) [25]. Specifically, NDPs allow embedding desired dynamical structure as a layer in deep networks. The parameters of the dynamical system are then predicted as outputs of the preceding layers in the architecture conditioned on the input. **The 'deep' part of the policy then only needs to reason in the lower-dimensional space of building a dynamical system that then lets the overall policy easily reason in the space of trajectories.** In this chapter, we employ the aforementioned DMPs as the structure for the dynamical system and show its differentiability, although they only serve as a design choice and can possibly be swapped for a different differentiable dynamical structure, such as RMPs [327].

However, it is hard for single dynamical systems to generalize to unseen state configurations, which is needed if we would like to operate a robot in the wild. Consider the examples in Figure 8.1 where we see a large diversity in the location and pose of objects. Depending on the initial state of the robot and the location of the containers, the joint trajectories will need to be diverse enough in terms of reachability and dynamics, to successfully perform the task of pouring or scooping. Hence, a direct attempt to fit a single policy to all these trajectories poses a big practical challenge

in optimization which is further aggravated when the input is a high-dimensional image – which we argue is the case in most interesting real-world robotic problems. Our solution is to harness the individual strengths of both dynamical systems for movement representations and deep network policies. Instead of directly fitting a single policy on diverse scenarios, we fit dynamical system-based policies first in local regions of the task space and then distill them together into a global one.

Our system consists of a library of *local* NDPs [25] and a single *global* NDP. Each local NDP exploits the strengths of DMPs: (a) overfit to operate in small regions of the task space and ensure task success at all times; (b) operate on privileged low-level state information as input. The global NDP is meant to operate on the entire space and only receives raw sensory data as input, e.g., raw images. This global policy is trained to not only maximize task performance but also to imitate the behavior of the local policies. The key here is that both local and global policies have different objectives: local policies place importance on task success in their local regions, and the global policy places importance on learning from images in a generalizable manner. Owing to this local-to-global structure, we call our framework *Hierarchical Neural Dynamic Policies* (*H-NDPs*).

Training H-NDPs for real-world robot learning comes with a practical challenge: there is no guarantee that the local trajectories distilled by the global NDP will indeed be successful when tried even on the same local locations as training unless it is completely overfitting in which case it will not generalize to new locations. Instead of hoping it to just work, we perform multiple iterations of this local-to-global procedure by re-training local NDPs by solving local tasks while being faithful to the global NDPs and then distilling the refined local ones into a new global NDP until convergence, as shown in Figure 8.2. Such an iterative process is standard practice in general [205, 41, 344] to prevent the distilled network from diverging. However, the added advantage of H-NDPs is that the embedded dynamical system enhances both safety, convergence, and overall performance, as we show in the results section later.

One of the major contributions is an exhaustive experimental evaluation of H-NDPs and several other baselines on real-world tasks of writing, scooping, and pouring with a robot. Our real-world experiments are conducted in realistic settings with raw high-dimensional images as inputs, with large variations in object positions and goal locations, involving several hundred hours of robot interaction. Finally, we also evaluate complex simulated tasks like throwing, catching, and picking. We show that H-NDPs achieve state-of-the-art performance across all the tasks in reinforcement as well as imitation learning settings

8.2 Related Work

Robot Learning for Dynamic Tasks Previous works have proposed and used Dynamic Movement Primitives (DMP) [349, 158, 311] for robot control [258, 300, 187].

Work has been done in representing dynamical systems, both as extensions of DMPs [66, 44, 404], and beyond DMPs by employing kernels [157] and designing Riemannian metrics [327]. Learning methods have been used to incorporate environment sensory information into the forcing function of DMPs [392, 320]. DMPs have also been prime candidates to represent primitives when learning hierarchical policies, given the range of motions DMPs can be used for in robotics [75, 389, 185, 288]. Parametrization of DMPs using gaussian processes has also been proposed to facilitate generalization [404, 297]. Recently, deep learning has also been used to map images to DMPs [281] and to learn DMPs in latent feature space [56]. However most of these works require pre-trained DMPs via expert demonstrations or are only evaluated in the supervised setting. Furthermore, either a single DMP is used to represent the whole trajectory or the demonstration is manually segmented to learn a different DMP for each segment. More recent work [25, 281, 327, 56], including our work Neural Dynamical Policies (NDPs) [25], has shown DMPs or dynamical systems such as RMPs can be incorporated in a differentiable, end-to-end deep learning setting, which is an attribute that H-NDPs leverage. In contrast to previous differentiable, end-to-end approaches that embed DMPs, our previous work (NDPs [25]) outputs a new dynamical system for each timestep to fit diverse trajectory behaviours across time. Since we embed dynamical structure into the deep network, NDPs can flexibly be incorporated not just in visual imitation but also in deep reinforcement learning setups, in an end-to-end manner.

Reparameterized Policy and Action Spaces A broader area of work that makes use of action reparameterization is the study of Hierarchical Reinforcement Learning (HRL). Works in the options framework [21, 396] attempt to learn an overarching policy that controls usage of lower-level policies or primitives. Lower-level policies are usually pre-trained therefore require supervision and knowledge of the task beforehand, limiting the generalizability of such methods. For example, Daniel et al.[75, 284] incorporate DMPs into option-based RL policies, using a pre-trained DMPs as the high level actions. This setup requires re-learning DMPs for different types of tasks and does not allow the same policy to generalize, since it needs to have access to an extremely large number of DMPs. Action spaces can also be reparameterized in terms of pre-determined PD controller [438] or learned impedance controller parameters [244]. While this helps for policies to adapt to contact rich behaviors, it does not change the trajectories taken by the robot. This often leads to high dimensionality, and thus a decrease sample efficiency. In addition, Whitney et al. [418] learn an action embedding based on passive data, however, this does not take environment dynamics or explicit control structure into account.

Structure in Policy Learning Various methods in the field of control and robotics have employed physical knowledge, dynamical systems, optimization, and more general task/environment dynamics to create structured learning. Works such as [68, 129] propose networks constrained through physical properties such as Hamiltonian

co-ordinates or Lagrangian Dynamics. However, the scope of these works is limited to toy examples such as a point mass, and are often used for supervised learning. Similarly, other works [325, 328, 294, 274] all employ dynamical systems to model demonstrations, and do not tackle generalization or learning beyond imitation. Fully differentiable optimization problems have also been incorporated as layers inside a deep learning setup [10, 58, 11]. While they share the underlying idea of embedding structure in deep networks such that some aspects of this structure can be learned end-to-end, they have not been explored in tackling complex robotic control tasks. Furthermore, it is common in RL setups to incorporate planning based on a system model [87, 88, 63, 20, 89]. However, this is usually learned from experience or from attempts to predict the effects of actions on the environment (forward and inverse models), and often tends to fail for complex dynamic tasks.

Hierarchical Frameworks for robot learning While DMPs have been used in previous works for building hierarchical policies [75, 389, 185, 288], these have mostly been constrained to discrete primitives [75, 288] and relatively simple settings from a perception standpoint. Previous works have also attempted to share knowledge between DMPs; for example Ruckert et al. [343] leverages shared basis functions for controlling multidimensional systems. To our knowledge, our work is the first to use a hierarchical local-to-global structure using DMPs.

Hierarchical frameworks are popular for deep imitation learning setups. One prominent example is Guided Policy Search (GPS [205, 203]) which uses a bottom-up approach by learning "expert" local controllers from state observations and then distill them into a image-based policy. While H-NDPs uses a similar bottom-up framework, we employ the structure of dynamical systems within our policy architecture, allowing us to perform more dynamic tasks than GPS. Furthermore, our local controllers are learnt from a few $(10 - 15)$ demonstrations, which are a lot easier to obtain than assuming fully observable environment and hand-engineering required for GPS.

Hierarchical learning has also long been explored in the context of RL from both top-down [408] as well as bottom-up [388, 262] perspective. Ghosh et al. [117] and Teh et al. [399] propose a hierarchical local-to-global framework to perform more complex, diverse tasks. Ghosh et al. [117] takes advantage of local RL policies trained via policy gradients and a global policy which imitates the former. The local-to-global interactions between neural networks can lead to suboptimal behavior, especially for more difficult dynamic tasks. On the other hand, the local-to-global interactions taking place within H-NDPs are in a much more structured space (the space of physically plausible trajectories), leading to a stronger performance by H-NDPs.

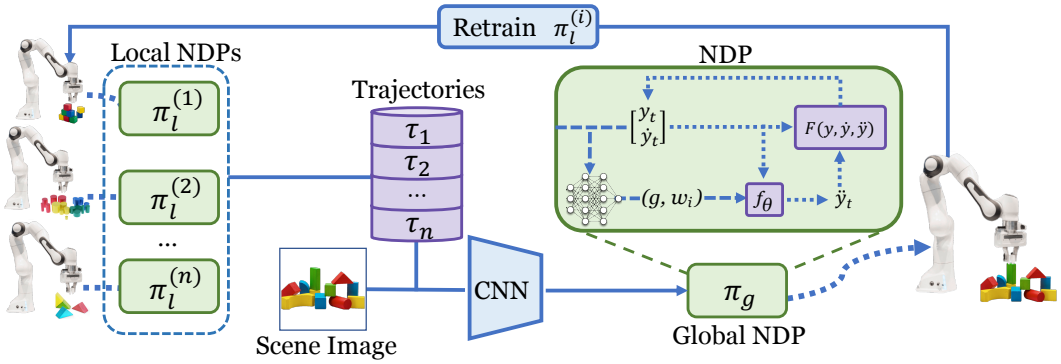


Figure 8.2: We train local Neural Dynamic Policies (NDPs) $\pi_l^{(i)}$ on each region i of the task space, from state observations. A global NDP π_g (usually taking in image input I_t) learns to imitate the local experts. We use the global NDP to retrain local NDPs which keeps the NDPs from diverging. These local-to-gloabl interactions happen in an iterative manner. NDPs make a good candidate for capturing such local-to-global interactions due to their shared structure and the fact that they operate over a smooth trajectory space.

8.3 Modeling Trajectories with Dynamical Systems

Consider a robotic arm exhibiting a certain behavior to accomplish some task. Given a choice of coordinate system, such as either joint-angles or end-effector position, let the state of the robot be y, velocity $\dot{y}$ and acceleration $\ddot{y}$. In mechanics, Euler-Lagrange equations are used to derive the equations of motion as a general second order dynamical system that perfectly captures this behavior [386, Chapter 6]. It is common in classical robotics to represent movement behaviors with such a dynamical system. Specifically, we follow the second order differential equation structure imposed by Dynamic Movement Primitives [158, 349, 287]. Given a desired goal state g, the behavior is represented as:

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f(x), \tag{8.1}$$

where α, β are global parameters that allow critical damping of the system and smooth convergence to the goal state. f is a non-linear forcing function which captures the shape of trajectory and operates over x which serves to replace time dependency across trajectories, giving us the ability to model time invariant tasks, e.g., rhythmic motions. x evolves through the first-order linear system:

$$\dot{x} = -a_x x \tag{8.2}$$

The specifics of f are usually design choices. We use a sum of weighted Gaussian radial basis functions [158] shown below:

$$f(x, g) = \frac{\sum \psi_i w_i}{\sum \psi_i} x(g - y_0), \quad \psi_i = e^{(-h_i(x - c_i)^2)} \tag{8.3}$$

where i indexes over n which is the number of basis functions. Coefficients $c_i = e^{\frac{-i\alpha_x}{n}}$ are the horizontal shifts of each basis function, and $h_i = \frac{n}{c_i}$ are the width of each of each basis function. The weights on each of the basis functions w_i parameterize the forcing function f. This set of nonlinear differential equations induces a smooth trajectory distribution that acts as an attractor towards a desired goal (see Figure 8.1, right). We now discuss how to combine this dynamical structure with deep neural network based policies in an end-to-end differentiable manner.

8.4 Neural Dynamic Policies (NDPs)

In our work, Bahl et al. [25], we condense actions into a space of trajectories, parameterized by a dynamical system, while keeping all the advantages of a deep learning based setup. We present a type of policy network, called Neural Dynamic Policies (NDPs) that given an input, image or state, can produce parameters for an embedded dynamical structure, which reasons in trajectory space but outputs raw actions to be executed. Let the unstructured input to robot be s, (an image or any other sensory input), and the action executed by the robot be a. We describe how we can incorporate a dynamical system as a differentiable layer in the policy network, and how NDPs can be utilized to learn complex agent behaviors in both imitation and reinforcement learning settings.

8.4.1 Neural Network Layer Parameterized by a Dynamical System

Throughout this chapter, we employ the dynamical system described by the second order DMP equation (8.1). There are two key parameters that define what behavior will be described by the dynamical system presented in Section 8.3: basis function weights $w = \{w_1, \ldots, w_i, \ldots, w_n\}$ and goal g. NDPs employ a neural network Φ which takes an unstructured input s[1] and predicts the parameters w, g of the dynamical system. These predicted w, g are then used to solve the second order differential equation (8.1) to obtain system states $\{y, \dot{y}, \ddot{y}\}$. Depending on the difference between the choice of robot's coordinate system for y and desired action a, we may need an inverse controller $\Omega(.)$ to convert y to a, i.e., $a = \Omega(y, \dot{y}, \ddot{y})$. For instance, if y is in joint angle space and a is a torque, then $\Omega(.)$ is the robot's inverse dynamics controller, and if y and a both are in joint angle space then $\Omega(.)$ is the identity function.
NDPs are defined as

$$\pi(a \mid s; \theta) \triangleq \Omega\big(\texttt{DE}\big(\Phi(s; \theta)\big)\big) \tag{8.4}$$

Where $\texttt{DE}(w, g) \rightarrow \{y, \dot{y}, \ddot{y}\}$ denotes solution of the differential equation (8.1). The forward pass of $\pi(a \mid s)$ involves solving the dynamical system and backpropagation requires it to be differentiable. We now show how we differentiate through the dynamical system to train the parameters θ of NDPs.

[1] robot's state y is not to be confused with environment observation s which contains world as well as robot state (and often velocity). s could be given by either an image or true state of the environment.

8.4.2 Training NDPs by Differentiating through the Dynamical System

To train NDPs, estimated policy gradients must flow from a, through the parameters of the dynamical system w and g, to the network $\Phi(s;\theta)$. At any time t, given the previous state of robot y_{t-1} and velocity $\dot{y}_{t-1}$ the output of the DMP in Equation (8.1) is given by the acceleration

$$\ddot{y}_t = \alpha(\beta(g - y_{t-1}) - \dot{y}_{t-1)} + f(x_t, g) \tag{8.5}$$

Through Euler integration, we can find the next velocity and position after a small time interval dt

$$\dot{y}_t = \dot{y}_{t-1} + \ddot{y}_{t-1}dt, \quad y_t = y_{t-1} + \dot{y}_{t-1}dt \tag{8.6}$$

In practice, this integration is implemented in m discrete steps. To perform a forward pass, we unroll the integrator for m iterations starting from initial $\dot{y}_0$, $\ddot{y}_0$. We can either apply all the m intermediate robot states y as actions on the robot using inverse controller $\Omega(.)$, or equally sub-sample them into $k \in \{1, m\}$ actions in between, where k is the NDP rollout length. This frequency of sampling could allow robot operation at a much higher frequency (.5-5KHz) than the environment (usually 100Hz). The sampling frequency need not be same at training and inference. We learn at a much higher frequency (100-200Hz) and downsample at inference due to physical constraints to 30-50Hz.
Now we can compute gradients of the trajectory from the DMP with respect to w and g using Equations (8.3)-(8.6) as follows:

$$\frac{\partial f(x_t, g)}{\partial w_i} = \frac{\psi_i}{\sum_j \psi_j}(g - y_0)x_t, \quad \frac{\partial f(x_t, g)}{\partial g} = \Sigma_i \frac{\psi_i w_i}{\sum_j \psi_j} x_t \tag{8.7}$$

Using this, a recursive relationship follows between, (similarly to the one derived by Pahic et al. [281]) $\frac{\partial y_t}{\partial w_i}$, $\frac{\partial y_t}{\partial g}$ and the preceding derivatives of w_i, g with respect to y_{t-1}, y_{t-2}, $\dot{y}_{t-1}$ and $\dot{y}_{t-2}$.
We now discuss how NDPs can be leveraged to train policies for imitation learning and reinforcement learning setups.

