

Generalizable Dexterity with Reinforcement Learning

Wenxuan Zhou

CMU-RI-TR-23-78

October 23, 2023



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

Thesis Committee:

David Held, *Carnegie Mellon University, Chair*

Abhinav Gupta, *Carnegie Mellon University*

Oliver Kroemer, *Carnegie Mellon University*

Vincent Vanhoucke, *Google*

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

Copyright © 2023 Wenxuan Zhou. All rights reserved.

To my husband, Huadian.

Abstract

Dexterity, the ability to perform complex interactions with the physical world, is at the core of robotics. However, existing research in robot manipulation has been focused on tasks with limited dexterity, such as pick-and-place. The motor skills of the robots are often quasi-static, have a predefined or limited sequence of contact events, and involve restricted object motions. In contrast, humans interact with their surroundings with dynamic and contact-rich manipulation skills, allowing us to perform a wider variety of tasks in a broader range of settings.

This thesis explores using Reinforcement Learning (RL) to equip robots with generalizable dexterity. RL solves sequential decision-making problems modeled as Markov Decision Processes (MDPs). RL has shown remarkable success in many domains such as games, making it a promising technique for developing advanced manipulation skills. Our research advocates for the following thesis statement: *Reconsidering how we frame the robotics problem as an MDP is effective and essential to achieve generalizable dexterity through RL.* We examine three challenges when applying RL to manipulation and discuss our approaches to overcome them by reconsidering the MDP formulation.

First, robot data is time-consuming and expensive to collect. To reuse robot data effectively, we propose an offline RL algorithm by constructing a latent action space of the MDP. In addition, we discuss a framework that effectively reuses robot data across environments with non-stationary dynamics.

Second, robot dexterity is often assumed to be limited by the hardware design of the robot. We propose to enhance the robot’s dexterity beyond its hardware limitations by exploiting the external environment, showing dynamic and contact-rich emergent behaviors. We demonstrate that rethinking how we define the environment of the MDP is effective in improving robot dexterity with RL.

Third, learning dexterous skills that can generalize is challenging. We propose an RL framework with an action representation that is spatially-grounded and temporally-abstracted which allows the robot to learn complex interactions that can generalize to unseen objects. This further supports our claim that rethinking the action space of the MDP can lead to generalizable dexterity.

Acknowledgments

I would like to express my gratitude to those who supported and inspired me throughout this academic journey.

First and foremost, I would like to express my gratitude to my advisor, David Held. I appreciate Dave's dedication to providing us with all kinds of support. From finding research topics to implementation details, his guidance allows us to overcome many obstacles in research. More importantly, I appreciate Dave's constant patience and willingness to listen to my thoughts and feelings. I always feel that I have his support and understanding whenever I face difficult situations.

I am grateful to my committee members, Abhinav Gupta, Oliver Kroemer, and Vincent Vanhoucke. Abhinav introduced me to the field of robot learning and always challenged me to think deeply about my research. Conversations with Oliver have been thought-provoking. I formed the theme of this thesis thanks to the discussion with him. I also appreciate Vincent for being on my thesis committee. Discussions with Vincent broadened my horizons and made me think beyond the directions I have been pursuing.

I am very fortunate to have worked at Google DeepMind and FAIR during summer internships. The internships enriched my research experience and exposed me to many exciting research topics. Thank you to my mentors, Steven Bohez, Chris Paxton, and Keerthana Gopalakrishnan. Your support during the internship was extremely valuable to me. I'm also grateful to all the collaborators during my internships, including Jan Humplik, Abbas Abdolmaleki, Dushyant Rao, Markus Wulfmeier, Tuomas Haarnoja, Nicolas Heess, Chuyuan Fu, Dorsa Sadigh, Karol Hausman, Pannag Sanketi, Quan Vuong, Ted Xiao, Pierre Sermanet, Chelsea Finn, Ying Xu, Zhuo Xu, Michael Ryoo, and many more.

I thank my collaborators, lab mates, and friends at CMU. Working with Sujay Bajracharya, Harshit Sikchi, Yi Gu, Fan Yang, and Bowen Jiang was my pleasure. I also am grateful to meet everyone in R-PAD, including Thomas Weng, Ben Eisner, Yufei Wang, Jenny Wang, Xingyu Lin, Siddharth Ancha, Brian Okorn, Daniel Sieta, Chuer Pan, Zhanyi Sun, and many others. I will miss the lab meeting discussions and the office chats. I also want to say thank you to all my friends, including Yi Sha,

Gengshan Yang, Xianyi Cheng, Shuyan Zhou, Yufei Ye, Donglai Xiang, Anqi Yang, Shikhar Bahl, Helen Jiang, Homanga Bharadhwaj, Raunaq Bhirangi, and many other friends. You have made this journey enjoyable and memorable.

I also want to express my gratitude to my parents. They believe in my potential more than anyone. They taught me to try my best regardless of the outcomes. Lastly, I thank my husband, Huadian Liu, for his love and support throughout this journey and for bringing happiness and inspiration to my life.

Contents

1	Introduction	1
2	PLAS: Latent Action Space for Offline Reinforcement Learning	5
2.1	Introduction	5
2.2	Related Work	7
2.3	Background	8
2.3.1	Preliminaries	8
2.3.2	Offline RL: From Pessimistic MDP to Policy Constraints	8
2.3.3	Variational Auto-encoder	10
2.4	Method	10
2.4.1	Policy in Latent Action Space (PLAS)	11
2.4.2	Generalization out of the dataset	12
2.4.3	Implementation Details	13
2.5	Experiments	14
2.5.1	Experiment Descriptions	14
2.5.2	Performance on Real-Robot Experiment	15
2.5.3	Performance on D4RL datasets	16
2.5.4	Overestimation of Learned Q-functions	17
2.5.5	Effect of the Optional Perturbation Layer	18
2.6	Conclusion	19
3	Forgetting and Imbalance in Robot Lifelong Learning with Off-policy Data	21
3.1	Introduction	21
3.2	Related Work	24
3.3	Preliminaries	25
3.3.1	Problem Definition: Lifelong reinforcement learning with environment variations	25
3.3.2	Off-Policy Reinforcement Learning Algorithms	26
3.4	Forward and Backward trade-off in Lifelong Reinforcement Learning	28
3.5	Offline Distillation Pipeline	30

3.6	Imbalanced Experience in Offline Distillation	31
3.7	Experiments	32
3.7.1	Experiment Setup	32
3.7.2	Offline Distillation for Lifelong Reinforcement Learning	34
3.7.3	Imbalanced Experience in Offline Distillation	35
3.8	Conclusion	40
4	Learning to Grasp the Ungraspable with Emergent Extrinsic Dexter- ity	43
4.1	Introduction	43
4.2	Related Work	45
4.2.1	Extrinsic dexterity	45
4.2.2	Grasping	46
4.2.3	Reinforcement Learning for Manipulation	47
4.3	Task Definition: Occluded Grasping	47
4.4	Learning dexterous grasping with Reinforcement Learning	48
4.4.1	Preliminaries: Goal-conditioned Reinforcement Learning	48
4.4.2	RL Problem Design	49
4.4.3	Extrinsic Environment	50
4.4.4	Choice of Low-level Controller	50
4.4.5	Multi-grasp Training and Grasp Selection	51
4.4.6	Improving Policy Generalization	52
4.5	Experiments	52
4.5.1	Training Curves and Ablations	52
4.5.2	Emergent Behaviors	53
4.5.3	Multi-grasp Experiments	55
4.5.4	Policy Generalization	57
4.5.5	Real-robot experiments	57
4.6	Limitations	59
4.7	Conclusion	60
5	HACMan: Learning Hybrid Actor-Critic Maps for 6D Non-Prehensile Manipulation	61
5.1	Introduction	61
5.2	Related Work	63
5.3	Preliminaries	65
5.4	Problem Statement and Assumptions	65
5.5	Method	66
5.5.1	Action Representation	66
5.5.2	Hybrid RL Algorithm	67
5.5.3	Representing the Goal as Per-Point Goal Flow	69

5.6	Experiment Setup	70
5.7	Simulation Results	71
5.8	Real robot experiments	74
5.9	Limitations	75
5.10	Conclusion	76
5.11	Extensions to HACMan	76
	5.11.1 HACLeg: Visual Manipulation with Legs	76
	5.11.2 HACMan++: A Spatially-grounded Skill Library for Manipulation	78
6	Conclusions	83
6.1	Summary	83
6.2	Future Directions	84
A	Appendix to PLAS (Chapter 2)	87
A.1	Implementation Details	87
A.2	D4RL Results	88
A.3	Sensitivity Analysis: Max Latent Action	88
A.4	Ablation Study: Perturbation Layer	92
A.5	Empirical Analysis on MMD Constraint	93
A.6	Robot Experiment	94
B	Appendix to Robot Lifelong Learning (Chapter 3)	97
B.1	Additional results on the offline distillation pipeline	97
B.2	Algorithm	97
B.3	Additional results with three environments and parallel sharing	100
	B.3.1 Forgetting and the effectiveness of Offline Distillation	100
	B.3.2 Imbalance experience in offline distillation	102
C	Appendix to Grasp the Ungraspable (Chapter 4)	107
C.1	Additional Results	107
	C.1.1 Sensitivity analysis on physical parameters	107
	C.1.2 Sensitivity analysis on object pose estimation noise	107
	C.1.3 Reward term weights	109
C.2	Implementation Details	109
	C.2.1 Simulation environment	109
	C.2.2 Grasp configurations	110
	C.2.3 Success rate calculation	110
	C.2.4 Observation and action space	110
	C.2.5 Low-level controller	111
	C.2.6 Multi-Grasp Training with Curriculum	112