8.4.3 Training NDPs for Imitation (Supervised) Learning

Training NDPs in imitation learning setup is rather straightforward. Given a sequence of input $\{s, s', \dots\}$, NDP $\pi(s;\theta)$ outputs a sequence of actions $a, a' \dots$. In our experiments, s is a high dimensional image input. Let the demonstrated action sequence be τ_{target}, we just take a loss between the predicted sequence as follows:

$$\mathcal{L}_{\text{imitation}} = \sum_s || \pi(s) - \tau_{\text{target}}(s) ||^2 \tag{8.8}$$

The gradients of this loss are backpropagated as described in Section 3.2 to train the parameters θ.

8.4.4 Training NDPs for Reinforcement Learning

Algorithm 6 Training NDPs for RL

Require: Policy π, k NDP rollout length, Ω low-level inverse controller
  for $1, 2, ...$ episodes **do**
    for $t = 0, k, \ldots,$ until end of episode **do**
      w, $g = \Phi(s_t)$
      Robot y_t, $\dot{y}_t$ from s_t (pos, vel)
      for $m = 1, ..., M$ (integration steps) **do**
        Estimate $\dot{x}_m$ via (8.2) and update x_m
        Estimate $\ddot{y}_{t+m}$, $\dot{y}_{t+m}$, y_{t+m} (8.5), (8.6)
        $a = \Omega(y_{t+m}, y_{t+m-1})$
        Apply action a to get s'
        Store transition (s, a, s', r)
      end for
      Compute Policy gradient $\nabla_\theta J$
      $\theta \leftarrow \theta + \eta \nabla_\theta J$
    end for
  end for

We now show how an NDP can be used as a policy, π in the RL setting. As discussed in Section 8.4.2, NDP samples k actions for the agent to execute in the environment given input observation s. One could use any underlying RL algorithm to optimize the expected future returns. For this, we use Proximal Policy Optimization (PPO) [355] and treat a independently when computing the policy gradient for each step of the NDP rollout and backprop via a reinforce objective.
There are two choices for value function critic $V^\pi(s)$: either predict a single common value function for all the actions in the k-step rollout or predict different critic values for each step in the NDP rollout sequence. We found that the latter works better in practice. We call this a *multi-action critic architecture* and predict k different estimates of value using k-heads on top of the critic network. Later, in the experiments we perform ablations over the choice of k. Algorithm 6 provides a summary of our method for training NDPs.

8.5 Hierarchical Neural Dynamical Policies (H-NDPs)

Consider the real world task of scooping from a bowl. The robot has to both plan a trajectory that will allow it to do the scooping motion properly, and understand any potential randomness, for instance if the bowl changes locations. A single policy likely will have a lot of trouble with this. We address such challenges by presenting Hierarchical Neural Dynamic Policies (H-NDPs). H-NDPs use a local-to-global

Algorithm 7 Training H-NDPs

Require: NDP Policy randomly initialized global policy $\pi_g(I)$, taking image I as input, with weights θ_g, M local regions R_i, for each region i a local NDP $\pi_l^{(i)}(.)$ (taking state s_t) as input, and corresponding NN weights $\theta_l^{(i)}$, initialize empty $\mathcal{D}$

for $1, 2, ...$ iterations **do**

  for $i = 1...m$ **do**

    Run policy $\pi_l^{(i)}$ on environment R_i for H steps

    Collect trajectory $F(\phi(.; \theta_l^{(i)}))$ and store into $\mathcal{D}$

    Compute $\mathcal{L}_{\text{local}} = \mathcal{L}_i(\theta_l^{(i)}) + \alpha_i D_{KL}(\pi_l^{(i)} \mid\mid \pi_g)$

    $\theta_l^{(i)} \leftarrow \theta_l^{(i)} - \eta \nabla_{\theta_l^{(i)}} \mathcal{L}_{\text{local}}$ (until convergence)

  end for

  Compute $\mathcal{L}_{\text{BC}} = \sum_{i=1}^{i=M} \mid\mid F(\phi(s_t; \theta_g)) - F(\phi(s_t; \theta_l^{(i)})) \mid\mid_2$

  Compute loss $\mathcal{L}_{\text{global}} = \mathcal{L}_{\text{BC}} + \mathcal{L}_g(\theta_g)$

  $\theta_g \leftarrow \theta_g - \eta \nabla_{\theta_g} \mathcal{L}_{\text{global}}$ (until convergence)

end for

learning scheme which makes it much easier for the agent to learn how to handle diversity in the task and deal with raw image inputs. We leverage structure provided by NDPs [25] for policy learning, allowing our hierarchical policies to operate in a shared space, and thus leads to smoother trajectories and more sample efficient learning. In this section, we describe how this setup works, both in the reinforcement and imitation learning settings.

8.5.1 Hierarchical Neural Dynamical Policies

We break down policy learning for a given task into two components: local controllers which operate from exact state observations, and a global policy which learns from raw sensory observations, for example robot poses and images. Both policies are NDP; the policy networks have an embedded dynamical system as a layer. Directly optimizing the global policy for the full task can be difficult, since dynamical systems can easily overfit to a single trajectory. Let π be an NDP and let $\phi(s; \theta)$ be the deep network inside the NDP, parameterized by weights θ. $\phi(s; \theta)$ outputs the DMP parameters which are used by the forward integrator F to solve the differential equation described in Equation 8.1. τ, the output of $\pi(. \mid s)$, is $F(\phi(s; \theta))$.

We divide the task space into M different regions. We train local NDPs $\pi_l^{(i)}$ on each region i (R_i) of the task space. For example, for a task like scooping, each bowl location would be its own region, and we would train a single NDP to solve the task for that specific bowl location. For the rest of this section, let the local policy $\pi_l^{(i)}$ be parameterized by network weights by $\theta_l^{(i)}$. For task i, we will compute loss on the NDP output, $\mathcal{L}_i(\theta_l^{(i)})$ and optimize with respect to $\theta_l^{(i)}$. This loss can be any

differentiable loss based on the policy output. In the next two sections, we will describe what $\mathcal{L}_i$ is in the case of RL and imitiation learning.
Once the local policies are trained, the global NDP π_g parameterized by network weights θ_g, learns to imitate the local policies. This makes it easier for the global policy to understand the difference between high level task details and low level task optimization. The global NDP conditioned on the current observation, s_t, learns to clone the behavior of the local NDPs in using the loss: $\mathcal{L}_{\text{BC}} = \sum_i \; || \; F(\phi(s_t; \theta_g)) - F(\phi(s_t; \theta_l^{(i)})) \; ||_2$. There is no guarantee, however, that a single iteration of behavior cloning will work. In practice, an iterative process is standard. Therefore, we fine-tune π_g on the loss function for the full task (union of all task regions R_i), $\mathcal{L}_g(\theta_g)$. This a hand-designed (or a simple binary) signal coming from the task, similar to a reward in RL. In summary, the overall global NDP training loss is:

$$\mathcal{L}_{\text{global}} = \mathcal{L}_{\text{BC}} + \mathcal{L}_g(\theta_g) \tag{8.9}$$

We would like to minimize the amount of human supervision, thus we do not want to create more task spaces but need more data to train the policy. One possibility is to retrain the local NDPs and collect more data, However this could easily lead to divergence. Thus we use the NDP π_g to reduce divergence by adding $\alpha_i D_{KL}(\pi_l^{(i)} \; || \; \pi_g)$ to the local NDP task loss $\mathcal{L}_i(\theta_l^{(i)})$. α_i is a hyperparameter for the weight of this extra loss term. We collect more data from the local experts and repeat the above steps until convergence. We call this process *iterative refinement*. This structure allows the local experts to adjust their outputs based on what is easier for the global policy to learn. NDPs make a good candidate for such a learning scheme due to their shared structure and the fact that they both operate over a smooth trajectory space, leading to much more efficient learning. Hence, the overall loss for the ith local NDPs is:

$$\mathcal{L}_{\text{local}} = \mathcal{L}_i(\theta_l^{(i)}) + \alpha_i D_{KL}(\pi_l^{(i)} \; || \; \pi_g) \tag{8.10}$$

We provide a detailed description of H-NDPs in Algorithm 7. The general idea of local-to-global learning has been widely studied in machine learning for generalization in complex domains [390, 41, 344]. For robot learning in particular, local-to-global structure has been exploited by for imitation learning by Levine et al. [205] and for RL by Ghosh et al. [117, 399]. In contrast to the black-box policy networks used in these works, our main contribution is to embed the structure of a nonlinear dynamical system within the network. NDPs make the interactions between global and local policies a lot more efficient, since both operate in the same DMP parameter space. This allows generalization to new configurations for dynamic tasks, a strong advantage of our method. We now discuss how to apply H-NDPs to both imitation and RL settings in the following subsections.

8.5.2 H-NDPs for Imitation Learning

In the imitation learning setting, we train the global NDP (π_g) via visual inputs and the local NDPs ($\pi_l^{(i)}$) are trained via supervised learning to imitate kinesthetic demonstrations. We start with a single demonstration for each R_i. Let this demonstration be $\tau_{\text{demo}}^{(i)}$. Therefore the local NDP loss $\mathcal{L}_i$ (described in Equation 8.10) for the IL case:

$$\mathcal{L}_i(\theta_l^{(i)}) = || \; F(\phi(s_t; \theta_l^{(i)})) - \tau_{\text{demo}}^{(i)} \; ||^2 \tag{8.11}$$

For simplicity, both local and global NDPs are set to be Gaussian with a fixed variance. The KL-divergence in the extra loss term to the local NDP loss (described in Equation 8.10), $D_{KL}(\pi_l^{(i)} \; || \; \pi_g)$ therefore simply becomes:

$$\alpha_i \; || \; F(\phi(s_t^{(i)}; \theta_l^{(i)})) - F(\phi(s_t^{(i)}; \theta_g)) \; ||_2 \tag{8.12}$$

Here, let $s_t^{(i)}$ be the observation received by the agent while performing task i. Naively using a constant α might make the local NDPs worse. For instance, at the beginning of training, the global NDP may not be successful for every task region. Instead, we deploy the trained global policy to collect $F(\phi(s_t^{(i)}; \theta_g))$, a trajectory for every local region i. We set $\alpha_i = 1$ only if $F(\phi(s_t^{(i)}; \theta_g))$ is judged successful by a human. Otherwise, we set it to 0. This does require the same person to judge the full iteration, in order to avoid inconsistencies. In the future, we hope to have an automatic process for this. Note that we use the KL penalty beteen trajectories, but these could be directly in the space of the NDP parameter distributions, as they are lower dimensional. Finally, $\mathcal{L}_g(\theta_g)$, the loss function for the global NDP is simply the imitation learning loss on the original demonstrations:

$$\mathcal{L}_g(\theta_g) = \sum_i || \; F(\phi(s_t^{(i)}; \theta_g)) - \tau_{\text{demo}}^{(i)} \; ||^2 \tag{8.13}$$

8.5.3 H-NDPs for Reinforcement Learning

In the RL framework, the objective is to learn a policy $\pi(a_t \mid s_t)$ that maximizes the sum of expected rewards $R_t = \mathbb{E}[\sum_{i=t}^{T} \gamma^i R(s_i, a_i, s_{i+1})]$. It can be difficult for RL policies to work in highly dynamic environments, especially when there is a high amount of stochasticity or variation in the task.

In the RL setting, similarly to the imitation learning setting, we split the task into i regions. Each local NDP is an RL policy and we use $\mathcal{L}_i = J_i$, where J_i is the surrogate policy gradient loss from an off-the-shelf policy optimization algorithm. Specifically, we use the loss from Proximal Policy Optimization (PPO [355]), and update the parameters of $\pi_i^{(l)}$ with respect to ∇J_i. Similarly, π_g is optimized via PPO on the loss $\mathcal{L}_g = J_g$. To avoid divergence of the global NDP, we compute the KL divergence from the global NDP to each local NDP, as in Equation 8.10, and add to the local

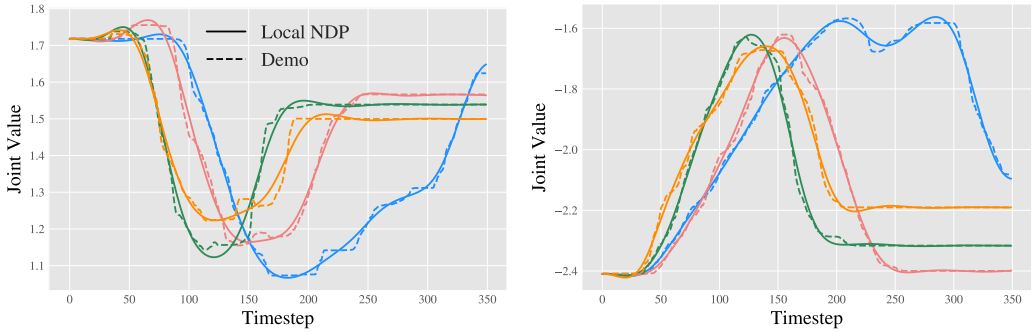


Figure 8.3: Visualizations of the original demonstrations and the trained local NDP on selected joints, for the real-world scooping task. The x-axis is the timestep and y is the joint value. Each curve representations a different demonstration. We can see that local NDPs can efficiently capture the desired motion in a smooth manner.

NDP training loss, $\mathcal{L}_i$. Since we use Gaussian policies with learned variance, then we have that: $D_{KL}(\pi_l^{(i)} \mid\mid \pi_g) = \log \frac{\sigma_g}{\sigma_i} + \frac{\sigma_i^2 + (\mu_i - \mu_g)^2}{2\sigma_g^2} - \frac{1}{2}$. Here the output of $\pi_i^{(l)}$ is $\mathcal{N}(\mu_i, \sigma_i^2)$ and that of π_g is $\mathcal{N}(\mu_g, \sigma_g^2)$. In practice, we found that setting α_i to either 0 or a very low value worked much better. An explanation for this phenomenon is that NDPs already contain more structure via the embedded dynamical system and hence do not need a KL-divergence constraint. The DMP parameter space these policies are in is already a lot more meaningful than the general neural network space.

8.5.4 Advantages of H-NDPs for Real-World Robotics

Due to their search over a physically smooth space, H-NDPs provide safer and more efficient learning. This can be a benefit in the real world, where hardware setups can be brittle and exploration can be dangerous. In fact, in Figure 8.3 we show the trajectory for multiple joints of sampled demonstrations and the output of the corresponding fitted local NDP, which learns a much smoother version of the demonstration. Secondly, since H-NDPs operate at a trajectory level, the policy π is only used every k steps. Hence fewer forward passes need to be taken by the policy network. With large networks and computationally expensive hardware (such as robot controllers and cameras), more forward passes can actually be an impediment to the learning algorithm, as it cannot execute the task at a high enough frequency. Additionally, we can also sample trajectories at arbitrary lengths, and can therefore output a more compact trajectory if needed.