C.2.7	RL Training	112
C.3	Automatic Domain Randomization	112
C.4	Real robot experiment	115
C.4.1	Implementation details	115
C.4.2	More information on the objects	118
C.4.3	Failure cases	119
D	Appendix to HACMan (Chapter 5)	123
D.1	Simulation Environment	123
D.1.1	Object dataset preprocessing	123
D.1.2	Collecting goal poses	125
D.1.3	Representing the goal as per-point goal flow	125
D.1.4	Success rate definition	126
D.1.5	Observation	126
D.1.6	Action	127
D.2	Algorithm and Training Details	128
D.2.1	HACMan (Ours)	128
D.2.2	Baselines	129
D.3	Supplementary Experiment Results	131
D.3.1	Additional ablations	131
D.3.2	Training curves and tables	132
D.3.3	Additional baseline: Global feature with query contact location	135
D.3.4	Extending Motion Parameters	136
D.3.5	Experiments in cluttered environments	137
D.3.6	Effect of longer training time	137
D.3.7	Effect of longer episode lengths	138
D.3.8	Per-category result breakdown	138
D.3.9	Final Distance to Goal	140
D.4	Real Robot Experiments	140
D.4.1	Real robot setup	140
D.4.2	Analysis	142
D.4.3	Failure cases	142
D.5	Discussion on non-prehensile manipulation	143
D.6	More discussion on the related work	144
	Bibliography	147

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

List of Figures

2.1	Overview: Instead of explicitly matching the action distribution of the agent policy with the behavior policy using divergence metrics such as KL or MMD, we implicitly constrain the policy to output actions within the support of the behavior policy through the latent action space.	6
2.2	Network architecture for PLAS: Given a state, the latent policy outputs a latent action, which is then input into the decoder. The latent action space implicitly defines a constraint over the action output. An optional perturbation layer can be added on top of the output from the decoder to allow controlled generalization out of the training distribution. . .	11
2.3	Real-robot experiment: (a) Experiment setup for the cloth sliding task. The cloth is fixed at the top left corner. (b) An example of the tactile sensor readings when the robot grasps at the edge of the cloth. (c) Training curves for the cloth sliding task on our method and the baselines. It shows the episode reward over five evaluation episodes every 150 training steps.	15
2.4	Training performance for medium-expert and medium-replay datasets on locomotion tasks. Each curve is averaged over 3 seeds. Shaded area shows one standard deviation across seeds.	16
2.5	We perform an analysis of Q-function errors of different methods, using the following metrics: (a) Mean-squared error of the Q-values (b) The percentage of overestimated Q-values (c) Mean of the positive errors (magnitude of overestimation) (d) Mean of the negative errors (magnitude of underestimation)	17
3.1	We investigate the problem of robot lifelong learning with non-stationary dynamics. In this example, a robot experiences joint deformation during training.	21
3.2	Forgetting in MPO: The figure shows a performance drop in the original environment after a switch of environment at 200k steps.	28
3.3	CRR is not as efficient as MPO when training from scratch.	29

3.4	We propose the Offline Distillation Pipeline: the agent is trained over the environments sequentially during online interaction phase and runs offline distillation at the end of training before deployment. “RB” means replay buffer in the figure.	30
3.5	Experiment Setup: A bipedal robot walking task with various environment variations.	33
3.6	Evaluation curves of the lifelong learning experiment across two stages. The Offline Distillation Pipeline effectively breaks the trade-off between forward and backward transfer and achieves better performance than the baselines at the end.	34
3.7	The imbalance issue in Offline Distillation: Training CRR with the combined dataset results in much lower performance in Env-A compared to training on the individual dataset. However, the worse performance corresponds to a higher average Q-value which indicates overestimation.	37
3.8	To study the imbalance issue, the figure shows the final performance of different variants of the offline distillation step at 5e6 policy updates. The Baseline corresponds to the performance of Combined Dataset in Figure 3.7.	38
3.9	Sensitivity analysis of data imbalance: Higher β makes CRR more robust to different ratios of dataset imbalance. The legend indicates the dataset size ratio of Env-A:Env-B corresponding to different colors.	40
4.1	We present a system for the “Occluded Grasping” task with extrinsic dexterity. The goal of this task is to reach an occluded grasp configuration (indicated by a transparent gripper). The figure shows an example of the emergent behavior of the policy and successful sim2real transfer.	44
4.2	Notations: ${}^W E$ denotes the pose of the end-effector in the world frame W . ${}^O g$ denotes the target grasp in the object frame O . Six marker locations m_i in green on the target grasp are used to calculate the occlusion penalty.	47
4.3	Outline of policy execution: Given the observations, the policy outputs an end-effector delta movement (Section 4.4.2) to the low-level controller (Section 4.4.4).	49
4.4	Training curves and ablations: Left: ablations on the reward function and the wall. Right: ablations on the controller.	53
4.5	Visualizations of the policies in different scenarios.	54
4.6	Left: Grasp configurations. Right: MultiGrasp Training results with and without curriculum.	56
4.7	Evaluation on the generalization of the policies by sampling 100 environments.	56

4.8	We evaluate the policy on the real robot with various test objects. The policy trained in simulation on box-shape objects can generalize to the real robot and other shapes. With ADR, the policy achieves 45% better success rate.	58
5.1	We propose HACMan (H ybrid A ctor- C ritic Maps for M anipulation), which allows non-prehensile manipulation of unseen objects into arbitrary stable poses. With HACMan, the robot learns to push, tilt, and flip the object to reach the target pose, which is shown in the first column and in the top row with transparency. The policy allows for dynamic object motions with complex contact events in both simulation (top) and in the real world (bottom). The performance of the policy is best understood from the videos on the website: https://hacman-2023.github.io	62
5.2	Illustration of our action space.	66
5.3	An overview of the proposed method. The point cloud observation includes the location of the points and point features. The goal is represented as per-point flow of the object points. The actor takes the observation as input and outputs an Actor Map of per-point motion parameters. The Actor Map is concatenated with the per-point critic features to generate the Critic Map of per-point Q-values. Finally, we choose the best contact location according to the highest value in the Critic Map and find the corresponding motion parameters in the Actor Map.	68
5.4	Baselines and Ablations. Our approach outperforms the baselines and the ablations, with a larger margin for more challenging tasks on the right. Success rates for simple tasks - pushing a single object to an in-plane goal - are high for all methods, but only HACMan achieves high success rates for 6D alignment of diverse objects.	71
5.5	Qualitative results for the object pose alignment task. HACMan shows complex non-prehensile behaviors that move the object to the goal pose (shown as the transparent object).	72
5.6	Goal-conditioned Critic Maps. Blue: goal point cloud. Color map: observed object point cloud. Lighter colors indicate higher Critic Map scores. Red arrows: motion parameters at a selected location. The policy uses different contact locations based on object geometries and goals.	74
5.7	Real robot experiments. HACMan achieves a 50% success rate over unseen objects with different geometries and physical properties, with 6D goal poses.	75

5.8	HACLeg demonstrates non-prehensile manipulation using leg to align the 6D pose of the object (white) to a given target pose (red).	77
5.9	In the real robot experiments, HACLeg enables object interactions such as (a) pushing, (b) flipping, or (c) moving the object to a distant goal through repeatedly pushing and walking.	77
5.10	HACMan++ Primitives. We consider a set of primitives in HACMan++. Each primitive is defined using a grounded location and a vector of motion parameters.	79
5.11	Overview of HACMan++. In HACMan++, we extend HACMan to incorporate multiple manipulation primitives, such as picking, placing, and poking. We compute the actor and critic maps for each primitives separately. Each primitive is represented as a one-hot encoding as an additional input to the actor and the critic network. The policy output is the type of primitive, a contact location, and a set of motion parameters, selected according to the highest score from the critic maps.	79
5.12	An example of the modified object pose alignment task. The goal of this task is to place the object at a target 6D pose in a different bin. The policy trained with HACMan++ learns to chain a sequence of prehensile and non-prehensile skills to achieve this task.	81
5.13	HACMan++ policies for the Maniskill2 benchmark. We also evaluate HACMan++ on the Maniskill2 benchmark, demonstrating the generality of the method in a wider range of manipulation tasks.	81
A.1	Sensitivity analysis on the max latent action for the latent policy: X-axis is the max latent action value. Y-axis is the normalized score.	92
A.2	Ablation study on the perturbation layer: X-axis is the max perturbation. Y-axis is the normalized score.	93
A.3	Simulated MMD loss with $N(0,1)$ as the behavior policy.	94
A.4	Simulated MMD loss with $N(0,1)$ as the behavior policy.	95
B.1	Comparison of off-policy algorithms for training from scratch which corresponds to the beginning stage of a lifelong learning experiment. This is an extension of Figure 3.6 Stage-1 result.	98
B.2	Comparison of off-policy algorithms during Stage-2 of the lifelong learning experiment. This is an extension of Figure 3.6 Stage-2 results.	98
B.3	Lifelong learning experiments with three stages. The policy trained with MPO during online interaction experiences significant forgetting on previous environments. With Offline Distillation at the end, the policy can recover the performance effectively over all the previous environments.	101

B.4	Evaluation of Offline Distillation with different transformation functions with a three-stage experiment setup.	103
B.5	Evaluation of Offline Distillation with different transformation functions with a parallel training experiment setup with three environments (type-A).	104
B.6	Evaluation of Offline Distillation with different transformation functions with a parallel training experiment setup with three environmentsv (type-B).	105
C.1	We evaluate the generalization of policies by changing one parameter at a time. The dashed lines indicate the default values of these parameters in the fixed environment.	108
C.2	We evaluate the sensitivity of the ADR policies on object pose estimation noise.	108
C.3	Training curves with different reward weights. For each plot, we train the policies by changing one of the weight terms to three different values.	108
C.4	Outline of policy execution: Given the observation, the policy outputs an end-effector delta movement. If the desired pose is within the joint limit of the robot, it will be sent to the low-level controller which operates at a higher frequency.	112
C.5	Training curves for the ADR policies.	114
C.6	Robot setup. We use one Azure Kinect camera for object pose estimation.	116
C.7	Illustrations of pose estimation pipeline for the non-box objects. The top row shows the scanned object model. The middle row shows bounding box calculation and pose definition. The last row shows an example of ICP.	117
C.8	Examples of pose estimation with ICP during an episode. The blue points are the observed point cloud from the camera. The red points are the object template that matches to the observed point cloud using ICP.	117
D.1	Training objects. 32 objects used in training.	124
D.2	Evaluation objects (unseen instance). 7 objects used in unseen instance evaluations. These instances are from the same categories as the training objects.	124
D.3	Evaluation objects (unseen category). 5 objects used in unseen category evaluations. They come from 4 randomly chosen categories.	124
D.4	Camera locations in simulation.	126
D.5	Additional ablations. All of the components of our method are essential to achieve the best performance when the task becomes more difficult.	131