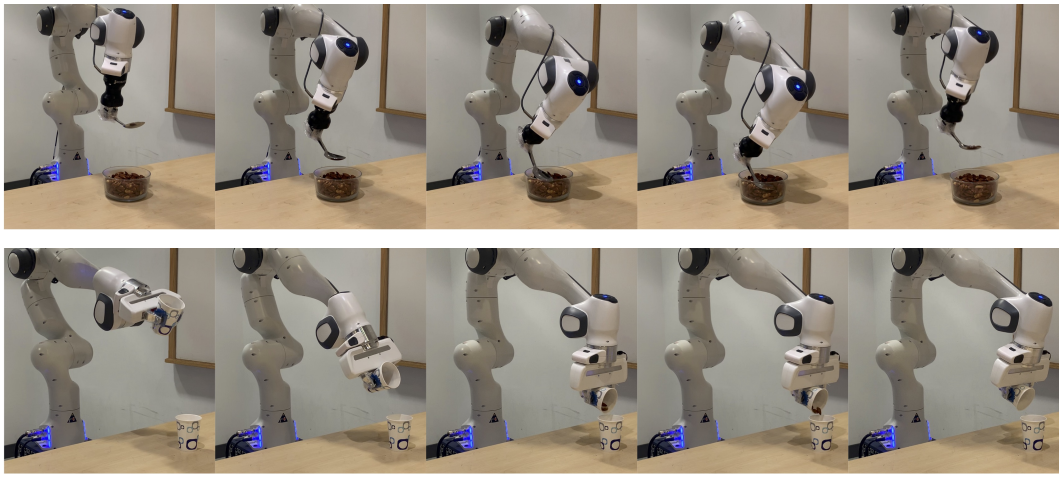


Figure 8.4: Sample trajectories for Scooping (top) and Pouring (bottom) tasks, on the Franka Panda robot

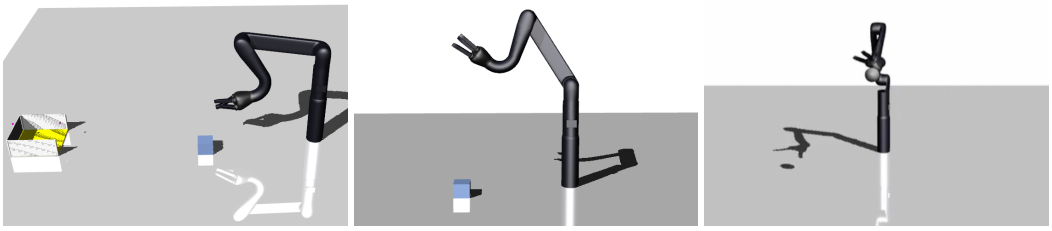
8.6 Experimental and Implementation Setup

8.6.1 Real Robot Tasks Setup

For all the real world tasks (scooping, pouring and writing), we use visual inputs for the global NDP and state inputs for the local NDP. We use a Franka Panda 7 DoF robot, controlled by joint angle control. We use the robot control code from Zhang et al. [447]. We run the controller at about 50 Hz for both local and global policies. We are in fact bottlenecked by the frequency of the image capture and processing software. Note that a neural network policy which needs image inputs every timestep would be significantly slower (5-10Hz). An H-NDP predicts one trajectory of k steps, thus needs only one forward pass per k steps allowing for a 50Hz controller. In all of our experiments, we use $k = 350$. To mimic real world conditions, we vary goal locations and intentionally change the scene a little bit (slightly shift the robot, camera or the object in the robots hand). For ease of use, we utilize the same initial robot joint positions. In each of the real world tasks, a human decides whether a trial is successful or failed. More details of each task can be found in the Appendix material.

8.6.2 Simulated Tasks Setup

We also perform simulation experiments on dynamic tasks, inspired from tasks performed by Ghosh et al. [117]. The simulated robot is a 6 DoF Kinova Jaco, which we control in joint angle space. All tasks are simulated in the MuJoCo [403] framework. Throwing involves first grabbing then throwing a cube inside a target box. To add diversity to the task, we vary the location of the goal box. These



(a) Throwing (b) Picking (c) Catching

Figure 8.5: Reinforcement Learning environments in MuJoCo [403].

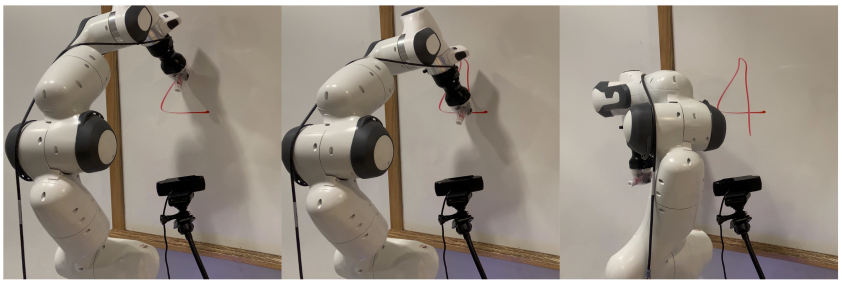
constitute different regions of the task space. For picking, the goal is to grasp a cube and lift it as highly possible. Here the diversity comes from varying the starting position of the block. Finally, we perform the task of catching a ball being launched in the air. The goal is for the robot to catch the ball and keep it in its hand till the end of the episode. Here, we randomize the starting location of the ball. Images of these tasks can be seen in Figure 8.5.

8.6.3 Policy Architecture and Network Pretraining

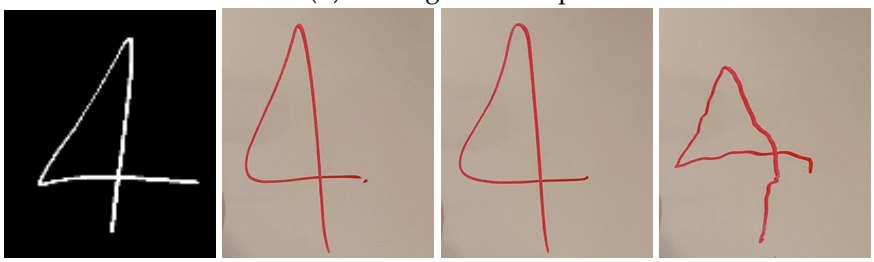
In the RL setting, we use the same architecture as Bahl et al. [25] (2 hidden layers of 100 neurons). In the imitation learning setting, we use a very small fully connected neural network (one hidden layer with 40 neurons) for our local policies, and a similar Convolutional Neural Network (CNN) architecture to that of GPS [203]. We also use a spatial softmax layer [203], which for each channel c $f_{cx} = \sum_{ij} s_{cij} x_{ij}$ and $f_{cy} = \sum_{ij} s_{cij} y_{ij}$, where i, j are pixel coordinates, and s_{cij} is the spatial softmax function for pixel a_{cij}. We then concatenate robot joint poses to this network, and pass them through two fully connected layers and output desired joint angles.
In order to provide the global policy with basic visual features, we pretrain the network to predict object pose data (this form of pretraining is common in robotics, e.g. in Levine et al. [203]). For our scooping and pouring task, we provide a similar form of pretraining, although with approximate poses. We do not use any AR markers for estimation. Instead, we sample and move the robot to a position, place the object there, and capture a training image. This naturally leads to imprecision in the training data, but is more realistic. For the digit writing task, we pretrain on an MNIST-like classification task, where we use a few digits written on a board by a human.

8.7 Results: H-NDPs for Imitation Learning

We evaluate H-NDPs on three real world tasks for imitation learning from images: Digit Writing, Scooping and Pouring. Videos can be found at https://shikharbahl.github.io/hierarchical-ndps/. One of the main focus of this work is thorough



(a) Writing Task Setup



(b) Input (c) Ours (d) Local NDP (e) GPS

Figure 8.6: Images showing the writing task setup. (a) shows the robot setup we used, a 7 DoF robot with a marker placed inside its end-effector, controlled via joint angles. (b) shows the input image (unseen at training time). (c) shows the output of our method, (d) shows the output of the local controller and (e) shows the final output of GPS [203]. We can see that our method produces a smooth and correct-looking 4.

scientific evaluation in the real world itself. This experimentation involved hundreds of hours of interaction that took several weeks on hardware to complete the real-world evaluation shown in Table 8.1 and Figure 8.7. We clearly separate training and testing scenarios for each of the tasks and describe them in the following subsections as well as in the appendix.
The goal of this empirical study is to answer following scientific questions across all the tasks:

- How much does the structure of dynamical systems contribute to the performance of H-NDPs?
- How much does iterative refinement of global policy contribute to the performance of H-NDPs?
- How much does the local-to-global structure contribute to the performance of H-NDPs?

We attempt to answer these questions by running baseline methods. Firstly, to understand the importance of dynamical systems in H-NDPs, we run comparisons against a method that uses iterative refinement as well as well as a local-to-global structure,

	#Demos	#Iter	Writing	Scooping	Pouring
No local-to-global structure:					
NDP	1x	1	0.2	0.2	0.0
Vanilla NN	1x	1	0.1	0.0	0.0
No local-to-global structure with 5x Demos:					
NDP	5x	1	0.5	0.3	0.0
Vanilla NN	5x	1	0.1	0.0	0.0
Local-to-global but no iterative refinement:					
GPS	5x	1	0.1	0.0	0.0
H-NDPs (ours)	5x	1	0.4	0.3	0.0
Both local-to-global and iterative refinement:					
GPS	1x	5	0.3	0.0	0.2
H-NDPs (ours)	1x	5	**0.8**	**0.6**	**0.3**

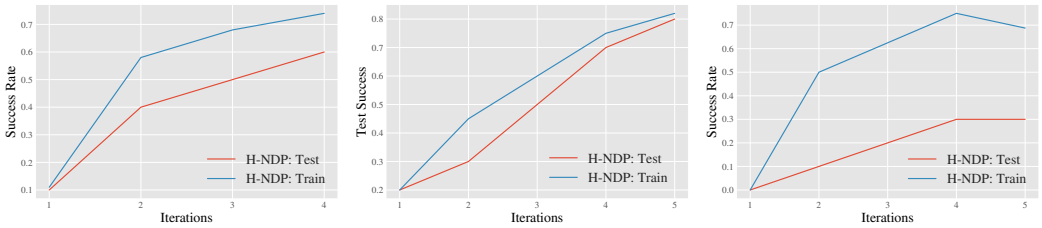
Table 8.1: Final results on the three real world tasks. We average the test success rate normalized to $[0-1]$ over 10 trials on held-out testing images/locations. We compare against vanilla NDP [25], vanilla NN imitation, and we replace NDPs in our method with vanilla neural networks (a similar method to GPS [203]). We can see that our method outperforms all the baselines substantially.

but with fully connected neural network layers instead of embedded dynamical systems. This method is our implementation of GPS [203]. For every other baseline, we also design a version of that baseline with only fully connected layers, calling it vanilla NN. Secondly, to address the question of the effect of iterative refinement, we train H-NDPs and GPS for only one iteration. To have a fair comparison, we train these baselines with 5x more demonstrations so as to provide effectively the same amount of data. However, note that these baselines have same number of interactions *but have 5x more supervision* because H-NDPs does not need more expert demonstrations after the first iteration. Finally, we test the importance of the local-to-global structure by introducing baselines that do not use it. NDP is the global policy that just learns from demonstrations. which is very similar to the method from Bahl et al. [25]. Vanilla NN is the fully connected counterpart of NDP.

8.7.1 Task 1: Digit Writing on the Whiteboard from Image Input

The goal in this task is for the robot to draw a digit on a whiteboard, given an image of the digit (ranging from 0 to 9). A dry-erase marker is attached to the robot hand. We collect 10 kinesthetic demonstrations (one for each digit 0-9) for training. We keep 10 digit images held-out which are not shown to the robot during training on which we compute the "test" success. We pretrain the global policy network using the procedure discussed in Section 8.6.3.

Role of Dynamical System Structure: In Table 8.1, we can see that H-NDPs achieve the highest test success rate, 80%. Our approach outperforms the GPS baseline by



(a) Scooping (b) Digit Writing (c) Pouring

Figure 8.7: Success rate for the three real-world tasks across iterations. Note that more iterations of the H-NDP method in fact does help in learning, both in the train and test (held-out/unseen) scenarios.

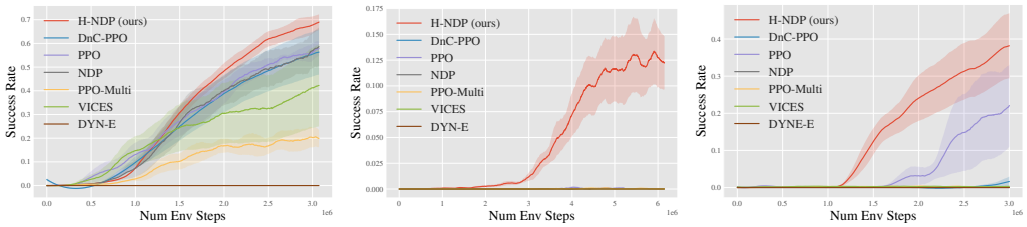
a large margin. We show a sample of the test results in Figure 8.6. The picture on the left is what the robot sees at test time, and the one the right is the final output. Compared to all the other methods, H-NDPs have the smoothest and most accurate result. Figure 8.6c shows a much smoother output by our method compared to that of GPS (Figure 8.6e). The major difference between the implementation of the two approaches is that GPS uses fully connected layers instead of dynamical-system based layers. Interestingly, when comparing all other baselines (no local-to-global structure, no iterative refinement, etc) the dynamical system based methods (in Table 8.1 these are H-NDPs and NDP) all outperform their fully connected counterparts. This clearly indicates that the role of dynamical system-based structure is crucial for writing.

Role of Iterative Refinement: We can see that in Table 8.1 that performance of H-NDPs drops without iterative refinement (with 5x more supervision performance still drops to about 50%). From Figure 8.7b, it is clear that our method benefits from iterative refinement, as the test and train success rates increase. Interestingly, at test time our method was able to capture the ”4” digit better than at training time. Despite a drop in performance, H-NDPs without iterative refinement still outperforms almost all other baselines.

Role of Local-to-Global: H-NDPs clearly outperform methods that do not employ a local-to-global structure (i.e. we set the KL penalty to 0). Both the NDP and Vanilla NN baselines perform significantly worse. This is true for methods that use 1x the demos as H-NDPs as well 5x. In fact, methods that use 1x the demos tended to fit to one demonstration and ignore the rest; i.e. Vanilla NN only output 8’s for all ten digits.

8.7.2 Task 2: Scooping from Image Input

In the scooping task, the robot has a spoon attached to its effector and its goal is to scoop almonds from a bowl. We vary the bowl locations, and the robot must infer from only from raw images where it should scoop. We have 18 distinct locations



(a) Throwing (b) Catching (c) Picking

Figure 8.8: Success rate for the three simulated RL tasks: throwing, catching and picking. Note that all these tasks are stochastic. Our method (red) outperforms all the baselines.

on the table for training and 10 distinct locations kept held-out for testing, shown in Figure 1 in the Appendix. We collect kinesthetic demonstrations on training locations. We provide the pretraining discussed in Section 8.6.3. Figure 8.4 shows a picture of this setup.