D.6	Baselines. It shows success rates on the train dataset over environment steps. The shaded area represents the standard deviation across three training seeds.	133
D.7	Ablations. It shows success rates on the train dataset over environment steps. The shaded area represents the standard deviation across three training seeds.	133
D.8	Comparison between our method and the additional baseline with query contact locations. The left figure shows the success rate of the simplest task variant - a single object with planar goals. The right figure shows the most challenging task variant - all objects with 6D goals. The shaded area represents the standard deviation across three training seeds. Our method performs better than the baseline in both cases.	135
D.9	Qualitative results for object pose alignment tasks in cluttered environments. HACMan shows complex non-prehensile behaviors that move objects to goal poses (shown as the transparent objects). The scene objects are colored in brown to distinguish from the target object to be manipulated to the goal pose.	137
D.10	Success rate with extended training. The success rate of our method reaches $91.1 \pm 7.3\%$ after 500k training steps, compared to 83.3% after 200k training steps.	138
D.11	Success rates at various maximum episode lengths. This line plot shows the success rates of HACMan evaluated on the four datasets. It is worth noting that the success rates for Unseen Instance (Common) and Train (Common) are marginally higher compared to Train and Unseen Category, similar to the pattern in Table 5.2.	139
D.12	Results breakdown. Object categories in the unseen instance set (orange) can be compared to the same object categories in the train set (blue) to see the level of instance generalization.	139
D.13	Distribution of distances to the goal at the end of the episode for our method in the “All Objects 6D Goals” experiment. The vertical dashed line represents the success threshold at a distance of 0.03m. The distribution has a median of 2.57cm, a mean of 3.66cm, and a standard deviation of 4.27cm.	140
D.14	Real robot setup.	141
D.15	Examples showcasing limitations of prehensile manipulation. The frames where prehensile manipulation is challenging are highlighted. The first row shows a cube placed at the corner of the bin, where any grasp is obstructed by the bin wall. Both the second and third rows depict instances where objects are too large to be grasped at specific poses.	144

D.16 **An example of non-quasi-static motion.** The figure shows an example of executing the motion parameters to flip a mug upright. After the gripper pushes against the edge of the mug (first two images), it relies on the inertia of the mug to finish the motion which is not quasi-static (last two images). 145

List of Tables

2.1	Comparison of different perturbation values on random and medium datasets. Scores are normalized. $\epsilon = 0$ is the performance of the latent policy without the additional perturbation layer.	18
4.1	Comparison of grasp selection methods: Side grasp policies achieve better performance when using the Q-function to select the grasp.	56
5.1	Features of the proposed action representation compared to the baselines.	71
5.2	Generalization to unseen objects.	73
A.1	D4rl Benchmark Results: Average Reward	89
A.2	D4rl Benchmark Results: Normalized Score	89
A.3	D4rl Results on More Datasets: Average Reward. For these datasets, we searched over 0.5, 1, 2 for max latent action and report the best results.	90
A.4	D4rl Results on More Datasets: Normalized Score	91
B.1	Combinations of different environment variations in the three environment experiments.	100
C.1	Hyperparameters for RL training.	113
C.2	Simulation parameters in Automatic Domain Randomization	115
C.3	Real robot evaluations with more object information. We highlight the out-of-distribution aspect of the object properties in bold.	118
C.4	Failure cases for Policy w/ ADR during real robot evaluation. The most common failures include dropping the object during rotation, repeated rotation, and unexpected object dynamics.	120
C.5	Failure cases for Policy w/o ADR during real robot evaluation. The most common failures include missing the initial contact, repeated rotation and unexpected object dynamics.	121
D.1	Hyperparameters.	129

D.2	Baseline-specific Hyperparameters.	130
D.3	Baselines. We compare our method with baselines with different action representations and observations. Our approach outperforms the baselines, with a larger margin for more challenging tasks. The success rate is reported with the mean and standard deviation across three seeds.	134
D.4	Ablations. We show that all of the components are essential to achieve the best performance when the task becomes more difficult. Each success rate is reported with the mean and standard deviation across three seeds.	134
D.5	Success rates with different motion parameters. All methods are evaluated on all train objects with 6D goals.	136
D.6	Success rates under different cluttered scenes. All methods are evaluated with 6D goals.	137
D.7	Additional analysis on the real robot experiments. An episode is considered a “flow success” if the average norm of the estimated flow is less than 3 cm. An episode is considered as an “actual success” if the object is aligned with the goal pose without point cloud registration failure.	143

Chapter 1

Introduction

Despite significant progress in robotics over recent decades, robots are still mostly limited to highly constrained interactions with the physical world due to their limited dexterity. For example, an extensively studied category of tasks in manipulation is pick-and-place, where the robot interacts with the object only during the gripper’s opening and closing [54, 74, 96, 145]. Other examples include planar pushing and articulated object manipulation, such as opening or closing a drawer, in which the objects have limited possible movement [22, 26, 81, 136]. The interactions in these tasks are typically quasi-static, involve very few contact events, and involve limited changes of the physical world.

However, performing more advanced tasks in the real world often requires robots to have more dexterous motor skills that are dynamic and contact rich. For instance, to pick up a book lying on a table, the robot may need to rotate the book before grasping it from the side [149]. Similarly, organizing a messy cabinet would require the robot to interact with the objects in the cabinet using more sophisticated skills than pick-and-place or planar pushing [68]. As a result, improving the dexterity of robots is a crucial step in building more intelligent and capable robots that can perform a wider range of tasks in the real world.

In this thesis, we aim to enhance the dexterity of robots with Reinforcement Learning (RL) [121]. In RL, decision-making is framed as a Markov Decision Process (MDP): an agent, such as a robot, interacts with the environment and receives rewards for its actions. The agent then learns to improve its decisions based on the rewards

to eventually master the given task. Since RL has shown tremendous progress in decision-making problems such as games and animated characters [69, 78, 114], it holds great promise for developing robot skills. However, applying RL to robotics presents additional challenges compared to games. In this thesis, we discuss and address three challenges when leveraging the power of RL for robot dexterity. We advocate for the following statement: *Reconsidering how we frame the robotics problem as an MDP is effective and essential to achieve generalizable dexterity through RL.*

Limited real world data. As the robot’s required motor skills become more complex, real robot data becomes increasingly crucial. More dynamic motor skills often depend on the details of the physics that are difficult to capture accurately in simulation. However, collecting real robot data is often a time-consuming and expensive process, resulting in limited data availability. Therefore, effectively reusing data becomes increasingly important.

In Chapter 2, we address the challenges of limited data by proposing an offline reinforcement learning algorithm that operates over an MDP with a latent action space [151]. The goal of offline reinforcement learning is to learn a policy from a fixed dataset, without further interactions with the environment [67]. Existing off-policy algorithms have limited performance on static datasets due to extrapolation errors from out-of-distribution actions [31]. This leads to the challenge of constraining the policy to select actions within the support of the dataset during training. We propose to learn the **P**olicy in the **L**atent **A**ction **S**pace (PLAS) such that this requirement is naturally satisfied. We evaluate our method on continuous control benchmarks in simulation and a deformable object manipulation task with a physical robot. We demonstrate that our method provides competitive performance consistently across various continuous control tasks and different types of datasets, outperforming existing offline reinforcement learning methods with explicit constraints.

In Chapter 3, we study the challenges of reusing robot data with non-stationary dynamics in RL [152]. Robots will experience non-stationary environment dynamics throughout their lifetime: the robot dynamics can change due to wear and tear, or its surroundings may change over time. Eventually, the robots should perform well in all of the environment variations it has encountered. At the same time, it should still be able to quickly adapt to a new environment. We identify two challenges of RL algorithms under such a lifelong learning setting with off-policy data: first,

existing off-policy algorithms struggle with the trade-off between being conservative to maintain good performance in the old environment and learning efficiently in the new environment, even if we keep all the data in the replay buffer. We propose the “Offline Distillation Pipeline” to break this trade-off by separating the training procedure into an online interaction phase and an offline distillation phase. Second, we find that training with the imbalanced off-policy data from multiple environments across the robot’s lifetime creates a significant performance drop. We identify that this performance drop is caused by the combination of the imbalanced *quality* and *size* among the datasets which exacerbate the extrapolation error of the Q-function. During the distillation phase, we apply a simple fix to the issue by constraining the policy closer to the behavior policy that generated the data. In the experiments, we demonstrate these two challenges and the proposed solutions with a simulated bipedal robot walking task across various environment changes.

Limited robot hardware. A robot’s dexterity are often assumed to be limited by its hardware design. Complex motion skills are often associated with complex robot hardware such as dexterous hands with high degree-of-freedom fingers [13, 87, 99]. In contrast, we take an alternative perspective to improve the robot’s dexterity beyond its limited hardware by redefining the environment of the MDP.

In Chapter 4, we investigate such an idea known as “Extrinsic Dexterity” [18, 149]. The key idea is that a *simple* gripper can solve more *complex* manipulation tasks if it can utilize the external environment such as pushing the object against the table or a vertical wall. Previous work in extrinsic dexterity usually has careful assumptions about contacts which impose restrictions on robot design, robot motions, and the variations of the physical parameters [16, 18, 43, 45]. In this work, we develop a system based on RL to address these limitations. We study the task of “Occluded Grasping” which aims to grasp the object in configurations that are initially occluded; the robot needs to move the object into a configuration from which these grasps can be achieved. We present a system with model-free RL that successfully achieves this task using a simple gripper with extrinsic dexterity. The policy learns emergent behaviors of pushing the object against the wall to rotate and then grasp it without additional reward terms on extrinsic dexterity. We discuss important components of the system including the design of the RL problem, multi-grasp training and selection, and policy generalization with automatic curriculum. Most importantly, the policy trained in

simulation is zero-shot transferred to a physical robot. It demonstrates dynamic and contact-rich motions with a simple gripper that generalizes across objects with various size, density, surface friction, and shape with a 78% success rate.

Limited generalization. Robots to be deployed in the real world need to face diverse scenarios. For example, in manipulation, robots might encounter unseen objects in diverse configurations. Thus, robots not only need to solve the task, but also need to generalize. However, learning complex motor skills that can generalize imposes challenges to the RL algorithms. In this thesis, we explore the effectiveness in rethinking the action space of the MDP to enable complex and generalizable dexterity.

In Chapter 5, we introduce **Hybrid Actor-Critic Map for Manipulation (HACMan)**, a reinforcement learning approach for 6D non-prehensile manipulation of objects using point cloud observations. HACMan proposes a *temporally-abstracted* and *spatially-grounded* object-centric action representation that consists of selecting a contact location from the object point cloud and a set of motion parameters describing how the robot will move after making contact. We modify an existing off-policy RL algorithm to learn in this hybrid discrete-continuous action representation. We evaluate HACMan on a 6D object pose alignment task in both simulation and in the real world. On the hardest version of our task, with randomized initial poses, randomized 6D goals, and diverse object categories, our policy demonstrates strong generalization to unseen object categories without a performance drop, achieving an 89% success rate on unseen objects in simulation and 50% success rate with zero-shot transfer in the real world. Compared to alternative action representations, HACMan achieves a success rate more than three times higher than the best baseline. With zero-shot sim2real transfer, our policy can successfully manipulate unseen objects in the real world for challenging non-planar goals, using dynamic and contact-rich non-prehensile skills.

Furthermore, we demonstrate the generality of HACMan when applied to other robot platforms and manipulation tasks. We apply HACMan to a quadruped robot and enable non-prehensile manipulation of objects using legs, breaking the boundaries of manipulation and locomotion. In addition, we extend HACman into a hierarchical RL framework by incorporating a library of spatially-grounded primitives. The extended framework, HACMan++, demonstrates complex synergies between prehensile and non-prehensile skills in a diverse set of manipulation tasks.

Chapter 2

PLAS: Latent Action Space for Offline Reinforcement Learning

2.1 Introduction

Reinforcement learning (RL) has achieved much success on many robotics tasks in simulation [80, 93]. However, it still has limited applications in the real world including real robots. One major challenge of applying RL in the real world is that it requires a large number of online interactions with the environment, usually more than millions of time steps. Offline Reinforcement Learning, or Batch Reinforcement Learning, aims to develop algorithms that can optimize the policy given a static dataset of transitions without any active data collection [65, 67]. This is especially important for robotics because algorithms that train from static datasets can provide additional flexibility in terms of data collection. We may take into account safety, use better exploration methods [92], and leverage demonstrations [100]. In addition, we can accumulate past experience during the development of the algorithm by re-using the replay buffer or evaluation trajectories from previous RL experiments. Furthermore, static datasets can be shared within the community, and thus, they are more likely to be scaled up in size.

In contrast to offline RL, off-policy RL uses a replay buffer that stores transitions that are actively collected by the policy throughout a training procedure. Past

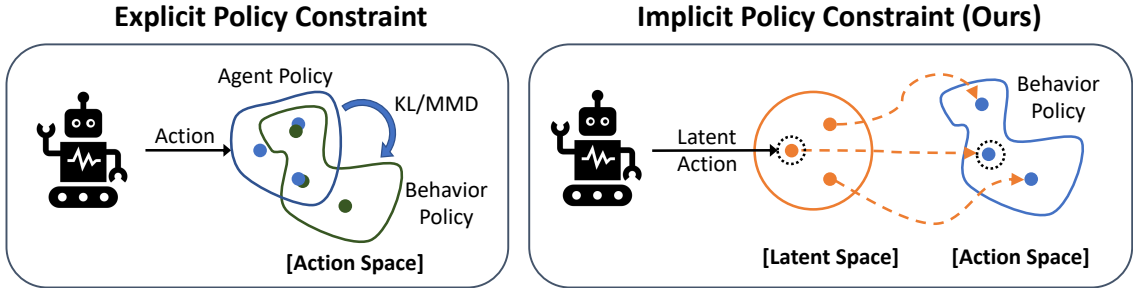


Figure 2.1: Overview: Instead of explicitly matching the action distribution of the agent policy with the behavior policy using divergence metrics such as KL or MMD, we implicitly constrain the policy to output actions within the support of the behavior policy through the latent action space.

work has shown that off-policy RL methods cannot be directly applied to static datasets due to the extrapolation error of the Q-function caused by out-of-distribution actions [31]. To avoid extrapolation error, we need to constrain the policy to select actions *within* the support of the dataset. On the other hand, the constraint cannot be “overly restrictive”; in the extreme case, an overly constrained policy will degenerate to behavior cloning on the dataset. The design of such a constraint remains a challenging problem.

We propose a simple yet effective method that trains the Policy in the Latent Action Space (PLAS) to implicitly constrain the policy to output actions within the support of the dataset instead of using explicit constraints, as illustrated in Figure 2.1. Following previous work, we model the “behavior policy” of the dataset as a Conditional Variational Autoencoder (CVAE). Our insight is that we can learn a policy in the latent action space of the CVAE and then use its decoder to output an action in the original action space of the environment. The latent action space *implicitly* constrains the policy by construction. The benefit of such a constraint is that it can be naturally satisfied without affecting the optimization of the other components and without being restricted by the density of the behavior policy distribution.

We demonstrate that PLAS allows generalization within the dataset and can provide consistently good performance for datasets with diverse actions. In cases where the Q-function generalizes well without significant extrapolation error, we augment our approach by allowing out-of-distribution actions in a controlled way to

achieve better performance. This explicit separation of in-distribution generalization and out-of-distribution generalization allows the user fine-grained control over the generalization of the method. We evaluate our method on the continuous control tasks from the d4rl benchmark datasets [29] as well as real-robot experiments on deformable object manipulation and show superior performance to previous methods, despite the simplicity of our approach.

2.2 Related Work

Offline Reinforcement Learning: Offline reinforcement learning studies the problem of learning policies from static datasets without any active data collection [65, 67]. Recent work proposes different approaches in this direction [4, 57, 94, 95, 141]. It has been empirically shown that the performance of off-policy algorithms drastically degrades when directly applied to static datasets due to out-of-distribution actions [31]. Several papers propose to avoid out-of-distribution actions by enforcing constraints on the policy such as using a KL-divergence constraint or maximum mean discrepancy (MMD) constraint [49, 62, 132]. In the most similar approach to our work, Fujimoto, *et al.* [31] (BCQ) propose to learn a generative model for the behavior policy and perturb the randomly generated samples to find good perturbed actions that maximize the Q-function. Our experiments show that these approaches have worse performance than our method, likely due to the difficulty of satisfying the constraints or balancing in-distribution vs out-of-distribution generalization.

Imitation Learning: The most naive way of using a static dataset is to perform behavior cloning. This approach is usually used when the dataset is generated by an expert policy. Behavior cloning only mimics the actions in the dataset and does not reason about which actions in the dataset are better than others. Imitation learning methods sometimes also assume access to an expert policy [103] and may allow interactive data in the environment [42], which is very different from offline RL.

Generative Models for Actions: Previous work has used a conditional variational autoencoder to model actions, although not in the offline RL setting. Mishra et al. [77] samples action sequences from the CVAE when they perform trajectory optimization

with the learned latent dynamics model. Krupnik et al. [61] extended the previous method to multi-agent RL by learning a disentangled latent action representation. In contrast to these works, we focus on demonstrating the capability of using a CVAE over actions to deal with the out-of-distribution issue in off-policy RL in the offline setting.

2.3 Background

2.3.1 Preliminaries

As is common in reinforcement learning, we define the environment as a Markov Decision Process (MDP) represented as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where \mathcal{S} is the state space, \mathcal{A} is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition probability function, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, and γ is the discount factor. The general objective of RL is to find a policy that maximizes the expectation of the return $G_t = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}, s_{t+k+1})$.

Given a policy π , the action-value function, or Q-function, is defined as $Q^\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$. Our method builds on top of the commonly used off-policy actor-critic procedure with a deterministic policy [30, 69]. The Q-function of the deterministic policy π is estimated based on the Bellman Operator:

$$\mathcal{T}\hat{Q}^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_t + \gamma\hat{Q}^\pi(s_{t+1}, \pi(s_{t+1}))] \quad (2.1)$$

The policy π_θ is updated following the Deterministic Policy Gradient [113]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \rho_\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s)}] \quad (2.2)$$

2.3.2 Offline RL: From Pessimistic MDP to Policy Constraints

In this section, we will discuss the objectives for offline reinforcement learning and the limitations of existing methods that build on top of off-policy RL.

In offline RL, we are given a fixed dataset $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})_i\}$ with a finite number of transitions. The difficulty comes from the fact that the static dataset

does not cover the entire state space and action space of the MDP. This is especially true when the state and action spaces are continuous. The objective of offline RL is typically to find the policy that maximizes the cumulative reward during its deployment in the environment. However, the performance of the policy will be limited by our knowledge over the MDP, which is inferred from a limited set of transitions.

Reconsidering this problem, another reasonable objective for offline RL is to maximize the cumulative reward of the MDP under the transitions that have been visited in the dataset. Following [57], we may assume a pessimistic MDP such that $r(s, a)$ is significantly small for any unvisited (s, a) . Optimizing under such a pessimistic MDP is an intuitive surrogate objective. In addition, [57] proves that the performance of any policy for such a pessimistic MDP is a lower bound in the true MDP.