Role of Dynamical System Structure: Similarly to the writing task, it is clear from Table 8.1 that H-NDPs drastically outperforms all baselines without dynamical system-based structure (GPS, vanilla NN, etc). In fact, all such baselines ended up executing the mean trajectory. For example, wherever the bowl would be placed, the networks would output the same trajectory towards the center of table. This clearly shows that trajectory level prediction is hard for traditional networks, and that dynamical systems are important for this task.

Role of Iterative Refinement: In Figure 8.7a, we see that both training and test success rates for H-NDPs go up as we perform more iterations of refinement. For H-NDPs, iterative refinement doubles the performance. However, even without iterative refinement, H-NDPs still obtains a 30% success rate, the highest of the baselines.

Role of Local-to-Global: We can see that H-NDPs obtains a higher success rate at test time than any of the baselines that do not use the local-to-global structure (1x and 5x demos both). The performance gain from 5x to 1x demonstrations is not very high. This indicates just adding more data is not as important as the global-to-local framework. While H-NDP's success rate is 60%, even in case of failure it would go close to the bowl but not actually scoop any almonds out.

8.7.3 Task 3: Pouring from Image Input

We perform experiments on a real world visual pouring task. The robot starts with a cup of almonds in its gripper, and must pour the almonds into a target cup, without any falling out. Just like scooping, the global policy must act from camera input only, and the target cup moves around to different locations. We collect kinesthetic demonstrations for 16 training locations and keep 10 held-out locations for testing as

in the case of scooping, shown in the Appendix (Figure 1). Pretraining is discussed in Section 8.6.3 and the task setup in Figure 8.4.

Role of Dynamical System Structure: This task is inherently more difficult than the others, possibly due to the size of the cups and a lot more accuracy is needed for a successful pour. H-NDPs, however, still outperforms every other baseline, including GPS (30% vs 20%). During testing we observed that in the failed trials robot would go close to the cup but miss it marginally. On the other hand, GPS completely missed the target most of the time. The other vanilla NN baselines which do not perform iterative refinement would actually produce infeasible and dangerous behavior. This shows that dynamical system is important for the pouring task.

Role of Iterative Refinement: In Table 8.1, we can see that all of the baselines without iterative refinement have 0 success. On the other hand, Figure 8.7c shows that H-NDPs also starts with a 0% success rate, but improves via iterative refinement. Additionally, most of the baselines produced the same trajectory for every input. This shows that iterative refinement is very helpful, especially in more challenging tasks.

Role of Local-to-Global: All the methods that do not use the local-to-global framework have a success rate of 0%. Both methods that do use the local-to-global structure (as well as iterative refinement), H-NDPs and GPS, are the only ones that achieve any success. Therefore, local-to-global structure is in fact important for the pouring task.

8.8 Results: H-NDPs for Reinforcement Learning

In the RL setting, we compare our method against several competing methods. We firstly test a similar method to Ghosh et al. [117] which we call PPO-DnC. This is very similar to the original method, however it uses PPO as a base algorithm, so that it can be compared apples-to-apples with H-NDP. In another baseline, we run the NDP algorithm [25], equivalent to training the global NDP only. Additionally, we run our base RL algorithm, vanilla PPO [355] as a comparison as well. To have a fair comparison, we also consider the baselines used by Bahl et al. [25], in fact using their provided code. We compare against the multi-action PPO from Bahl et al. [25], Variable Impedance Control in End-Effector Space (VICES) [244] and Dynamics-Aware Embeddings (DYN-E) [418]. The latter two methods provide alternative parameterizations for action space: in VICES [244] the policy directly outputs parameters for a PID controller, and in DYN-E [418] an action-based encoder is learnt from environment interaction.
We can see in the RL results in Figure 8.8. We present the result of 3 random seeds run on the same codebase. We plot the success rate versus the number of environment steps taken. H-NDP, our method, outperforms all the baselines discussed above

either in terms of sample efficiency or absolute performance. This difference is especially stark for the catching task (Figure 8.8b), since the randomness is the starting position of the ball, and even a small perturbation can have a large effect on the trajectory. H-NDPs are able to capture this high level variation quite well, while the baselines cannot. In the other two tasks, shown in Figure 8.8c and Figure 8.8a, H-NDPs are still more sample efficient and have a better final performance, even though the baselines get a relatively higher performance compared to that of catching. This is likely due to the fact that randomness in throwing and picking isn't as drastic as catching. However, overall, we can clearly see that H-NDPs provide a strong performance boost, likely due to the embedded dynamical system which allows for smooth trajectories and efficient distillation of knowledge.

Chapter 9

PlayFusion: Language Conditioned Skills

9.1 Motivation

Humans reuse past experience via a broad repertoire of skills learned through experience that allows us to quickly solve new tasks and adapt across environments. For example, if one knows how to operate and load a dishwasher, many of the skills (e.g., opening the articulated door, adjusting the rack, putting objects in) will transfer seamlessly. How to learn such skills for robots and from what kind is a long-standing research question. Robotic skill abstraction has been studied as a way to transfer knowledge between environments and tasks [397, 400, 301]. It has been common to use primitives as actions in the options framework [21, 396], which are often hand-engineered [75, 389, 185, 288, 71, 273] or learned via imitation [296, 22, 295]. These allow for much more sample-efficient control but require knowledge of the task and need to be tuned for new settings. On the other hand, there have been efforts to *automatically* discover skills using latent variable models [102, 365, 251, 363, 181, 418, 227, 132]. While they can work in any setting, such models are often extremely data-hungry and have difficulty scaling to the real world due to the data quality at hand.

As a result, real-world paradigms are based on imitation or offline reinforcement learning (RL) but both these require several assumptions about the datasets. In imitation learning, human teleoperators must perform tasks near-perfectly, reset the robot to some initial state, perform the task near-perfectly again, and repeat several times. In offline RL, data is assumed to contain reward labels, which is impractical in many real-world setups where reward engineering is cumbersome. In contrast, it is much easier to collect uncurated data from human teleoperators if they are instructed only to explore, resulting in *play data* [227, 132, 69]. Learning from play (LfP) has emerged as a viable alternative to traditional data collection methods for behavior generation. It offers several advantages: (1) it is efficient because large

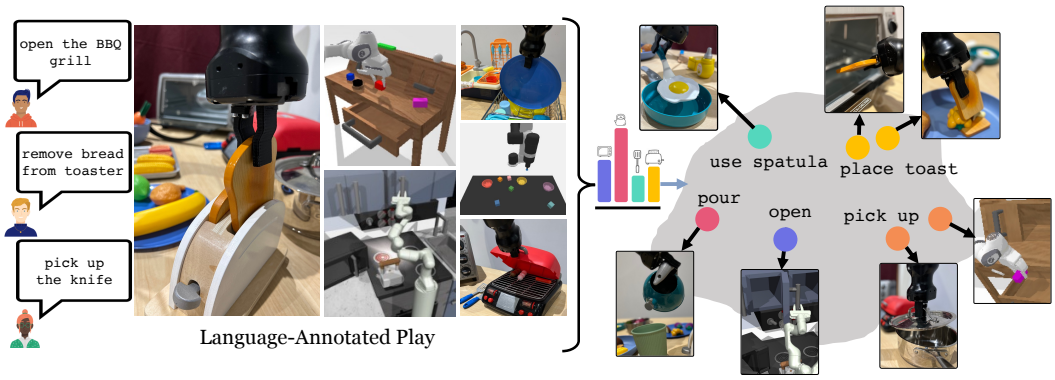


Figure 9.1: Across multiple real-world and simulated robotic settings, we show that our model can extract semantically meaningful skills from language-annotated play data. Such data is highly multimodal and offers no optimality guarantees. Video results of PlayFusion are available at https://play-fusion.github.io.

datasets of play can be collected without the need for setting up and executing perfect demonstrations, and (2) the data collected is rich and diverse because it contains a broad distribution of behavior ranging from completions of complex tasks to random meandering around the environment. An important quality of such data is that it is grounded with some semantic goal that the "player" is aiming to achieve. We believe a simple abstraction for this is *language* instructions, which can describe almost any play trajectory.

A major challenge in learning from play is that the data is highly multimodal, i.e., there are many different ways to achieve a specific goal, and given a sample from the play data, there are many different goals that could have generated it. One popular way to handle highly multimodal data is by modeling the full distribution via generative models. In recent years, there has been remarkable progress in large generative models [324, 323, 335, 39], especially in the class of diffusion models [155, 415], which have been shown to generate high-resolution images – a property well suited for vision-based robotic control. In fact, diffusion models have shown to be effective in capturing complex, continuous actions [166, 8, 415, 147, 61] in the context of robotics. However, these diffusion model-based approaches have not been empirically shown yet to work on unstructured data. We argue that the ability of diffusion models to fully capture complex data paired with their potential for text-driven generation can make them good candidates to learn from language-annotated play data.

One additional consideration is that in reality, humans only deal with a few skills. Almost every task manipulation task involves some grasping and some post-grasp movement. We believe that learning *discrete skills* will not only make the whole process more efficient but will also allow interpolation between skills and generalizations to new tasks. To address this, we propose **PlayFusion**, a diffusion model

which can learn from language-annotated play data via discrete bottlenecks. We maintain the multimodal properties of our current system while allowing for a more discrete representation of skills. Empirically, we show that our method outperforms state-of-the-art approaches on six different environments: three challenging real-world manipulation settings as well as the CALVIN [247], Franka Kitchen [132], and Ravens [442, 371] simulation benchmarks.

9.2 Related Work

Goal and Language Conditioned Skill Learning One method of specifying the task is via goal-conditioned learning, often by using the actual achieved last state as the goal [169, 12, 116, 121]. There is also recent work on using rewards to condition robot behavior [55], but this requires a reward-labeled dataset, which makes stronger assumptions than play data. Furthermore, there is a large body of work on language-conditioned learning [228, 371, 165, 7, 270, 246, 373], which specifies the task through language instructions. Instead of conditioning the policy on fully labeled and curated data, we take advantage of unstructured play data which is annotated with language in hindsight.

Learning from Play Unlike demonstrations, play data is not assumed to be optimal for any specific task as it is collected by human teleoperators who are instructed only to explore. Play-LMP and MCIL [227, 229] generate behaviors by learning motor primitives from play data using a VAE [179, 333]. RIL [132] is a hierarchical imitation learning approach and C-BeT [69] generates behaviors using a transformer-based policy and leverages action discretization to handle multimodality. LAD [446] incorporates diffusion for learning from play, but keeps several components of VAE-based approaches for encoding latent plans; we forgo those elements completely.

Behavior Modeling with Generative Models A promising architecture for behavior modeling with generative models is the diffusion probabilistic model [155, 360, 138, 137]. Diffuser [166], Decision Diffuser [8], Diffusion-QL [415] and IDQL [147] apply diffusion models to the offline reinforcement learning (RL) problem. In real-world robotic applications, Diffusion Policy [61] demonstrated strong results in visuomotor policy learning from demonstrations. Different from these works, we learn from play data containing semantic labels instead of offline RL datasets or expert demonstrations. Some approaches [70, 217] incorporate diffusion in robotics but not for generating low-level actions.

Discrete control A key challenge in robot learning is the exponentially large, continuous action space. Option or skill-based learning is appealing as it can circumvent this problem and allow the agent to learn in a structured, countable action space [375, 376, 254, 27]. Learned action discretization [360, 69] has allowed approaches to scale to complex tasks. C-BeT [69] applies real-world robotic control

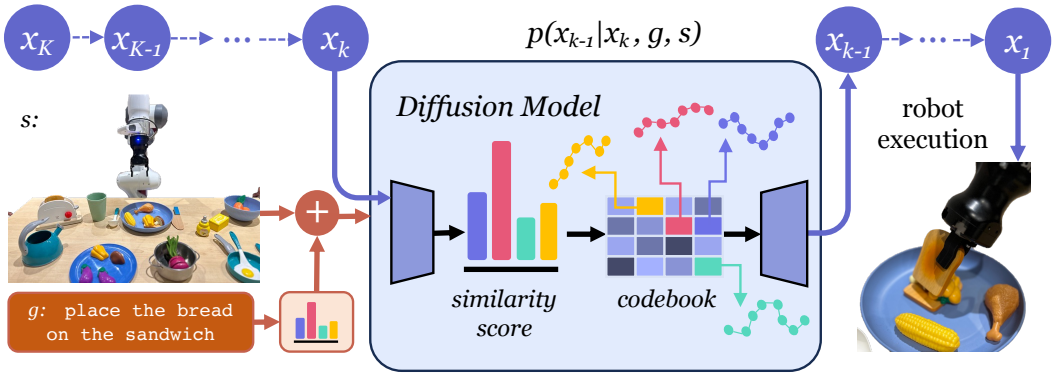


Figure 9.2: Overview of how PlayFusion extracts useful skills from language-annotated play by leveraging discrete bottlenecks in both the language embedding and diffusion model U-Net. We generate robot trajectories via an iterative denoising process conditioned on language and current state.

with transformers [55, 167, 421] to the goal-conditioned setting; [280] train a dynamics model over discrete latent states. We leverage the discrete properties of VQ-VAEs and their natural connection to language-labeled skills.

9.3 Background

Denoising Diffusion Probabilistic Models (DDPMs) DDPMs [155] model the output generation process as a denoising process, which is often referred to as Stochastic Langevin Dynamics. To generate the output, the DDPM starts by sampling x^K from a Gaussian noise distribution. It then performs a series of denoising iterations, totaling K iterations, to generate a sequence of intermediate outputs, $x^k, x^{k-1}, \cdots, x^0$. This iterative process continues until a noise-free output x^0 is produced. The denoising process is governed by the following equation:

$$x^{k-1} = \alpha(x^k - \gamma\epsilon_\theta(x^k, k) + N(0, \sigma^2 I)) \tag{9.1}$$

Here, ϵ_θ represents the noise prediction network with a learnable parameter θ, and $N(0, \sigma^2 I))$ denotes the Gaussian noise added at each iteration. This equation is used to generate intermediate outputs with gradually decreasing noise levels until a noise-free output is obtained. To train the DDPM, the process begins by randomly selecting x^0 from the training dataset. For each selected sample, a denoising iteration k is randomly chosen, and a noise ϵ^k is sampled with the appropriate variance for the selected iteration. The noise prediction network is then trained to predict the noise by minimizing:

$$\mathcal{L} = ||\epsilon^k - \epsilon_\theta(x^0 + \epsilon^k, k)||^2 \tag{9.2}$$

Discrete Representations We utilize VQ-VAE [406] inspired models in PlayFusion as they can provide a way to discretize the skill space. Given an input x, a VQ-VAE trains an encoder E to predict latent $E(x) = z$ and maintains a codebook of discrete latent codes e. The VQ layer selects j as $\arg\min_i ||z - e_i||$, finding the closest code to the embedding, which is used to reconstruct x. The training loss is

$$\mathcal{L}_{\text{VQVAE}} = \mathcal{L}_{\text{recon}}(x, D(e_j)) + ||z - sg(e_j)||_2 + ||sg(z) - e_j||_2 \tag{9.3}$$

where D is the VQ-VAE decoder. The reconstruction loss is augmented with a quantization loss, bringing chosen codebook embedding vectors e_j toward the encoder outputs in order to train the codebook, as well a loss to encourage the encoder to "commit" to one of the embeddings.

Learning from Play Data (LfP) In the LfP setting, we are given a dataset $\{(s, a)\} \in S \times A$. There are no assumptions about tasks performed in these sequences or the optimality of the data collection method. Similar to the formulation of [69], the goal is to learn a policy $\pi = S \times S \rightarrow A$ where the input is the current state s_t and goal $g = s_T$. In some cases, (including ours), the goals are instead described via language annotations.

9.4 PlayFusion: Discrete Diffusion for Language-Annotated Play

Humans do not think about low-level control when performing everyday tasks. Our understanding of skills like door opening or picking up objects has already been grounded in countless prior experiences, and we can comfortably perform these in new settings. Skills are acquired through our prior experiences – successes, failures, and everything in between. PlayFusion focuses on learning these skills through *language-annotated* play data.

However, learning from play data is still difficult as continuous control skills are not easy to identify due to several challenges: (1) data can come from multiple modalities as there are many actions that the robot could have taken at any point, (2) we want the model to acquire a vocabulary of meaningful skills and (3) we want to generalize *beyond* the training data and have the model transfer skills to new settings. To address the challenges, we leverage recent advances in **diffusion-model** large-scale text-to-image generation. Such models [61, 166, 415] can inherently model multimodality via their iterative denoising process. To effectively transfer skills to new settings, we propose a modified diffusion model with the ability to **discretize** learned behavior from language-annotated data. Figure 9.2 shows an overview of our method.

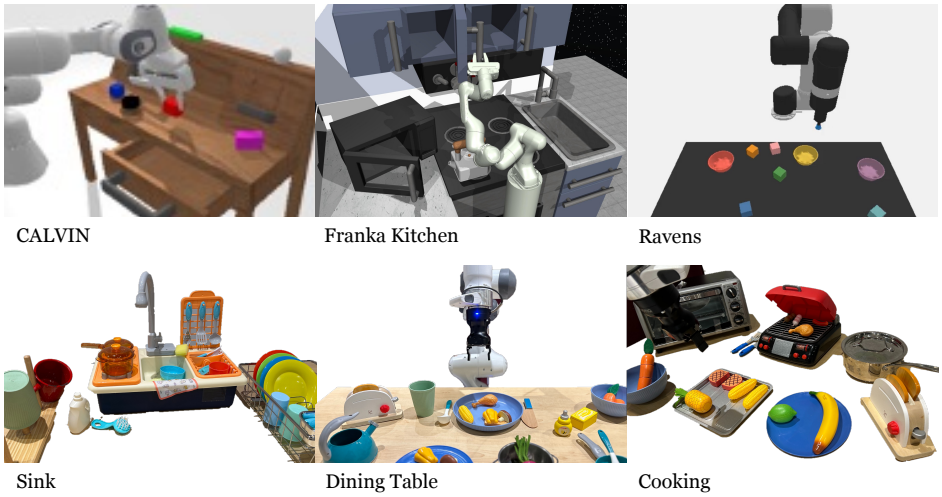


Figure 9.3: Simulated (top row) and real-world (bottom row) environments used for our evaluations. In each real-world setup, the robot is tasked with picking up one of the objects (e.g., plate, cup, carrot, bread, corn) and relocating it to a specified location (e.g., drying rack, plate, toaster, grill, pot).

9.4.1 Language Conditioned Play Data

Our setup consists of language conditioned play data [227] $D_{\texttt{play}} = \{(s_t^{(i)}, a_t^{(i)})\}_{i=1}^{N}$: long sequences of robot behavior data containing many kinds of behaviors, collected by human operators instructed to perform interesting tasks. In this setting, we assume that there is some optimality to the data, i.e. $a_t \sim \mathcal{F}(s_t, z_g)$, where z_g is a latent variable that models the intention of the operator. We thus leverage *language* labels to estimate z_g. Given a sequence $\tau = \{s_i, a_i\}_{t=k}^{H}$, τ is labeled with an instruction l which is passed into a language model [330], g_{lang}, and we refer to the embedding as z_l throughout the chapter. One can also use goal images, but we might not have access to these at test time. While our method can use any z_g as conditioning, assume that the play data has access to language annotations l. Our policy $\pi(a_t|s_t, z_l)$ contains a few simple components. We use a ResNet[154]-based visual encoder ϕ_v to encode s_t (a sequence of images) and an MLP based langauge encoder ϕ_l to downproject the language embedding z_l. The policy uses $g = [\phi_l(z_l), \phi_v(s_t)]$ as conditioning to the action decoder f_{act}. Previous approaches [227, 247] use latent variable models to deal with multimodality. We find that modelling f_{act} as a diffusion process enables us to circumvent this.

9.4.2 Multi-modal Behavior Generation via Diffusion

With f_{act}, we aim to predict robot actions given the current state, using a DDPM to approximate the conditional distribution $P(a_t|s_t)$. In our setting, we additionally condition on the goal g. Formally, we train the model to generate robot actions a_t

conditioned on goal g and current state s_t, so we modify Equations 9.1 and 9.6 to obtain:

$$a_t^{k-1} = \alpha(a_t^k - \gamma\epsilon_\theta(g, s_t, a_t^k, k) + N(0, \sigma^2 I)) \tag{9.4}$$

$$\mathcal{L} = ||\epsilon^k - \epsilon_\theta(g, s_t, a_t^0 + \epsilon^k, k)||_2 \tag{9.5}$$

We use the notation above for simplicity, but in practice, we predict a sequence of T_a future actions $a_t, \cdots, a_{t+T_a}$ instead of only the most immediate action, a technique known as *action chunking*. This is done in some recent works [61, 450] and is shown to improve temporal consistency.

9.4.3 Discrete Diffusion for Control

Moreover, humans often break down tasks into smaller skills, which are often repeatable. In fact, most tasks can be achieved with a relatively small set. On the other hand, both the latent goals that we learn as well as the action diffusion process are continuous. Making sure learnt skills are discrete can not only allow for better performance but also better generalization to new settings. However, naively enforcing discretization can lead to suboptimal behavior. We want to ensure that conditioned on a latent goal, g, action predictions from f_{act} are both multimodal and yet only represent a few modes. Thus, we propose a discrete *bottleneck* instead.
For the action generation process to represent a useful skill space, we want to enforce discreteness where the actions interact with latent goal. PlayFusion adds a vector quantization bottleneck in the diffusion process, specifically in the network $\epsilon_\theta(x) = \epsilon_\theta(g, s_t, a_t^0 + \epsilon^k, k)$. ϵ_θ is U-Net which fuses the language conditioning into the action denoising. We modify the U-Net architecture with a codebook of discrete latent codes e_u, a discrete bottleneck for the diffusion model. Given an input x the U-Net encoder produces a latent $\psi_\epsilon(x)$, which is passed into the decoder to produce $\epsilon_\theta(x) = \gamma_\epsilon(\psi_\epsilon(x))$. This bottleneck layer selects j as $\arg\min_i ||\psi_\epsilon(x) - e_i||$, finding the closest code to the embedding, which is used to reconstruct x. To account for this, we augment the training procedure with the quantization and commitment losses, similar to VQ-VAE.

Generalization via discrete language conditioning Consider an agent that has learnt skills formed from the atomic units `A`, `B`, `C` and `C`, of the form `A + B`, `B + C` and `C + D`. To truly extend its capabilities beyond the initial training data, the agent must learn to interpolate and extrapolate from these existing skills, being able to perform tasks like `A + D` that it hasn't explicitly been trained on. Given that the action generation in the diffusion process is already quantized, our hypothesis is that a discrete goal space will be synergestic and allow the policy to compose skills better. Thus, we maintain a codebook of discrete latent codes e_l for the language embeddings output by the language goal network $\phi_l(z_l)$, selecting $e_{l,j}$ which is closest to $\phi_l(z_l)$. The full loss function that we use to train PlayFusion is as follows:

$$\mathcal{L}_{PlayFusion} = ||\epsilon^k - \epsilon_\theta(x^0 + \epsilon^k, k)||_2 + \beta_1 \underbrace{||sg(\psi_\epsilon(x) - e_{u,j})||_2}_{\texttt{U-Net quantization loss}} + \beta_1 \underbrace{||\psi_\epsilon(x) - sg(e_{u,j})||_2}_{\texttt{U-Net commitment loss}}$$
$$+ \beta_2 \underbrace{||sg(\phi_l(z_l)) - e_{l,j}||_2}_{\texttt{lang. quantization loss}} + \beta_2 \underbrace{||\phi_l(z_l) - sg(e_{l,j})||_2}_{\texttt{lang. commitment loss}} \tag{9.6}$$

where β_1 and β_2 are coefficients to determine the tradeoff between covering a diversity of possible behaviors and encouraging behaviors belonging to similar skills to be brought close to each other.

Sampling from PlayFusion Given a novel language instruction at test time z', we obtain the quantized encoding $\phi_l(z')$, combining it with the visual encoding to get conditioning g'. We sample a set of actions $a_{t:t+k} \sim \mathcal{N}(0, 1)$, pass them through the discrete denoising process in Equation 9.4.

9.5 Experiments

In this section, we investigate PlayFusion and its ability to scale to complex tasks, as well as generalization to new settings. We ask the following questions: (1) Can PlayFusion allow for learning complex manipulation tasks from language annotated play data? (2) Can our method perform efficiently in the real-world setup beyond the simulated environment? (3) How well can PlayFusion generalize to out of distribution settings? (4) Can PlayFusion in fact learn discrete skills? (5) How do various design choices, such as quantization, language conditioning, etc., affect PlayFusion? We aim to answer these through experiments in three different simulation and real world settings.

Environmental Setup We test our approach across a wide variety of environments in both simulations as well as the real world. For simulation, we evaluate three benchmarks: (a) CALVIN [247], (b) Franka Kitchen [132], and (c) Language-Conditioned Ravens [442, 371]. For the real-world setup, we create three different environments: `cooking`, `dining table` and `sink`, shown in Figure 9.3. More details of the environment setup are in the supplementary.

Baselines We handle task conditioning in the same way for our method as well as all baselines, using the same visual and language encoders. We compare our method with the following baselines: (a) *Learning Motor Primitives from Play* (*Play-LMP*): Play-LMP [227] generates behaviors by learning motor primitives from play data using a VAE, which encodes action sequences into latents and then decodes them into actions. (b) *Conditional Behavior Transformer* (*C-BeT*): C-BeT [69] generates

	Simulation					Real World		
	CALVIN A	**CALVIN B**	**Kitchen A**	**Kitchen B**	**Ravens**	**Dining Table**	**Cooking**	**Sink**
C-BeT [69]	26.3 ± 0.8	23.4 ± 0.9	$\mathbf{45.6 \pm 2.3}$	$\mathbf{24.4 \pm 2.3}$	13.4	20.0	0.0	10.0
Play-LMP [227]	19.9 ± 1.0	22.0 ± 0.4	1.9 ± 1.5	0.0 ± 0.0	0.2	0.0	0.0	0.0
GCBC [227]	23.2 ± 2.0	30.4 ± 1.4	38.0 ± 3.3	15.5 ± 4.5	1.6	5.0	0.0	5.0
Ours	$\mathbf{45.2 \pm 1.2}$	$\mathbf{58.7 \pm 0.7}$	$\mathbf{47.5 \pm 2.0}$	$\mathbf{27.7 \pm 0.9}$	**35.8**	**45.0**	**30.0**	**20.0**

Table 9.1: Success rates for PlayFusion and the baselines on simulation and real-world settings. PlayFusion consistently outperforms all of the baselines.

behaviors using a transformer-based policy and leverages action discretization to handle multimodality. (c) *Goal-Conditioned Behavior Cloning* (*GCBC*): GCBC [227, 94] is conditional behavior cloning.

9.5.1 Results in Simulation and Real World

PlayFusion in simulation Table 9.1 shows success rates for PlayFusion, Play-LMP, C-BeT, and GCBC on the simulation benchmarks. On both CALVIN setups, we outperform the baselines by a wide margin, which demonstrates the effectiveness of our method in large-scale language-conditioned policy learning from complex, multimodal play data. The baselines perform comparatively better on the Franka Kitchen environments, where the training datasets are smaller and the data covers a more narrow behavior distribution and the benefit of handling multimodality is smaller; however, PlayFusion still outperforms or matches all baselines. PlayFusion also achieves significantly higher success rate than the baselines on Ravens (see appendix for per-task results), which is not as large-scale as CALVIN but covers a large portion of the state space due to the diversity of instructions.

Long horizon tasks Using the Long Horizon CALVIN evaluation suite, we test the ability of agents to stitch together different tasks, with transitioning between tasks being particularly difficult. One such long horizon chain might be "turn on the led" $\rightarrow$ "open drawer" $\rightarrow$ "push the blue block" $\rightarrow$ "pick up the blue block" $\rightarrow$ "place in slider". We rollout 128 different long horizon chains containing five instructions each and record the number of instructions successfully completed. As shown in Table 9.2, we find that PlayFusion significantly outperforms the baselines in both CALVIN A and CALVIN B. The diffu-

		No. of Instructions				
	Av. Seq Len	1	2	3	4	5
CALVIN A :						
C-BeT	0.262	25.2	1.0	0.0	0.0	0.0
Play-LMP	0.175	16.5	1.0	0.0	0.0	0.0
GCBC	0.194	19.4	0.0	0.0	0.0	0.0
CALVIN B :						
C-BeT	0.272	27.2	0.0	0.0	0.0	0.0
Play-LMP	0.117	11.7	0.0	0.0	0.0	0.0
GCBC	0.291	27.2	1.9	0.0	0.0	0.0
Ours (A)	**0.417**	**37.1**	**2.9**	**1.0**	0.0	0.0
Ours (B)	**0.611**	**54.4**	**6.0**	0.0	0.0	0.0

Table 9.2: Average sequence length on Long Horizon CALVIN and success rate for the n-th instructions.

sion process gracefully handles the multimodality of not only each individual task in the chain but also of the highly varied data the agent has seen of transitions between tasks.

Generalization in the real world Table 9.1 shows results for PlayFusion and the baselines in our real world evaluation setups. These setups are particularly challenging for two reasons: (1) inherent challenges with real-world robotics such as noisier data and constantly changing environment conditions such as lighting, and (2) they are designed to test skill-level compositional generalization. Specifically, the agents are required to compose skills `A + B` and `C + D` into `A + D`; for example, they might be trained on "pick up the carrot and place it in the pan" and "pick up the bread and put it in the toaster" and must generalize to "pick up the carrot and put it in the toaster". Our method significantly outperforms the baselines in these settings, showcasing the ability of the diffusion model in modeling complex distributions and the emergence of learned skills via the discrete bottleneck. Video results are at https://play-fusion.github.io.

9.5.2 Analysis of Discrete Representations

Learning discrete skills Table 9.3 studies the impact of our discrete bottlenecks (for Ravens results, see the appendix). The success rate is, on average, worsened with the removal of either the U-Net discretization and the language embedding discretization. We also qualitatively study whether semantically similar skills are actually mapped to similar areas of the latent space and should therefore be brought together by the discrete bottleneck. In Figure 9.4, we show that skills involving similar locations (e.g., pan) or objects (e.g., carrot) are encoded into similar embeddings. In Figure 9.4, we show the embeddings of different trajectories. The top two rows share the first skill (which is to remove the lid from the pan) and place an object in the pan. The bottom two rows share the second skill (grasping the carrot). Embeddings that contain the same skill have a similar pattern, which further indicates that the latent skill space being learned is somewhat discretized.

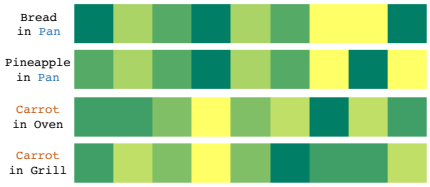


Figure 9.4: Visualization of the codebook embeddings for various real-world skills.