An additional reason to optimize for this pessimistic MDP is the extrapolation error of approximated Q-functions [31]. In off-policy algorithms, we bootstrap $Q(s_t, a_t)$ by using $Q(s_{t+1}, \pi(s_{t+1}))$ according to the Bellman operator as in Equation (2.1). If $(s_{t+1}, \pi(s_{t+1}))$ is not in the dataset, $Q(s_{t+1}, \pi(s_{t+1}))$ can be arbitrarily wrong. This error caused by out-of-distribution *actions* will be accumulated and exacerbated by the policy update. (Note that out-of-distribution *states* do not occur during training.) Thus, optimizing the policy under the pessimistic MDP is equivalent to forcing the policy to select known actions that avoid any error accumulation. This is the motivation for constraining the policy to be within the support of the dataset. On the other hand, the constraint should not be overly restrictive and should not be affected by the distribution of the dataset as proposed in [62]. The policy should have the full flexibility to choose actions within the support.

Existing offline RL methods enforce such a constraint in different ways. BCQ [31] constrains the policy by sampling from the behavior policy. However, the policy is then restricted by the distribution of the behavior policy. BEAR [62] and BRAC [132] incorporate the constraint on the policy as a regularization term into the optimization process for the policy or the Q-function. This regularization term is calculated by a divergence metric, such as KL-divergence or MMD. There are two practical challenges to these approaches. First, this additional loss term creates a trade-off between optimizing the original objective and satisfying the constraint. Although BEAR uses

a Lagrangian multiplier to solve a constrained optimization problem, in practice, the constraint is almost never satisfied [132]. Second, the choice of the divergence metric, such as KL and MMD, might be overly restrictive given datasets that have diverse actions. We provide further discussion on MMD constraint in Appendix A.5. These past approaches have shown that properly enforcing an explicit policy constraint is difficult; this observation motivates our approach.

2.3.3 Variational Auto-encoder

Since our method uses a conditional variational autoencoder, we include a brief background of VAE in this section in its most general form based on [59] and [20]. Given a dataset $X = \{x^{(i)}\}_{i=1}^N$, the goal of a VAE is to generate samples that are from the same distribution as the data points, in other words, to maximize $p(x)$ for all $x^{(i)}$. This is achieved by introducing a latent variable z sampled from a prior distribution $p(z)$ and modeling a decoder $p_\theta(x|z)$ with parameter θ . Directly maximizing the marginal likelihood $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$ is intractable. Instead, Kingma and Welling [59] propose to approximate the true posterior $p_\theta(z|x)$ by training an encoder $q_\phi(z|x)$. In this way, they derive the following evidence lower bound (ELBO) on the log-likelihood of the data:

$$\max_{\theta} \log p(x) \geq \max_{\theta, \phi} \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z)] - \mathcal{D}_{KL}[q_\phi(z|x)||p_\theta(z)] \quad (2.3)$$

The term $\log p_\theta(x|z)$ (where z is sampled from $q_\phi(z|x)$) represents the reconstruction loss. The second term is the KL-divergence between the encoder output $q_\phi(z|x)$ and the prior of z , $p_\theta(z)$, usually set to be $\mathcal{N}(0, 1)$. Thus, optimizing for this objective enables us to train a model that generates samples similar to the data distribution by sampling z and then passing it into the decoder.

2.4 Method

In this section, we introduce our method PLAS (Policy with Latent Action Space) that implicitly constrains the policy to be within the support of the behavior policy. Our method disentangles the in-distribution and out-of-distribution generalization

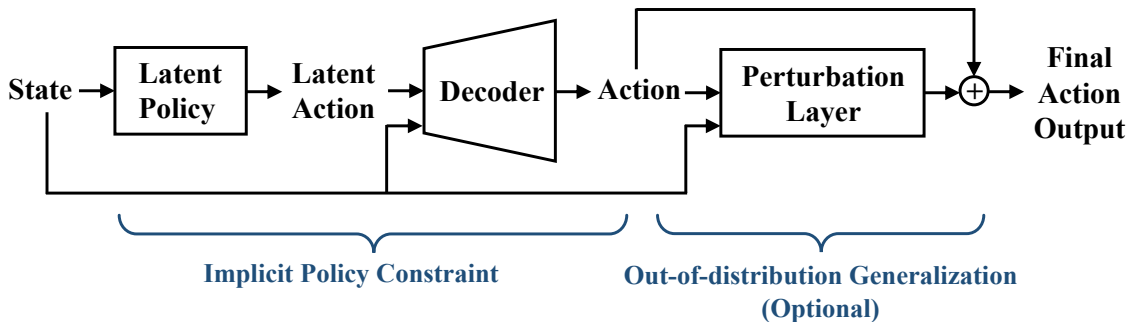


Figure 2.2: Network architecture for PLAS: Given a state, the latent policy outputs a latent action, which is then input into the decoder. The latent action space implicitly defines a constraint over the action output. An optional perturbation layer can be added on top of the output from the decoder to allow controlled generalization out of the training distribution.

of actions, enabling fine-grained control over the generalization of the method. The network architecture of our method is shown in Figure 2.2.

2.4.1 Policy in Latent Action Space (PLAS)

Given a static dataset, we use a conditional variational autoencoder (CVAE) to model the behavior policy $p(a|s)$, as in other recent methods [31, 62, 132]. The CVAE is trained to reconstruct actions conditioned on the states. Converting Equation 2.3 into our problem formulation, the objective of the CVAE is to maximize $\log p(a|s)$ by maximizing its lower bound:

$$\max_{\alpha, \beta} \log p(a|s) \geq \max_{\alpha, \beta} \mathbb{E}_{z \sim q_{\alpha}} [\log p_{\beta}(a|s, z)] - \mathcal{D}_{KL}[q_{\alpha}(z|a, s) || P(z|s)] \quad (2.4)$$

where z is the latent variable, α and β are the parameters of the encoder and the decoder, respectively. This is similar to Equation 2.3, except that all terms are conditioned on the state s . A trained decoder $p_{\beta}(a|s, z)$ provides a mapping from the latent space to the action space, conditioned on the state.

In order to constrain the policy to be within the support of the dataset, we propose to train a deterministic policy $z = \pi(s)$ to map from a state s to a “latent action” z ; we then use the pretrained decoder $p_{\beta}(a|s, z)$ to project the latent action into the actual action space as shown in Figure 2.2. This is significantly different from

BCQ which samples from a fixed range of the latent space. The latent policy has the flexibility of choosing the latent actions and thus avoids being affected by the density of the dataset distribution. The CVAE is trained to maximize $\log p(a|s)$, equivalent to maximizing the expected output of the decoder $\mathbb{E}_{p(z|s)}[p_\beta(a|s, z)]$ as shown:

$$p(a|s) = \int_z p_\beta(a|s, z)p(z|s)dz = \mathbb{E}_{p(z|s)}[p_\beta(a|s, z)] \quad (2.5)$$

Thus, for values of z that have a high probability under the prior $p(z|s)$, the decoder $p_\beta(a|s, z)$ will output a high probability action under the behavior policy distribution $p(a|s)$ in expectation.

If we constrain the latent policy $z = \pi(s)$ to output a latent action z that has a high probability under the prior $p(z|s)$, then the full policy, formed by $p_\beta(a|s, z = \pi(s))$, is likely to have a high probability under the behavior policy $p(a|s)$. Fortunately, this constraint is simple to enforce; since the prior $p(z|s)$ is set to a normal distribution $\mathcal{N}(0, 1)$, we simply define $z = \pi(s)$ such that $z_i \in [-\sigma, \sigma]$ for each latent dimension i for some hyperparameter σ (see Section 2.4.3 for details). Furthermore, for each state s , the latent policy has the flexibility to choose any latent action z in this constrained latent action space, leading to an easier optimization compared to previous work with explicit constraints.

2.4.2 Generalization out of the dataset

The latent policy provides a natural constraint to stay within the support of the dataset. However, in some cases when the Q-function can generalize well, we may relax the pessimistic objective and allow the policy to select out-of-distribution actions to improve its performance. The benefit of the out-of-distribution actions is more likely to happen when the environment (both transition probabilities and the reward function) is smooth and when the dataset has limited quality and diversity. To do so, we add a perturbation layer to the output of the decoder that outputs a residual over the action. The residual is limited to a specific range $[-\epsilon, \epsilon]$, where ϵ is a hyperparameter. Mathematically, this enforces the final output action to be close to the actions within the dataset in terms of the L_∞ norm. This perturbation layer is inspired by BCQ. However, BCQ forms a policy by sampling from the generative model; the perturbation layer is used to prevent “sampling from the generative model

for a prohibitive number of times” [31]. In our case, we don’t perform any sampling since our policy is deterministic; the perturbation layer is instead specifically designed for out-of-distribution generalization. In the experiments, we will demonstrate that when the dataset has enough coverage in the state-action space, this additional layer is not necessary.