Balancing the discrete bottlenecks In Table 9.4, we study the effects of different β_1 and β_2 values on CALVIN A performance, i.e., the relative weightings for the additional terms in the loss function corresponding to the U-Net discretization and language embedding discretiza-

Methods	CALVIN A	CALVIN B
Ours	**45.2** $\pm$ **1.2**	**58.7** $\pm$ **0.7**
No U-Net discretiz.	**45.3** $\pm$ **2.1**	55.1 $\pm$ 1.4
No lang discretiz.	40.3 $\pm$ 1.6	54.1 $\pm$ 1.2

Table 9.3: Effect of discrete bottlenecks.

tion. We find that $\beta_1 = \beta_2 = 0.5$ results in the best performance. In general, equally weighing the four additional losses (two for U-Net and two for language) leads to improved performance over imbalanced weightings. $\beta_1 = \beta_2 = 0.5$ is also better than $\beta_1 = \beta_2 = 1$, indicating that over-incentivizing discretization can be detrimental to diffusion model learning. Further analyses can be found in the appendix.

9.5.3 Ablations of Design Choices

Effect of language model Although our method is orthogonal to the language model used, we test its sensitivity to this. As shown in Table 9.4, we find that common models such as MiniLM [330], Distilroberta [218], MPNet [384], and BERT [93] have similar performance, showing that PlayFusion is mostly robust to this design choice. We hypothesize that the discrete bottleneck applied to the language embeddings helps to achieve this robustness. CLIP [315] embeddings result in much lower success rates, most likely due the fact that Internet images may not contain similar "play data" instructions.

	Success Rate
Effect of conditioning:	
Global	54.1
Conditional Noise	40.2
Visual Pre-training	38.1
Effect of language model:	
all-MiniLM-L6-v2	47.1
all-distilroberta-v1	48.4
all-mpnet-base-v2	48.8
BERT	48.8
CLIP (ResNet50)	35.2
CLIP (ViTB32)	43.9
Loss weights (U-Net & Language) :	
0.5 & 0.5	47.1
1 & 1	45.1
0.1 & 1	45.5
1 & 0.1	43.4
0.25 & 0.75	37.7
0.75 & 0.25	43.4

Table 9.4: Effects of conditioning, language model, and loss weights.

Effect of conditioning Table 9.4 studies various different possibilities for conditioning the diffusion model generations on language and vision in CALVIN A. When working with diffusion models there are multiple different ways we can approach how to feed it goals, images of the scene etc. We found that PlayFusion is mostly robust to this, with global conditioning providing benefits for smaller models (such as those in the real world). We also attempted to condition the diffusion model noise on the goal but found that this negatively impacted performance. For the visual conditioning, we studied the effect of initializing the image encoder with large-scale pre-trained models [272]), finding that it does not help, and PlayFusion can learn the visual encoder end-to-end from scratch.
For data scaling curves and more analyses on design choices, see the appendix.

9.6 Conclusion

In this chapter, we introduced a novel approach for learning a multi-task robotic control policy using a denoising diffusion process on trajectories, conditioned on language instructions. Our method exploits the effectiveness of diffusion models in handling multimodality and introduces two discrete bottlenecks in the diffusion model in order to incentivize the model to learn semantically meaningful skills. PlayFusion does require the collection of teleoperated play data paired with after-the-fact language annotations, which still require human effort despite being already

less expensive and time-consuming to collect than demonstrations. It would be interesting to label the play data with a captioning model or other autonomous method. Furthermore, there is room for improvement in our performance on our real-world setups. Additionally, our real-world experiments could be expanded to even more complex household settings such as study rooms, bedrooms, and living rooms. Overall, our approach can significantly enhance the ability of robots to operate autonomously in complex and dynamic environments, making them more useful in a wide range of applications.

Chapter 10

Conclusions, Discussion, and Future Work

In this thesis, we discussed how to deploy robots in the wild, specifically in the context of manipulation. The main bottleneck for this has been the lack of data available. Our proposed solution is to use humans as a proxy for robots, in order to break out of the chicken-and-egg cycle. The work presented in this thesis has addressed several key challenges in enabling scaling robots to more dynamic, real-world scenarios. In Chapter 2, we focus on bridging the embodiment gap between humans and robots. We show that robots can learn by watching a single human video of humans doing multiple tasks. In Chapters 3 and 4, we aim to learn from offline, internet-scale human videos, instead of a single video showing the task. We find that the best prior for manipulation to extract from these are visual affordances. We show that we can scale these affordances to many different tasks and robot learning setups. In Chapter 5, we operationalize affordances for training general-purpose policies, across many different tasks. We do so by building a structured world model that works with both human and robot data. We move to a more flexible setting of policy pretraining in Chapter 7 and show that we can train dexterous policies with explicit human actions. Finally, in order to enable faster deployment in the real world, we are able to address the exponentially large and continuous action space in continuous control by incorporating physical priors within policies (Chapter 8) or via skill learning (Chapter 9).

10.0.1 Discussion and Future Work

There are many challenging and interesting problems ahead in robotics, as well as many open questions. While it is possible to leverage human data to compensate for the lack of robot data, to be able to deploy robots effectively we must enable them to learn by themselves, continuously, in new settings. This way the robot must be able to find new tasks, as well as ask for help when it is stuck. For tackling many

real-world challenges, it must also have a more physical understanding. This section outlines some of the open questions and future work to address these.

Can we do every task with human data? An important question will be to answer is what is the eventual role human data will play in training large-scale, deployable models and policies for robotics. Is human data just a way to pre-train models, or can we extract data granular enough to enable any task? The current hypothesis is that human videos and internet-scale data will allow for *generalization* to new environments, and potentially some task-level generalization, but to actually learn low-level actions and control, *interaction data is needed*. For example, to do the most agile or dynamic tasks, human data may not help, as current approaches cannot extract this type of action information. Thus, we must develop approaches that can seamlessly integrate data from many sources, both offline and online. For example, models must be able to handle actions from human videos, and unstructured internet data such as text or YouTube videos. Additionally, such models must be able to integrate robot data, both from the real world and simulators, to truly enable all tasks.

Is exploration important? A lot of work, including some presented in this thesis, has focused on exploration. However, with the scale that human and robot data in conjunction allow us, is it possible to cover all possible scenarios? There may be certain combinations of tasks that are not possible for humans or the people collecting robot data – which is why eventually the robot will need to *improve* and *explore* to discover new settings. Oftentimes, it is difficult to even define the task. Even though action and behavior priors from human videos can help, exploration methods, including our previous work [269, 309, 267], can be inefficient. Even if exploration is bootstrapped from human videos, for example in our previous work [249, 23, 250], the robot must be able to find new things, practice when it sees them, and possibly *adapt*. When faced with a difficult task, humans usually look for answers in the largest dataset available: the internet. When in a difficult position, a robot should also be able to search the internet for possible solutions. To achieve this, the robot will need to (1) try to discover new goals by searching the internet or other databases, (2) learn how to break a difficult task down into important sub-parts, (3) learn to practice with such goals and (4) ask for help when it is stuck – all in an iterative manner.

How can we learn about the physics of the world? For many important tasks, knowledge of touch and forces is crucial, especially for settings that require high precision (for example assembly). This type of information is difficult to obtain for human videos and is traditionally acquired via interaction, usually with the help of sensors: force-torque, tactile, audio, etc. This type of data is not abundant on the internet, thus it might be difficult to use the approaches presented in this thesis.

Additionally, data across different sensors might vary drastically: for example, tactile sensors give widely different levels of granularity of information. A solution to this challenge might be to build a centralized repository of different sensors on the same human and robot data and use that to train predictive models, which can either hopefully serve as foundations or be used to get some coarse labels. There is enough information in videos to be able to be able to draw at least coarse visual correspondences to other modalities. Even if the above dataset and models exist, it is also unclear how to exactly incorporate this physical information into current policy learning approaches.

Are current deep learning methods sufficient for robotics? Advances in deep learning have been crucial in move learning-based robotics. We are closer than ever to the longstanding dream of having robots autonomously do many daily tasks for us. However, many of the optimization methods and networks we use were in fact not designed with robotics in mind. Embodied agents and tasks are very different from computer vision or natural language setups. Thus, is it possible to incorporate inductive biases into the networks that encode certain agent-centric properties – for example information about objects, physics, or dynamics of different robots? Some of the work in this thesis (Chapter 8) has incorporated the structure of dynamical systems within policies. It is important to design both policies and optimization methods which heavily consider these various physical concepts. This would allow robotic agents to handle the complexity of new tasks, and to generalize better. These agents could potentially share knowledge across many tasks. Efforts have been made to make networks aware of objects, 3D scenes, or transformations [457, 327, 441, 212]. However, it is difficult to scale these approaches to multiple tasks. An alternative approach would be to analyze existing networks and understand what physical properties exist within them, which could give us a better understanding on what inductive biases are important.

Can we design better affordances? Consider the complex task of cleaning a kitchen with a mop. This demands an intricate understanding of the mop's materials, the deformability of the mop head when pressure is applied, how the water will spread across various surfaces, and the specific motions that will produce a clean surface. Moreover, a mope is not restricted to the task of cleaning floors: for instance, it could be used to dust industrial equipment, move items, or clean ceilings/walls. The visual affordances presented in [24, 250] (Chapters 3, 4 and 5) cannot capture tasks with complex motions, such as mopping, or writing on whiteboard. Additionally, settings with multiple action modalities are difficult to express in the visual affordance setup [24] in Chapter 3. Can we utilize passive human datasets to learn such a multi-functional and multi-sensory foundational model for robotics? Fortunately, human video datasets are usually accompanied by different levels of granularity of semantic information and actions: "cooking pasta" or "taking pasta out of box", "pouring

water into a bowl", and "stirring pasta". This information, combined with state-of-the-art action extraction approaches that leverage large-scale physically grounding datasets [90] can allow for more robust and accurate action extraction. The ideal affordance model can both leverage underlying intent and high-level semantics and the specific kinematics of the agent in the human data. Given any scene and a task description at test-time, the goal is to be able to predict the desired contacts in 3D and the exact trajectory that has to be followed by a robot arm. Instead of a strong prior, that requires practice to achieve the task [23, 248, 250, 24], this model should be zero-shot.

Final Thoughts Overall, there is potential for many exciting advances to be made in robot learning, especially when it comes to deploying in the wild. Over the years, many advances in learning-based methods (in computer vision, deep learning, NLP, RL, etc) have given us the tools to work with. Generalization to new tasks and settings has always been the goal. While learning new tasks is difficult, with all of the learning tools at our disposal, doing *one task* extremely well, in every setting, is within reach, especially with the right type of initialization (i.e., human data), action formulation (affordances for example), and practice or improvement scheme.

Bibliography

[1] Allegro hand. https://www.wonikrobotics.com/research-robot-hand. 85, 93

[2] Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/. 42, 84

[3] xarm6 by ufactory. https://www.ufactory.cc/xarm-collaborative-robot. XI, XV, 87, 90, 92

[4] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what?-anticipating temporal occurrences of activities. In *CVPR*, 2018. 29, 71

[5] 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. 43

[6] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, 2016. 42

[7] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 119

[8] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022. 118, 119

[9] 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. 1, 7

[10] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017. 100

[11] Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable mpc for end-to-end planning and control. In *NeurIPS*, 2018. 29, 100

[12] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017. 1, 7, 34, 119

[13] 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. 42

[14] 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. *IJRR*, 2020. 43, 55

[15] Anonymous. Title anonymized for review. In *In Submission*, 2024. 5

[16] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 33

[17] 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. 29, 56

[18] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation, 2022. 83

[19] Haruhiko Asada and J-JE Slotine. *Robot analysis and control*. John Wiley & Sons, 1991. 43

[20] Christopher G. Atkeson and Juan Carlos Santamaria. A comparison of direct and model-based reinforcement learning. In *ICRA*, 1997. 100

[21] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. 55, 99, 117

[22] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Hierarchical neural dynamic policies. *RSS*, 2021. 5, 83, 84, 89, 91, 93, 117

[23] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *RSS*, 2022. VIII, 5, 7, 8, 29, 33, 34, 37, 46, 58, 84, 130, 132

[24] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. 2023. 5, 41, 44, 54, 58, 71, 131, 132

[25] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *NeurIPS*, 2020. XI, XV, 5, 29, 55, 83, 84, 89, 97, 98, 99, 102, 105, 110, 112, 115

[26] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, 2016. 28, 71

[27] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013. 119

[28] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016. 34

[29] Gedas Bertasius, Hyun Soo Park, Stella X Yu, and Jianbo Shi. First person action-object detection with egonet. *arXiv preprint arXiv:1603.04908*, 2016. 28

[30] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, pages 4009–4018, 2021. 87

[31] Raunaq Bhirangi, Tess Hellebrekers, Carmel Majidi, and Abhinav Gupta. Reskin: versatile, replaceable, lasting tactile skins. *arXiv preprint arXiv:2111.00071*, 2021. 43

[32] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 70

[33] 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. 83

[34] 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. 10

[35] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv:1606.01540*, 2016. 1, 8

[36] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 55

[37] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022. 70

[38] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020. 2, 54

[39] 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. 29, 82, 118

[40] Arthur E Bryson and Yu-Chi Ho. *Applied optimal control: optimization, estimation, and control*. Routledge, 2018. 33

[41] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *SIGKDD*, 2006. 98, 106

[42] Kaylee Burns, Zach Witzel, Jubayer Ibn Hamid, Tianhe Yu, Chelsea Finn, and Karol Hausman. What makes pre-trained visual representations successful for robust manipulation? *ArXiv*, 2023. 69, 71, 72, 75

[43] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 2007. 29

[44] Sylvain Calinon, Irene Sardellitti, and Darwin G. Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. *IROS*, 2010. 99

[45] 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. 87

[46] 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. 86

[47] Zhe Cao, Ilija Radosavovic, Angjoo Kanazawa, and Jitendra Malik. Reconstructing hand-object interactions in the wild. In *CVPR*, pages 12417–12426, 2021. 29

[48] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CVPR*, 2021. 70, 71, 75, 78

[49] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 11, 17

[50] 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. 2, 11, 29, 58, 84

[51] Boyuan Chen, Pieter Abbeel, and Deepak Pathak. Unsupervised learning of visual 3d keypoints for control. *arXiv preprint arXiv:2106.07643*, 2021. 11

[52] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *CoRL*, 2020. 70

[53] Lili Chen, Shikhar Bahl, and Deepak Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *CoRL*, 2023. 5

[54] 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. 49

[55] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 119, 120

[56] Nutan Chen, Maximilian Karl, and Patrick Van Der Smagt. Dynamic movement primitives in latent space of time-dependent variational autoencoders. In *International Conference on Humanoid Robots (Humanoids)*, 2016. 99

[57] 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. 43

[58] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018. 100

[59] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 11

[60] 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. 82

[61] 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. 118, 119, 121, 123

[62] Vivian Chu, Tesca Fitzgerald, and Andrea L Thomaz. Learning object affordances by leveraging the combination of human-guidance and self-exploration. In *International Conference on Human-Robot Interaction*, 2016. 29

[63] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *ArXiv e-prints*, May 2018. 100

[64] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, pages 4754–4765, 2018. 56

[65] Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan

Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. ICRA, 2024. 70

[66] Adam Conkey and Tucker Hermans. Active learning of probabilistic movement primitives. *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019. 99