Algorithm 1: Off-policy RL with PLAS

Input: Dataset $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})_i\}$
// VAE Training
 Initialize encoder E_α and decoder D_β with parameters α and β .
for $i \leftarrow 1$ **to** M **do**
 | Sample a minibatch of k state-action pairs (s_t, a_t) from \mathcal{D} .
 | Optimize α and β using Equation 2.4.
end
// Policy Training
 Initialize the latent policy network π_θ , critic networks Q_{ϕ_1}, Q_{ϕ_2} and their corresponding target networks $\pi_{\theta'}, Q_{\phi'_1}$ and $Q_{\phi'_2}$ with $\theta' \leftarrow \theta, \phi'_1 \leftarrow \phi_1, \phi'_2 \leftarrow \phi_2$.
for $i \leftarrow 1$ **to** N **do**
 | Sample a minibatch of k transitions $\{(s_t, a_t, r_t, s_{t+1})_{i=1,\dots,k}\}$ from \mathcal{D} .
 | For each transition, generate a latent action using the latent policy:
 | $\psi_{t+1} = \pi_\theta(s_{t+1})$.
 | Decode latent actions using the decoder $a_{t+1} = D_\beta(\psi_{t+1})$.
 | Set $y = \lambda \min_{i=1,2} Q'_{\phi_i}(s_{t+1}, a_{t+1}) + (1 - \lambda) \max_{i=1,2} Q'_{\phi_i}(s_{t+1}, a_{t+1})$.
 | Update critic by minimizing: $L = (Q_{\phi_i}(s, a) - (r + \gamma y))^2$ for $i = 1, 2$.
 | Update actor according to Equation 2.2.
 | Update target networks: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta', \phi'_i \leftarrow \tau\phi_i + (1 - \tau)\phi'_i$ for $i = 1, 2$.
end

2.4.3 Implementation Details

The full algorithm is summarized in Algorithm 1. Our algorithm can be built on top of off-policy algorithm such as DDPG [69] or TD3 [30]. We use a deterministic policy $z = \pi(s)$ to output a latent action. The policy uses a tanh activation at the output layer to limit the max latent action. This limit is set to 2 by default, which corresponds to 2σ for the latent variable $p(z) = \mathcal{N}(0, 1)$. We use a soft Clipped

Double Q-learning with parameter λ to weight the two Q-functions. Following common practice, we use target networks to stabilize training with hyperparameter τ . The code for our algorithm is based on the BCQ repository, and we mostly follow the hyperparameters of BCQ. Further implementation details and hyperparameters can be found in Appendix A.1.

2.5 Experiments

We evaluate our algorithm on a wide range of continuous control tasks, including a physical robot experiment on deformable object manipulation and the d4rl benchmarks [29] including OpenAI Gym locomotion tasks, Adroit, Franka Kitchen, etc. For the d4rl benchmarks, our analysis in the main text is focused on the locomotion datasets; the full results including the other environments can be found in Appendix A.2. We compare our method with the following baselines: BCQ [31], BEAR [62], and BRAC [132]. We use the author’s implementation of these algorithms with recommended hyperparameters reported in these papers. Note that we use the latent policy without the perturbation layer by default because our primary focus is on the policy constraint and in-distribution generalization. The experiments that use the perturbation layer are explicitly mentioned.

2.5.1 Experiment Descriptions

Real-Robot Experiment: The task for the real-robot experiment is to slide along the edge of the cloth as far as possible with a tactile sensor. The experiment setup is shown in Figure 2.3(a) and an example of the tactile sensor reading is shown in Figure 2.3(b). More details of this experiment can be found in Appendix A.6. The dataset consists of the replay buffer from a previous online RL experiment with around 7000 timesteps of transitions and 5 episodes (around 300 timesteps) of expert demonstrations from a trained policy. We train the policy for 2400 steps in total for each experiment with evaluations every 150 steps over 5 episodes.

Locomotion Datasets: We mainly focus on the locomotion environments from the d4rl datasets in this section including Walker2d-v2, Hopper-v2, and Halfcheetah-v2. For each environment, there are four types of datasets: random, medium, medium

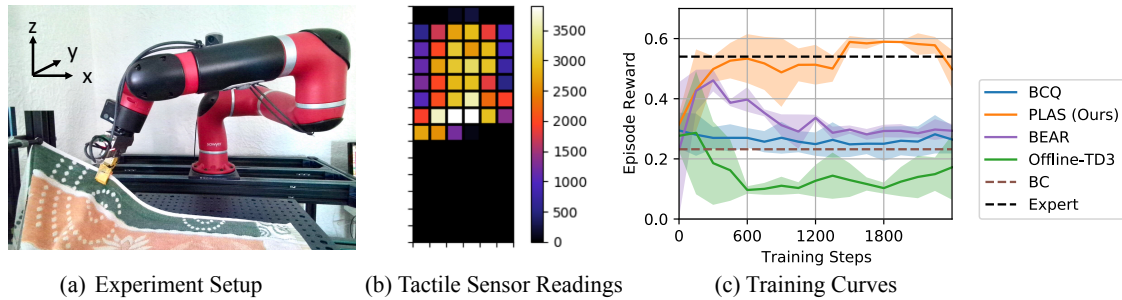


Figure 2.3: Real-robot experiment: (a) Experiment setup for the cloth sliding task. The cloth is fixed at the top left corner. (b) An example of the tactile sensor readings when the robot grasps at the edge of the cloth. (c) Training curves for the cloth sliding task on our method and the baselines. It shows the episode reward over five evaluation episodes every 150 training steps.

expert, and medium replay datasets. Random datasets are generated by randomly initialized policies. Medium datasets are generated from rollouts of a “medium” performance policy trained with Soft Actor-Critic up to a certain performance. Medium-expert datasets are generated by combining the medium datasets and expert datasets. Medium-replay datasets are the replay buffers created during the training of the medium policies. Note that medium-replay datasets are much smaller than the other types of datasets, making it more challenging to obtain stable training performance. We train the policy for 500 epochs and each epoch has 1000 training steps. The policy is evaluated every 1 epoch over 10 episodes.

2.5.2 Performance on Real-Robot Experiment

Figure 2.3(c) shows the evaluation performance of different methods across the training process for the cloth sliding task. The brown dashed line indicates the behavior cloning (BC) policy. It fails as expected because the average quality of the dataset is poor. The “Offline-TD3” baseline is to directly run TD3 over the offline dataset without any extra online data collection. The performance is even worse than BC, which shows the necessity of designing offline RL algorithms that can utilize fixed datasets for real-world applications. BCQ also doesn’t perform well, possibly because it is overly constrained by the dataset distribution, resulting in similar performance as behavior cloning. BEAR achieves reasonable performance at the beginning of

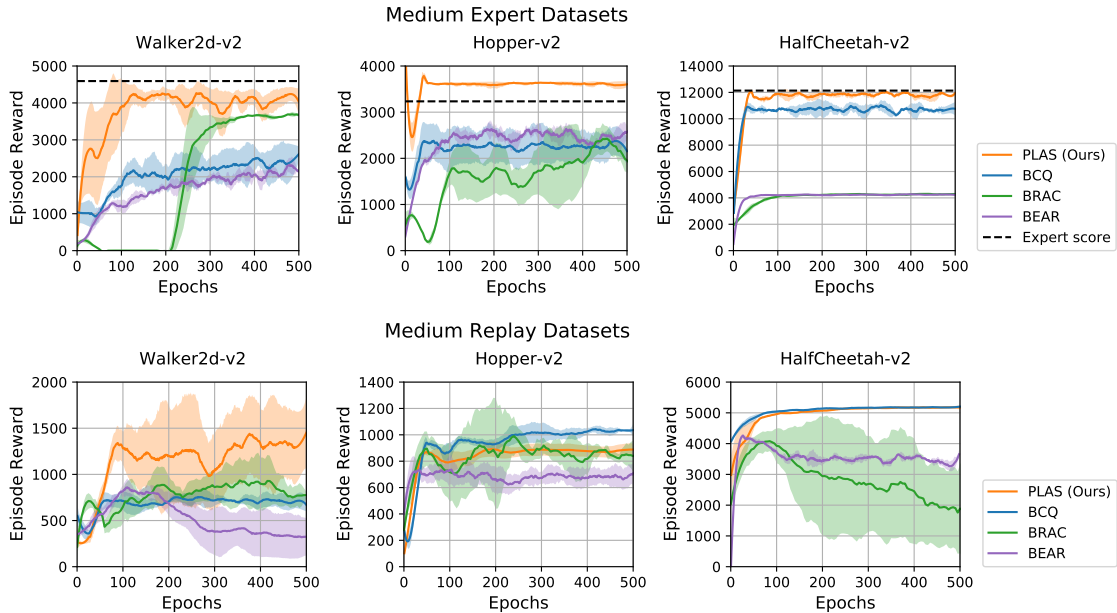


Figure 2.4: Training performance for medium-expert and medium-replay datasets on locomotion tasks. Each curve is averaged over 3 seeds. Shaded area shows one standard deviation across seeds.

training but it drops soon after. Our method outperforms all the baselines, and the final performance is similar to the expert policy.

2.5.3 Performance on D4RL datasets

To more systematically benchmark the performance of our method with the other offline RL algorithms, we ran experiments on the d4rl benchmarks. We focus the discussions and analysis on the locomotion environments here; full results on the d4rl datasets can be found in Appendix A.2 including Locomotion, Maze2d, AntMaze, Adroit Hand, Franka Kitchen environments. To highlight the performance of our method over the medium-expert datasets and the medium-replay datasets, we include the training curves in Figure 2.4. These two types of datasets are especially important because they have diverse coverage over states and actions generated by a mixture of policies. These kinds of diverse datasets are also more likely to appear in real-world applications with data collected from different sources. The diversity allows the potential to learn a good policy from the data; on the other hand, diversity also

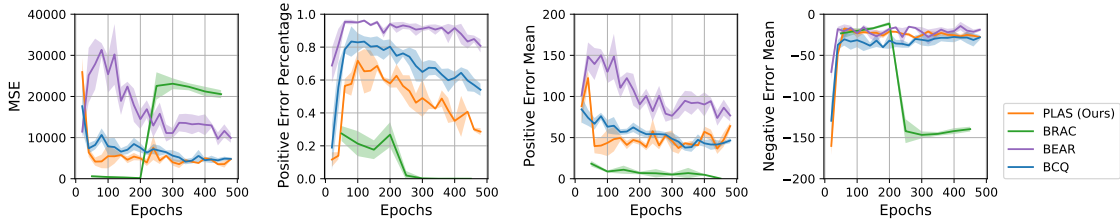


Figure 2.5: We perform an analysis of Q-function errors of different methods, using the following metrics: (a) Mean-squared error of the Q-values (b) The percentage of overestimated Q-values (c) Mean of the positive errors (magnitude of overestimation) (d) Mean of the negative errors (magnitude of underestimation)

introduces difficulties for the policy constraints. Figure 2.4 shows that we consistently achieve performance that is similar to or better than the best baselines on these datasets, demonstrating the effectiveness of our method in fully utilizing the datasets.

2.5.4 Overestimation of Learned Q-functions

We analyze the quality of the learned Q-function in detail with the Walker2d medium-expert dataset for different algorithms. By definition, the Q-value $Q_{\pi}(s_t, a_t)$ is equal to the expected return starting from state s_t following action a_t ; our learned Q-function attempts to estimate this value. Thus, we evaluate the Q-values by comparing them to the true returns for the transitions during rollouts. The true return is calculated by the cumulative discounted reward until termination or up to r_{t+1000} , since the reward after 1000 steps is negligible due to the discount factor. We define the estimation error to be $Q(s_t, a_t) - G(s_t, a_t)$, where $G(s_t, a_t)$ is the empirical return. Positive error corresponds to overestimation bias and negative error corresponds to underestimation bias. In Figure 2.5, we show four metrics on the quality of the Q-function over N transitions from 50 episodes:

- **Mean Squared Error (MSE)** - measures the overall quality: $\sum_i (Q(s_i, a_i) - G(s_i, a_i))^2$
- **Positive Error Percentage** - percentage of overestimation: $\frac{1}{N} \sum_i (\mathbb{1}(Q(s_i, a_i) - G(s_i, a_i) > 0))$
- **Positive Error Mean** - the mean value of the positive errors, which indicates the average magnitude of over-estimation: Average of $Q(s_i, a_i) - G(s_i, a_i)$ for

Table 2.1: Comparison of different perturbation values on random and medium datasets. Scores are normalized. $\epsilon = 0$ is the performance of the latent policy without the additional perturbation layer.

Dataset	$\epsilon = 0$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.5$
walker2d-random	3.1	6.8	2.4	1.3	-0.3
hopper-random	10.5	11.1	11.6	12.2	13.3
halfcheetah-random	25.8	25.7	27.4	27.6	28.3
walker2d-medium	44.6	64.8	66.9	62.1	39.2
hopper-medium	32.9	35.5	17.5	2.5	2.1
halfcheetah-medium	39.3	41.3	42.2	42.2	40.4

$$Q(s_i, a_i) - G(s_i, a_i) > 0$$

- **Negative Error Mean** - the mean value of the negative errors, which indicates the average magnitude of under-estimation: Average of $Q(s_i, a_i) - G(s_i, a_i)$ for $Q(s_i, a_i) - G(s_i, a_i) < 0$

MSE measures the overall estimation bias, and the other three metrics capture the direction of the bias. Our method achieves consistently low MSE during training compared with the baselines. Note that BRAC has low MSE during the beginning of training because the return is close to 0, as shown in the training curves in Figure 2.4. Although both our method and BRAC achieve similar performance at the end of training on the Walker2d medium-expert dataset (though our method converges faster), MSE indicates that our method results in a better Q-function. In terms of the direction of the bias, BEAR has a large overestimation bias and BRAC has a large underestimation bias. Overestimation bias is usually considered more harmful than underestimation for Q-learning based algorithms [30]. As a result, although BRAC has a higher MSE than BEAR, the evaluation performance is still better. Our method does not have significant underestimation or overestimation in this case.

2.5.5 Effect of the Optional Perturbation Layer

In Section 2.3.2, we mentioned that when the Q-function generalizes well, allowing the policy to select some out-of-distribution actions might be helpful. This motivates us to introduce an additional perturbation layer as mentioned in Section 2.4.2. We evaluate the benefit of this optional perturbation layer with different max perturbation limits,

with $\epsilon = 0$ being the latent policy alone. The results for a selective set of environments are summarized in Table 2.1. Note that the action space in these tasks is defined to be $(-1, 1)$; thus $\epsilon = 0.5$ allows a very high range of perturbation. We found that the importance of the perturbation layer depends on both the dataset and the environment. Allowing out-of-distribution actions often leads to improved performance for random datasets. The “medium” datasets tend to have peak performance with smaller values of ϵ ; a larger value likely leads to errors in the Q-function evaluation due to out-of-distribution state-action pairs. The full results including medium-expert and medium-replay datasets are in Appendix A.4. We found that medium-expert and medium-replay datasets usually do not benefit from the perturbation layer.

2.6 Conclusion

We propose a straightforward approach to offline RL that implicitly constrains the policy to be within the support of the dataset without being restricted by the density of the dataset distribution. Furthermore, we study the effect of an additional perturbation layer that allows out-of-distribution generalization of Q-functions. We demonstrate that our approach can effectively learn a policy with real-world data in the cloth sliding experiment and achieves competitive performance over offline RL benchmarks. By enabling a more efficient use of data from various sources, PLAS paves the way for future possibilities of using RL on real robots.

2. *PLAS: Latent Action Space for Offline Reinforcement Learning*

Chapter 3

Forgetting and Imbalance in Robot Lifelong Learning with Off-policy Data

3.1 Introduction

Lifelong learning, also commonly known as continual learning, studies the problem of learning with a stream of tasks sequentially with incremental, non-stationary data [35, 123]. Lifelong learning has been an important topic in artificial intelligence and it naturally reflects the challenges faced by animals and humans [37]. In this work, we study the problem of lifelong robot reinforcement learning in the face of changing environment dynamics (Figure 3.1). Non-stationary environment dynamics present a practical and important challenge for training reinforcement learning policies on robots in the real world. Especially on low-cost or low-tolerance robots, the robot dynamics can change due to wear and tear both during training and deployment. Also, in most

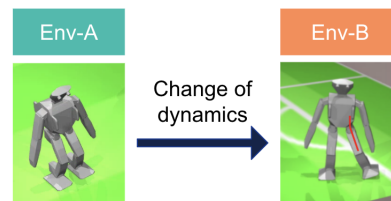


Figure 3.1: We investigate the problem of robot lifelong learning with non-stationary dynamics. In this example, a robot experiences joint deformation during training.

natural settings, the robot’s environment will change over time, for instance when the robot encounters new terrains or objects. Ideally, when the robot encounters the same or a similar environment again during deployment, it should be able to draw on the entirety of its past experience to retrieve the previously learned skill. In addition, one important property of sequential environment variations in the real world is that the task boundaries may be unknown or not well defined. For example, deformation of the robot can happen gradually. This limits the applicability of many existing approaches in lifelong learning which rely on well-defined task boundaries [60, 73, 106]. We aim to investigate a practical solution for lifelong robot learning across environment variations without the need of task boundaries.

One important challenge in lifelong learning is the trade-off between remembering the old task (backward transfer) and learning the new task efficiently (forward transfer). The most widely studied aspect of this trade-off is the catastrophic forgetting issue of neural networks [28]. We follow the memory-based method to avoid forgetting by simply saving all the incoming data in the replay buffer and train the policy with an off-policy algorithm, which does not require task boundaries [102]. However, we find that even if we save all the data across environment variations, “forgetting” still happens. In this case, the additional challenge of forgetting is due to the extrapolation error of the Q-function which is widely discussed in the Offline RL literature [31]: when the agent does not have access to the previous environments, it becomes “offline” over these environments. Thus, the agent cannot correct for the overestimation error of the Q-function by collecting more data in these environments. Conversely, if we use “conservative” (or “pessimistic”) algorithms that force the policy to stay close to the existing replay buffer to maintain the performance in the old environments, it affects data collection and creates difficulties in learning in the new environment [50]. Different from the stability-plasticity dilemma of neural networks often discussed in the lifelong learning literature, this trade-off between forward and backward transfer is specific to RL due to off-policy data. We propose the *Offline Distillation Pipeline* to disentangle this trade-off into two stages. To learn the task in the latest environment efficiently, we can use any RL algorithm suitable for online data collection without worrying about forgetting. To obtain a policy for deployment that effectively accumulates previous experience across environment variations, we can distill the entire dataset into a policy by treating it as an offline RL problem.

In addition, we investigate a practical consideration of lifelong learning where the stream of experience is imbalanced across environment variations. For example, the agent might be trained on one environment much longer than the other. The ideal lifelong learning algorithm should be robust to such imbalanced experience. In the Offline Distillation Pipeline, we find that training a policy with the imbalanced datasets from multiple environments can sometimes lead to much worse performance than training on each dataset individually. Through the experiments, we provide evidence for the following hypothesis: both the imbalanced quality and the imbalanced size of the datasets become extra sources of extrapolation error in offline learning. The imbalanced quality makes the Q-function biased towards larger values. The imbalanced size leads to more fitting error of the policy network on the smaller dataset, which exacerbates the bootstrapping error caused by out-of-distribution actions. Furthermore, we find that keeping the policy to be closer to the dataset could be a simple yet effective solution to this issue without requiring task boundaries.

In summary, we identify two practical challenges in lifelong robot learning over environment variations and provide corresponding analysis and solutions. The contributions of this work include the following:

- We identify the trade-off between learning in the new environments and remembering the old environments in existing off-policy RL algorithms even when all the data is kept in the replay buffer. We connect this problem to the Offline RL literature and propose the Offline Distillation Pipeline to break this trade-off without the need for task boundaries.
- We identify that the dataset imbalance can lead to unexpected performance drop in offline learning and characterize its relationship with extrapolation error with thorough empirical analysis.

We evaluate our method on a bipedal robot walking task in simulation with different environment changes. The proposed pipeline is shown to achieve similar or better performance than the baselines across the sequentially changing environments even with imbalanced experience.

3.2 Related Work

Lifelong Learning: Lifelong learning has been widely studied in machine learning literature [35, 56, 123]. When given a stream of non-stationary data or non-stationary tasks, the agent should maintain the performance of previous tasks (backward transfer) while learning the new task efficiently (forward transfer). One direction of lifelong learning literature focuses more on the issue of backward transfer caused by the catastrophic forgetting of neural networks [28]. Existing methods in this direction can be expansion-based [106, 108], regularization-based [60], gradient-based [73] or memory-based [102]. There has also been a line of work in task-agnostic continual learning [5, 6, 146], where the task boundaries are unknown or not well-defined. We follow the task-agnostic memory-based method from Rolnick et al. [102] by saving all the transitions in the replay buffer. In this work, we show that there are additional challenges in lifelong reinforcement learning besides the catastrophic forgetting issue of the neural networks. Another direction in lifelong learning focuses on maximizing forward transfer without worrying about forgetting where the performance is only measured by the new task. For example, recent work Xie and Finn [134] studies the problem of learning a sequence of tasks and proposes to selectively use past experience to accelerate forward transfer.