[67] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark. https://github.com/open-mmlab/mmaction2, 2020. 86

[68] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020. 99

[69] Zichen Jeff Cui, Yibin Wang, Nur Muhammad, Lerrel Pinto, et al. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022. 117, 119, 121, 124, 125

[70] Yilun Dai, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *arXiv preprint arXiv:2302.00111*, 2023. 119

[71] Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *NeurIPS*, 2021. 29, 55, 117

[72] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, , Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *IJCV*, 2022. 28, 29

[73] 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. X, 4, 11, 42, 44, 59, 60, 63, 71, 82, 83, 84, 86, 89, 93

[74] 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 *ECCV*, 2018. 28, 29, 71

[75] Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 2016. 55, 99, 100, 117

[76] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard Higgins, Sanja Fidler, David Fouhey, and Dima Damen. Epic-kitchens visor benchmark: Video segmentations and object relations. In *NeurIPS Track on Datasets and Benchmarks*, 2022. 28

[77] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard Higgins, Sanja Fidler, David Fouhey, and Dima Damen. Epic-kitchens visor benchmark: Video segmentations and object relations. *NeurIPS*, 2022. 71

[78] Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *ICCV*, 2017. 29

[79] 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. 84

[80] 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 *CVPR*, 2013. 10

[81] Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2634–2641, 2013. 42, 71, 84

[82] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019. 54, 70

[83] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pages 2071–2084. PMLR, 2021. 70

[84] 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. 69, 71, 72, 75, 76

[85] 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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 1, 8, 70, 84, 91, 92, 93

[86] Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration. *arXiv preprint arXiv:2008.07682*, 2020. 29, 41

[87] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on machine learning (ICML)*, 2011. 56, 100

[88] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. 100

[89] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Found. Trends Robot*, 2013. 56, 100

[90] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023. 132

[91] 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. 69, 70, 75

[92] Eadom Dessalene, Chinmaya Devaraj, Michael Maynord, Cornelia Fermuller, and Yiannis Aloimonos. Forecasting action through contact representations from first person video. *TPAMI*, 2021. 28

[93] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018. 2, 82, 127

[94] Yiming Ding, Carlos Florensa, Mariano Phielipp, and Pieter Abbeel. Goal-conditioned imitation learning. *arXiv preprint arXiv:1906.05838*, 2019. 125

[95] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016. 72

[96] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017. 2, 8

[97] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *CVPR*, pages 1801–1810, 2019. 10

[98] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018. 54, 56

[99] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021. 29, 54

[100] D Eigen and R Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. corr, abs/1411.4734. *arXiv preprint arXiv:1411.4734*, 2014. 28, 71

[101] Sarah Elliott, Zhe Xu, and Maya Cakmak. Learning generalizable surface cleaning actions from demonstration. In *International Symposium on Robot and Human Interactive Communication*, 2017. 29

[102] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. 117

[103] 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. 10, 11, 42, 84

[104] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 11, 17, 18

[105] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017. 56

[106] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368, 2017. 2, 8

[107] 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. 87

[108] David F Fouhey, Vincent Delaitre, Abhinav Gupta, Alexei A Efros, Ivan Laptev, and Josef Sivic. People watching: Human actions as a cue for single view geometry. In *ECCV*, 2012. 28

[109] David F Fouhey, Xiaolong Wang, and Abhinav Gupta. In defense of the direct perception of affordances. *arXiv preprint arXiv:1505.01085*, 2015. 41

[110] Justin Fu, John Co-Reyes, and Sergey Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. In *NeurIPS*, pages 2577–2587, 2017. 16

[111] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 35

[112] Antonino Furnari, Sebastiano Battiato, Kristen Grauman, and Giovanni Maria Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 2017. 28, 71

[113] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *ICCV*, 2019. 29

[114] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *TPAMI*, 2020. 29, 71

[115] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. Red: Reinforced encoder-decoder networks for action anticipation. *arXiv preprint arXiv:1707.04818*, 2017. 29, 71

[116] Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals without reinforcement learning. *arXiv preprint arXiv:1912.06088*, 2019. 34, 119

[117] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. *arXiv preprint arXiv:1711.09874*, 2017. 100, 106, 109, 115

[118] James Jerome Gibson. *The senses considered as perceptual systems*, volume 2. 3, 27, 71

[119] JJ Gibson. The ecological approach to visual perception. *Houghton Mifflin Comp*, 1979. 3, 27, 71

[120] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *ICCV*, 2021. 29, 71

[121] Anirudh Goyal, Abram Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C Humphreys, Ksenia Konyushova, et al. Retrieval-augmented reinforcement learning. In *International Conference on Machine Learning*, pages 7740–7765. PMLR, 2022. 119

[122] Ankit Goyal, Arsalan Mousavian, Chris Paxton, Yu-Wei Chao, Brian Okorn, Jia Deng, and Dieter Fox. Ifor: Iterative flow minimization for robotic object rearrangement. In *CVPR*, 2022. 34

[123] Mohit Goyal, Sahil Modi, Rishabh Goyal, and Saurabh Gupta. Human hands as probes for interactive object understanding. In *CVPR*, 2022. IX, 3, 27, 28, 33, 35, 36, 37, 41, 44, 54, 58, 71

[124] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, pages 5842–5850, 2017. 10, 11, 17

[125] 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. 42, 82, 84

[126] 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. XI, 68, 69, 71, 75, 78

[127] 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, 42, 44, 82, 83, 84

[128] 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 *CVPR*, 2022. 28, 29

[129] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *NeurIPS*, 2019. 99

[130] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018. 1, 8, 70

[131] Abhinav Gupta, Scott Satkin, Alexei A Efros, and Martial Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011. 28, 71

[132] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019. 117, 119, 124

[133] Reymundo A Gutierrez, Vivian Chu, Andrea L Thomaz, and Scott Niekum. Incremental task modification via corrective demonstrations. In *ICRA*, 2018. 29

[134] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. 56, 57

[135] 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. 42

[136] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. 1

[137] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 56, 57, 119

[138] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018. 56, 57, 119

[139] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018. 56

[140] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020. X, 56, 57, 60, 62, 64

[141] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020. 70

[142] 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. 84

[143] 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 *ICRA*, 2020. 43

[144] 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. 45

[145] 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 *ICRA*, 2020. 84, 86

[146] Nicklas Hansen, Yixin Lin, Hao Su, Xiaolong Wang, Vikash Kumar, and Aravind Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. *arXiv preprint arXiv:2212.05698*, 2022. 56

[147] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023. 118, 119

[148] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *CVPR*, 2018. 11, 17

[149] M Hassanin, S Khan, and M Tahtali. Visual affordance and function understanding: a survey. arxiv. *arXiv preprint arXiv:1807.06775*, 2018. 28, 30, 71

[150] 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. 68, 69, 70, 71, 72, 75, 91

[151] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 11

[152] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 72

[153] 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. 82

[154] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 32, 49, 63, 79, 86, 88, 89, 93, 122

[155] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 118, 119, 120

[156] De-An Huang and Kris M Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*, 2014. 29, 71

[157] Yanlong Huang, Leonel Rozo, João Silvério, and Darwin G Caldwell. Kernelized movement primitives. *The International Journal of Robotics Research*, 2019. 99

[158] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 2013. 84, 96, 98, 101

[159] 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. 42, 84

[160] Ashesh Jain, Avi Singh, Hema S Koppula, Shane Soh, and Ashutosh Saxena. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *ICRA*, 2016. 29, 71

[161] Unnat Jain, Svetlana Lazebnik, and Alexander G Schwing. Two can play this game: Visual dialog with discriminative question generation and answering. In *CVPR*, 2018. 29

[162] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander G. Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020. 29

[163] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G. Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019. 34

[164] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *CVPR*, 2022. 55

[165] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022. 55, 119

[166] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022. 118, 119, 121

[167] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021. 120

[168] 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. 15, 41, 46

[169] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993. 119

[170] 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. 29, 42, 53, 55, 70, 83, 97

[171] 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. 1, 7, 42, 55

[172] 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. 10, 42, 71, 84

[173] Aditya Kannan, Kenneth Shaw, Shikhar Bahl, Pragna Mannam, and Deepak Pathak. Deft: Dexterous fine-tuning for real-world hand policies. *CoRL*, 2023. 5, 71

[174] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *IJRR*, 2011. 30

[175] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *ICRA*, 2011. 42

[176] Charles C Kemp, Aaron Edsinger, Henry M Clever, and Blaine Matulevich. The design of stretch: A compact, lightweight mobile manipulator for indoor human environments. *arXiv preprint arXiv:2109.10892*, 2021. 18

[177] 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. 43

[178] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 74, 76

[179] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 15, 119

[180] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 46

[181] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019. 117

[182] 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. 45, 48

[183] Jens Kober, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust robot movements to new situations. *RSS*, 2011. 96

[184] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Advances in neural information processing systems*, 21, 2008. 42

[185] Jens Kober and Jan Peters. Learning motor primitives for robotics. In *ICRA*, 2009. 55, 99, 100, 117

[186] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *TPAMI*, 2015. 29, 71

[187] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *IROS*, 2010. 55, 96, 98

[188] 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. 42

[189] Tejas Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *arXiv preprint arXiv:1906.11883*, 2019. 11

[190] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. 1

[191] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *RSS*, 2021. 55

[192] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *arXiv preprint arXiv:2109.10813*, 2021. 29

[193] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020. 18, 21, 29, 91, 92, 93

[194] 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. 84

[195] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014. 29, 71

[196] S. M. Lavalle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions*, 2000. 30

[197] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 30

[198] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019. 56

[199] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv:1804.01523*, 2018. 54

[200] 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. 11, 84

[201] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020. 1

[202] Sungho Lee, Seoung Wug Oh, DaeYeun Won, and Seon Joo Kim. Copy-and-paste networks for deep video inpainting. In *ICCV*, 2019. VIII, 16, 17

[203] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016. XII, XV, 1, 29, 33, 53, 83, 97, 100, 110, 111, 112

[204] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 68, 70

[205] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML*, 2013. 42, 98, 100, 106

[206] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018. 70

[207] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In *ISER*, 2016. 1, 8, 42

[208] 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 *ICRA*, 2019. 84

[209] Yin Li, Miao Liu, and James M Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *ECCV*, 2018. 28

[210] 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. *arXiv preprint arXiv:1509.02971*, 2015. 42

[211] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *CVPR*, pages 7083–7093, 2019. 11

[212] Yen-Chen Lin, Pete Florence, Andy Zeng, Jonathan T Barron, Yilun Du, Wei-Chiu Ma, Anthony Simeonov, Alberto Rodriguez Garcia, and Phillip Isola. Mira: Mental imagery for robotic affordances. In *Conference on Robot Learning*, pages 1916–1927. PMLR, 2023. 131

[213] 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. 43

[214] Iou-Jen Liu, Unnat Jain, Raymond A Yeh, and Alexander Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In *ICML*, 2021. 34

[215] Miao Liu, Siyu Tang, Yin Li, and James M Rehg. Forecasting human-object interaction: joint prediction of motor attention and actions in first person video. In *ECCV*, 2020. 29

[216] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *CVPR*, 2022. IX, 3, 29, 30, 33, 36, 41, 44, 54, 58, 71

[217] W Liu, Y Du, T Hermans, S Chernova, and C Paxton. Structdiffusion: Language-guided 304 creation of physically-valid structures using unseen objects. *arXiv preprint arXiv:2211.04604*, 305:2, 2022. 119

[218] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 127

[219] 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 *CVPR*, 2022. 28, 71

[220] 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. 41, 44

[221] 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. 42

[222] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *ICRA*, 2018. 10

[223] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 11

[224] 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. 9, 84, 85

[225] Dylan P Losey, Andrea Bajcsy, Marcia K O'Malley, and Anca D Dragan. Physical interaction as communication: Learning robot objectives online from human corrections. *IJRR*, 2022. 29

[226] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 71

[227] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. *arXiv preprint arXiv:1903.01973*, 2019. 117, 119, 122, 124, 125

[228] Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020. 119

[229] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020. 119

[230] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017. 43

[231] 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. 27, 42

[232] 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. 71

[233] Guilherme J Maeda, Gerhard Neumann, Marco Ewerton, Rudolf Lioutikov, Oliver Kroemer, and Jan Peters. Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks. *Autonomous Robots*, 2017. 29

[234] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017. 1, 7, 97

[235] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? 2023. 68, 70, 71, 75, 78

[236] 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. 1, 8, 83

[237] Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning (CoRL)*, 2021. 43

[238] 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. 43

[239] 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. 56, 71, 84

[240] 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. 29

[241] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021. 76

[242] Pragna Mannam, Kenneth Shaw, Dominik Bauer, Jean Oh, Deepak Pathak, and Nancy Pollard. A framework for designing anthropomorphic soft hands through interaction, 2023. 41

[243] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022. 43

[244] 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. 42, 55, 99, 115

[245] Esteve Valls Mascaro, Hyemin Ahn, and Dongheui Lee. Intention-conditioned long-term human egocentric action forecasting@ ego4d challenge 2022. *arXiv preprint arXiv:2207.12080*, 2022. 29, 71

[246] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022. 119

[247] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. 119, 122, 124

[248] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Alan: Autonomously exploring robotic agents in the real world. *arXiv preprint arXiv:2302.06604*, 2023. 34, 42, 132

[249] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Alan: Autonomously exploring robotic agents in the real world. *ICRA*, 2023. 130

[250] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *RSS*, 2023. 5, 130, 131, 132

[251] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020. 117

[252] Lina Mezghan, Sainbayar Sukhbaatar, Thibaut Lavril, Oleksandr Maksymets, Dhruv Batra, Piotr Bojanowski, and Karteek Alahari. Memory-augmented reinforcement learning for image-goal navigation. In *IROS*, 2022. 34

[253] Himangi Mittal, Pedro Morgado, Unnat Jain, and Abhinav Gupta. Learning state-aware visual representations from audible interactions. In *NeurIPS*, 2022. 29

[254] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. 37, 119

[255] Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L Sidner, and Daniel Miller. Interactive hierarchical task learning from a single demonstration. In *International Conference on Human-Robot Interaction*, 2015. 29

[256] Mathew Monfort, Bowen Pan, Kandan Ramakrishnan, Alex Andonian, Barry A McNamara, Alex Lascelles, Quanfu Fan, Dan Gutfreund, Rogerio Feris, and Aude Oliva. Multi-moments in time: Learning and interpreting models for multi-action video understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 17

[257] 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. 42

[258] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013. 55, 98

[259] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. *arXiv preprint arXiv:2011.06431*, 2020. 10

[260] Austin Myers, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance detection of tool parts from geometric features. In *ICRA)*, 2015. 28, 71

[261] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 2013. 96

[262] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018. 100

[263] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017. 56

[264] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *CoRL*, 2020. 56

[265] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *ICCV*, 2019. 3, 27, 28, 30, 36, 41, 44, 54, 58, 71

[266] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. In *CVPR*, 2020. 28

[267] Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. In *CoRL*, 2020. 130

[268] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *ICRA*, 2017. 34

[269] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018. 34, 42, 70, 82, 86, 130

[270] 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. 119

[271] 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. XIV, 27, 29, 36, 37, 38, 44, 49, 54, 56, 61, 63, 68, 71, 72, 75, 82, 84, 86, 88, 89, 91, 93

[272] 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. 42, 64, 127

[273] Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *ICRA*, 2022. 29, 117

[274] Klaus Neumann and Jochen Steil. Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems*, 2015. 100

[275] Anh Nguyen, Dimitrios Kanoulas, Luca Muratore, Darwin G Caldwell, and Nikos G Tsagarakis. Translating videos to commands for robotic manipulation with deep recurrent neural networks. In *ICRA*, 2018. 11

[276] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 11

[277] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 68, 72, 75

[278] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 2018. 1

[279] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *ICML*, 2017. 34

[280] Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aaron Van Den Oord, and Oriol Vinyals. Vector quantized models for planning. In *International Conference on Machine Learning*, pages 8302–8313. PMLR, 2021. 120

[281] Rok Pahic, Andrej Gams, Aleš Ude, and Jun Morimoto. Deep encoder-decoder networks for mapping raw images to dynamic movement primitives. *ICRA*, 2018. 99, 103

[282] Tian Pan, Yibing Song, Tianyu Yang, Wenhao Jiang, and Wei Liu. Videomoco: Contrastive video representation learning with temporally adversarial examples. In *CVPR*, 2021. 11

[283] Jyothish Pari, Nur Muhammad, Sridhar Pandian Arunachalam, Lerrel Pinto, et al. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 11, 29, 36, 84

[284] Simone Parisi, Hany Abdulsamad, Alexandros Paraschos, Christian Daniel, and Jan Peters. Reinforcement learning vs human programming in tetherball robot games. In *IROS*, 2015. 55, 99

[285] Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. Interesting object, curious agent: Learning task-agnostic exploration. *NeurIPS*, 2021. 34

[286] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. *arXiv preprint arXiv:2203.03580*, 2022. 29

[287] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009. 10, 84, 101

[288] Peter Pastor, Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, and Stefan Schaal. Skill learning and task outcome prediction for manipulation. In *ICRA*, 2011. 55, 99, 100, 117

[289] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 16, 34, 42

[290] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 2, 8, 34

[291] 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. 84, 85

[292] 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. 43

[293] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 63

[294] Nicolas Perrin and Philipp Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 2016. 100

[295] Karl Pertsch, Ruta Desai, Vikash Kumar, Franziska Meier, Joseph J Lim, Dhruv Batra, and Akshara Rai. Cross-domain transfer via semantic skill imitation. *arXiv preprint arXiv:2212.07407*, 2022. 117

[296] Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pages 188–204. PMLR, 2021. 117

[297] Affan Pervez and Dongheui Lee. Learning task-parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, 2018. 99

[298] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 42

[299] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *ICML*, 2007. 63

[300] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. In *International Conference on Humanoid Robots*, 2003. 1, 55, 98

[301] Marc Pickett and Andrew G Barto. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML*, volume 19, pages 506–513, 2002. 117

[302] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016. 82

[303] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *ICRA*, 2016. 29, 53, 97

[304] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016. 68, 70

[305] Sören Pirk, Mohi Khansari, Yunfei Bai, Corey Lynch, and Pierre Sermanet. Online object representations with contrastive learning. *arXiv preprint arXiv:1906.04312*, 2019. 11

[306] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 70, 75

[307] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *NeurIPS*. Morgan-Kaufmann, 1988. 83

[308] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 10

[309] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019. 1, 7, 34, 130

[310] 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. 42

[311] 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. 29, 84, 98

[312] Yuzhe Qin, Hao Su, and Xiaolong Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation, 2022. 83

[313] 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. In *ECCV*, 2022. 29, 56

[314] 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. In *ECCV*, 2022. 84

[315] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 127

[316] 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. 2, 45, 54, 75, 78

[317] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. *CoRL*, 2022. 27

[318] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. *CoRL*, 2022. 56, 68, 71, 75

[319] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *L4DC*, 2021. 29

[320] Akshara Rai, Giovanni Sutanto, Stefan Schaal, and Franziska Meier. Learning feedback terms for reactive planning and control. In *ICRA*, 2017. 99

[321] 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. 43

[322] Santhosh K Ramakrishnan, Dinesh Jayaraman, and Kristen Grauman. An exploration of embodied visual exploration. *arXiv preprint arXiv:2001.02192*, 2020. 34

[323] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2, 118

[324] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 118

[325] Muhammad Asif Rana, Anqi Li, Dieter Fox, Byron Boots, Fabio Ramos, and Nathan Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. *arXiv preprint arXiv:2005.13143*, 2020. 100

[326] Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. Imitation learning for locomotion and manipulation. In *International Conference on Humanoid Robots*, 2007. 1, 10

[327] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018. 29, 97, 99, 131

[328] Harish Ravichandar, Iman Salehi, and Ashwin Dani. Learning partially contracting dynamical systems from demonstrations. In *CoRL*, 2017. 100

[329] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. 56

[330] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. 122, 127

[331] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 28:91–99, 2015. VIII, 12, 16

[332] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014. 15, 46

[333] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 46, 119

[334] Nicholas Rhinehart and Kris M Kitani. Learning action maps of large environments via first-person vision. In *CVPR*, 2016. 29, 71

[335] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 118

[336] 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. XI, 9, 12, 42, 45, 84, 85, 86, 87

[337] 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. 10, 42, 44, 45, 69, 71, 73, 80, 84, 85, 86, 88

[338] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *ICCV Workshops*, 2021. 29

[339] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 10, 75

[340] Anirban Roy and Sinisa Todorovic. A multi-scale cnn for affordance segmentation in rgb images. In *ECCV*, 2016. 28, 71

[341] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999. 61

[342] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013. 17

[343] Elmar Rückert and Andrea d'Avella. Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. *Frontiers in computational neuroscience*, 2013. 100

[344] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015. 98, 106

[345] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8), 1964. 13

[346] Johann Sawatzky, Abhilash Srikantha, and Juergen Gall. Weakly supervised affordance detection. In *CVPR*, 2017. 28, 71

[347] Rosario Scalise, Jesse Thomason, Yonatan Bisk, and Siddhartha Srinivasa. Improving robot success detection using static object data. In *IROS*. IEEE, 2019. 11

[348] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999. 1, 75

[349] 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, 84, 96, 98, 101

[350] Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997. 69

[351] 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. 84

[352] Jürgen Schmidhuber. Curious model-building control systems. In [*Proceedings*] *1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991. 57

[353] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers, 1991. 57

[354] 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. 87

[355] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017. 1, 104, 107, 115

[356] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from pixels. 2017. 2, 17, 18, 21, 29, 82

[357] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018. 68

[358] Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 486–487. IEEE, 2017. 10

[359] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016. 1, 7, 10, 17, 58

[360] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. 119

[361] Rutav M Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. In *ICML*, 2021. 27, 29

[362] Dandan Shan, Jiaqi Geng, Michelle Shu, and David Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, 2020. XI, XIV, 10, 12, 27, 29, 31, 41, 42, 44, 54, 58, 63, 69, 71, 72, 73, 75, 78, 79, 80, 84

[363] Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pages 8624–8633. PMLR, 2020. 117

[364] 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):1419–1434, 2021. 2, 11, 29, 58, 84

[365] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019. 117

[366] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *arXiv:1810.07121*, 2018. 10

[367] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019. 2, 9, 10, 17, 29, 33, 58, 84

[368] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand:low-cost, efficient, and anthropomorphic hand for robot learning. *RSS*, 2023. XIV, 79

[369] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *CoRL*, 2022. 5, 29, 43, 56, 58, 71

[370] Kenneth Shaw and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *in Submission, ICRA*, 2023. 88, 93

[371] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *CoRL*, 2022. 34, 55, 119, 124

[372] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022. 55

[373] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023. 119

[374] 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. 43

[375] D. Silver, Julian Schrittwieser, K. Simonyan, Ioannis Antonoglou, Aja Huang, A. Guez, T. Hubert, L. Baker, Matthew Lai, A. Bolton, Yutian Chen, T. Lillicrap, F. Hui, L. Sifre, George van den Driessche, T. Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 2017. 119

[376] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016. 119

[377] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 91, 93

[378] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *RSS*, 2022. 29

[379] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: learning a robotic hand imitator by watching humans on youtube. *arXiv preprint arXiv:2202.10448*, 2022. 43, 45

[380] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube, 2022. 85, 86, 88

[381] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*, 2019. 2, 9, 10, 17, 19, 21, 29, 58, 84

[382] K. Sohn, X. Yan, and H. Lee. Learning Structured Output Representation using Deep Conditional Generative Models. In *NeurIPS*, 2015. 46

[383] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *NeurIPS*, 2015. 16

[384] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020. 127

[385] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *Robotics and Automation Letters*, 2020. 1, 8, 11, 29, 70

[386] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*. John Wiley & Sons, 2020. 101

[387] Jordi Spranger, Roxana Buzatoiu, Athanasios Polydoros, Lazaros Nalpantidis, and Evangelos Boukas. Human-machine interface for remote training of robot tasks. *arXiv preprint arXiv:1809.09558*, 2018. 29

[388] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, 2002. 100

[389] F. Stulp, E. A. Theodorou, and S. Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *Transactions on Robotics*, 2012. 55, 99, 100, 117

[390] Ron Sun, Edward Merrill, and Todd Peterson. From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive science*, 2001. 106

[391] 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. 43

[392] Giovanni Sutanto, Zhe Su, Stefan Schaal, and Franziska Meier. Learning sensor feedback models from demonstrations via phase-modulated neural networks. In *ICRA*, 2018. 99

[393] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 1

[394] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 68

[395] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, pages 1057–1063, 2000. 1

[396] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999. 55, 99, 117

[397] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. 117

[398] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*, 2017. 34

[399] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *NeurIPS*, pages 4499–4509, 2017. 100, 106

[400] Sebastian Thrun, Christos Faloutsos, Tom Mitchell, and Larry Wasserman. Automated learning and discovery state-of-the-art and research topics in a rapidly growing field. *Ai Magazine*, 20(3):78–78, 1999. 117

[401] From One Hand to Multiple Hands: Imitation Learning for Dexterous Manipulation from Single-Camera Teleoperation. Qin, yuzhe and su, hao and wang, xiaolong, 2022. 43

[402] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *IROS*, 2018. 55

[403] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012. XII, 1, 8, 83, 109, 110

[404] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *Transactions on Robotics*, 2010. 99

[405] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *NeurIPS*, 33:19667–19679, 2020. 64

[406] Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, pages 6309–6318, 2017. 121

[407] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 32

[408] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *ICML*, 2017. 100

[409] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017. 29, 71

[410] Carl Vondrick, Deniz Oktay, Hamed Pirsiavash, and Antonio Torralba. Predicting motivations of actions by leveraging text. In *CVPR*, 2016. 29, 71

[411] Jianren Wang, Sudeep Dasari, Mohan Kumar Srirama, Shubham Tulsiani, and Abhinav Gupta. Manipulate by seeing: Creating manipulation controllers from pre-trained representations. 2023. 71

[412] 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. 10, 42, 71, 84

[413] Xiaolong Wang, Rohit Girdhar, and Abhinav Gupta. Binge watching: Scaling affordance learning from sitcoms. In *CVPR*, 2017. 41

[414] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 11, 17

[415] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022. 118, 119, 121

[416] Justin Wasserman, Karmesh Yadav, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Last-mile embodied visual navigation. In *CoRL*, 2022. 34

[417] Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017. 56

[418] William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware embeddings. *arXiv preprint arXiv:1908.09357*, 2019. 70, 99, 115, 117

[419] Bohan Wu, Suraj Nair, Li Fei-Fei, and Chelsea Finn. Example-driven model-based reinforcement learning for solving long-horizon visuomotor tasks. *arXiv preprint arXiv:2109.10312*, 2021. 54

[420] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning. *arXiv preprint arXiv:2206.14176*, 2022. 56

[421] Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. *arXiv preprint arXiv:2305.02968*, 2023. 120

[422] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. VIII, 16

[423] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv:2203.06173*, 2022. 27

[424] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 29, 54, 56, 78, 82, 86, 91, 93

[425] Haoyu Xiong, Quanzhou Li, Yun-Chun Chen, Homanga Bharadhwaj, Samarth Sinha, and Animesh Garg. Learning by watching: Physical imitation of manipulation skills from human videos. *arXiv preprint arXiv:2101.07241*, 2021. 11, 58

[426] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. *arXiv:2204.13226*, 2022. 27, 29

[427] Mengyuan Yan, Gen Li, Yilin Zhu, and Jeannette Bohg. Learning topological motion primitives for knot planning. In *IROS*, 2020. 29

[428] 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. 42

[429] Yezhou Yang, Yi Li, Cornelia Fermüller, and Yiannis Aloimonos. Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. In *AAAI*, 2015. 11

[430] Jianglong Ye, Jiashun Wang, Binghao Huang, Yuzhe Qin, and Xiaolong Wang. Learning continuous grasping function with a dexterous hand from human demonstrations. *arXiv preprint arXiv:2207.05053*, 2022. 29

[431] Yufei Ye, Abhinav Gupta, and Shubham Tulsiani. What's in your hands? 3d reconstruction of generic objects in hands. In *CVPR*, 2022. 29

[432] 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. 42

[433] Yufei Ye, Xueting Li, Abhinav Gupta, Shalini De Mello, Stan Birchfield, Jiaming Song, Shubham Tulsiani, and Sifei Liu. Affordance diffusion: Synthesizing hand-object interactions. In *CVPR*, 2023. 71

[434] Tsuneo Yoshikawa. Dynamic manipulability of robot manipulators. *Transactions of the Society of Instrument and Control Engineers*, 21(9):970–975, 1985. 43

[435] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv preprint arXiv:2008.04899*, 2020. 11, 29, 84

[436] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018. 9, 33

[437] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017. 43

[438] Ye Yuan and Kris Kitani. Ego-pose estimation and forecasting as real-time pd control. In *ICCV*, 2019. 99

[439] 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. 58

[440] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *CoRL*, 2022. 10

[441] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021. 131

[442] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *CoRL*, 2020. 29, 55, 119, 124

[443] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *arXiv:1803.09956*, 2018. 29, 34, 55

[444] Kuo-Hao Zeng, Luca Weihs, Ali Farhadi, and Roozbeh Mottaghi. Pushing it out of the way: Interactive visual navigation. In *CVPR*, 2021. 34

[445] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022. 72

[446] Edwin Zhang, Yujie Lu, William Wang, and Amy Zhang. Lad: Language augmented diffusion for reinforcement learning. *arXiv preprint arXiv:2210.15629*, 2022. 119

[447] Kevin Zhang, Mohit Sharma, Jacky Liang, and Oliver Kroemer. A modular robotic arm control stack for research: Franka-interface and frankapy. *arXiv preprint arXiv:2011.02398*, 2020. 109

[448] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, 2018. 29

[449] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *ICCV*, pages 2248–2255, 2013. 11, 17

[450] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. 123

[451] Yibiao Zhao and Song-Chun Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, 2013. 28, 71

[452] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using sift features and mean shift. *Computer vision and image understanding*, 2009. 73

[453] 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. 2, 44, 62

[454] 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. 36, 87

[455] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020. 18

[456] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 10, 19

[457] Xupeng Zhu, Dian Wang, Ondrej Biza, Guanang Su, Robin Walters, and Robert Platt. Sample efficient grasp learning using equivariant models. *arXiv preprint arXiv:2202.09468*, 2022. 131

[458] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *ICRA*, 2017. 34

[459] Yixin Zhu, Chenfanfu Jiang, Yibiao Zhao, Demetri Terzopoulos, and Song-Chun Zhu. Inferring forces and learning human utilities from videos. In *CVPR*, 2016. 28, 71

[460] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014. 28, 71

[461] 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. 42, 84