Offline Reinforcement Learning: The additional forgetting issue in off-policy reinforcement learning discussed in this work is related to offline RL. Thus, the proposed Offline Distillation Pipeline is based on this line of work. Offline RL investigates the problem of learning a policy from a static dataset without additional data collection [24, 65, 67]. Such a problem setting challenges existing off-policy algorithms due to the mismatch between the state-conditioned action distribution induced by the policy and the dataset distribution [31]. Previous work has proposed to fix this issue by constraining the policy to be close to the dataset explicitly [50, 62, 112, 130, 132] or implicitly [31, 151], learning a conservative Q-function [63], or modifying the reward based on model uncertainty [57, 141]. We use Critic Regularized Regression (CRR) from Wang et al. [130] to perform offline distillation. In terms of related work in imbalanced dataset in offline RL, Zhang et al. [147] investigates imbalanced offline datasets collected by a variety of policies, in contrast to having a mixed

dataset from multiple environments in our case. While most of the work in offline RL focuses on one task, Yu et al. [142] studies multi-task offline RL with the goal of improving single task performance by selectively sharing the data across tasks. In contrast, we aim at learning a universal policy for all the tasks which does not rely on task boundaries during training. The difference in the objectives is mainly due to the difference in the domains of interests: the “tasks” are defined to have different reward functions in Yang et al. [137] while defined to be different dynamics in our case.

Distillation: The proposed pipeline is also related to knowledge distillation [41]. In RL, policy distillation has been used to compress the network size [105, 108], improve multitask learning [122, 126], or improve generalization [48]. In contrast to policy distillation methods which distill the knowledge from networks to networks, we directly distill the data into a policy. This eliminates the need of task boundaries and additional data collection in previous methods. Nonetheless, the proposed pipeline with offline distillation may still share similar benefits of modifying network size or improving generalization because it trains a new policy from scratch [48].

3.3 Preliminaries

3.3.1 Problem Definition: Lifelong reinforcement learning with environment variations

We define the lifelong learning problem across environment variations to be a time-varying Markov Decision Process (MDP) \mathcal{M} as a tuple $(\mathcal{S}, \mathcal{A}, P_t, r, \gamma)$, with state space \mathcal{S} , action space \mathcal{A} , non-stationary dynamics function $P_t : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, \infty)$ that may change over time t , reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [r_{min}, r_{max}]$, and discount factor γ . In contrast to our work, most reinforcement learning literature considers static dynamics, which is a special case when $P_t = P$ for all t . In reinforcement learning, the objective is to optimize the policy to maximize the return given by $G_t = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}, s_{t+k+1})$. We also define a policy $\pi(a|s)$ and its corresponding Q-function $Q^\pi(s, a) = \mathbb{E}_{\pi, P_t}[G_t | s_t = s, a_t = a]$ where the expectation is taken over the trajectories that start from an initial state s , an initial action a , and follow $s_{t+1} \sim P_t(s|s_t, a_t)$ and $a_{t+1} \sim \pi(a|s_{t+1})$ for the following timesteps.

In this work, we assume that the agent experiences P_t for a fixed amount of time $[0, T]$ during training, and will be evaluated and deployed at time T . This involves efficient data collection across $[0, T]$ and being able to recall the skills at time T . We formulate our problem as maximizing the return of the policy over the support of the environment distribution $p(P_{t=[0,T]})$. Intuitively, although the agent might experience one environment more than the other during training, we treat different environments as equally important. For example, the agent experiences $P_t = P_A$ for $t \leq t_A$ and then experiences $P_t = P_B$ for $t_A < t \leq t_A + t_B$. In this case, the objective can be defined as maximizing the performance of $\mathbb{E}_{P_A}[G] + \mathbb{E}_{P_B}[G]$ during evaluation at the end of training. We aim to learn a policy that works well on both P_A and P_B regardless of t_A and t_B . Although we have two distinct stages with two environments in this example, in general the task boundaries may not always be accessible or well-defined since P_t can change continuously. Without task boundaries, we cannot directly optimize the policy over the support of P_t instead of the density of P_t . However, we aim to treat the importance of different environments equally during evaluation.

3.3.2 Off-Policy Reinforcement Learning Algorithms

The proposed Offline Distillation Pipeline is built on top of two RL algorithms: Maximum a Posteriori Policy Optimisation (MPO) [1, 2] and Critic Regularized Regression (CRR) [130]. As will be discussed later in Section 3.5, our pipeline uses MPO to update the policy during data collection, and uses CRR for offline distillation. Although MPO and CRR are both off-policy RL algorithms and share a lot of similarities, they are designed for different problem settings. MPO works well in the online setting, i.e. when data collection is allowed. CRR is designed for offline reinforcement learning, i.e. to learn from a fixed dataset without additional data collection. To stabilize learning in the offline setting, CRR attempts to avoid selecting actions outside of the dataset, which also renders it more “conservative”. More discussion of the connections between CRR and MPO can be found in Abdolmaleki et al. [3], Jeong et al. [50]. Both algorithms alternate between policy evaluation and policy improvement. Both algorithms perform **policy evaluation** to estimate the Q-function $\hat{Q}^\pi(s_t, a_t)$ using the Bellman Operator \mathcal{T} :

$$\mathcal{T}\hat{Q}^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1} \sim \pi(a|s_{t+1})}[r_t + \gamma\hat{Q}^\pi(s_{t+1}, a_{t+1})]. \quad (3.1)$$

We will discuss their differences in the **policy improvement** step below.

Maximum a Posteriori Policy Optimisation (MPO): To improve the current policy $\pi_{old}(a|s)$ given the corresponding Q-function $Q^{old}(a, s)$ and a state distribution $\mu(s)$, MPO performs two steps. In the first step, for each state $s \sim \mu(s)$, an improved policy $q(a|s) \propto \pi_{old}(a|s)f(Q^{old}(s, a))$ is obtained where f is a transformation function that gives higher probabilities to actions with higher Q-values. In the second step, a new parametric policy π_{new} is obtained by distilling the improved policies $q(a|s)$ into a new parametric policy using the supervised learning loss:

$$\pi_{new} = \operatorname{argmax}_{\pi} \int \mu(s) \int q(a|s) \log \pi(a|s) f(Q^{old}(s, a)) da ds \quad (3.2)$$

In practice, we represent $q(a|s)$ as a non-parametric policy consists of samples from $\pi_{old}(a|s)$ for each state and re-weighting each sample by $f(Q^{old}(s, a))$. If the exponential function is chosen as the transformation function f , the improved policy can be written as $q(a|s) \propto \pi_{old}(a|s) \exp(Q^{old}(s, a)/\beta)$ where β is the temperature term. This is the solution to the following KL regularized RL objective that keeps the improved policies q close to the current policy π_{old} while maximising the expected Q-values:

$$q = \operatorname{argmax}_q \mathbb{E}_{s \sim \mu(s)} [\mathbb{E}_{a \sim q(\cdot|s)} [Q^{old}(s, a)] - \beta \text{KL}(q(\cdot|s) || \pi_{old}(\cdot|s))] \quad (3.3)$$

Critic Regularized Regression (CRR): CRR follows a similar procedure as MPO for the policy improvement step. The major difference is the way of constructing the improved policy q . Since CRR is designed for offline RL, the optimization objective is to improve the policy according to the Q-function while keeping the policy close to the dataset distribution. In CRR, we construct the improved policies based on the joint distribution of $\mu_{\mathcal{B}}(a, s)$ by sampling state-action pairs from the dataset \mathcal{B} . Thus, the improved policy for each state $s \sim \mu_{\mathcal{B}}(s)$ is defined as a joint distribution $q(a, s) \propto \mu_{\mathcal{B}}(a, s) f(Q^{old}(s, a))$ instead of a conditional distribution $q(a|s)$ as in MPO. Similarly, Equation 3.2 can be modified to obtain a new parametric policy π_{new} :

$$\pi_{new} = \operatorname{argmax}_{\pi} \int q(a, s) \log \pi(a|s) da ds \quad (3.4)$$

When the transformation function is the exponential function, the improved policy can be written as $q(a, s) \propto \mu_{\mathcal{B}}(a, s) \exp(Q^{old}(s, a)/\beta)$ which is a solution to the following objective similar to Equation 3.3:

$$q = \operatorname{argmax}_q \mathbb{E}_{s,a \sim q}[\hat{Q}^\pi(s, a)] - \mathbb{E}_{s,a \sim \mathcal{B}}[\beta \text{KL}[q(a, s) || \mu_{\mathcal{B}}(a, s)]]. \quad (3.5)$$

In Equation 3.5, when the temperature β is higher, the constraint on staying close to the dataset becomes stronger, which makes the policy more “conservative”. A common practice is to replace the Q-value by the advantage in the transformation function: $q(a, s) \propto \mu_{\mathcal{B}}(a, s) \exp(A^{old}(s, a)/\beta)$. Besides the exponential function, another popular choice of the transformation is the indicator function: $q(a, s) \propto \mu_{\mathcal{B}}(a, s) \mathbb{1}[A^{old}(s, a) > 0]$ where A is the advantage function. The indicator function corresponds to an exponential transformation clipped to $[0, 1]$ with $\beta \rightarrow 0$ which is less “conservative”.

3.4 Forward and Backward trade-off in Lifelong Reinforcement Learning

To build a pipeline for lifelong learning, we need to first deal with the catastrophic forgetting issue of neural networks, as widely discussed in the literature [35]. We follow the memory-based approach from Rolnick et al. [102] by saving all the transitions across the agent’s life-cycle and run off-policy algorithms such as MPO [2]. In off-policy algorithms, the policy is used for exploration in the latest environment while being trained on the entire history of data. However, we still observe that “forgetting” happens in the old environment following this setup. Figure 3.2 shows an example in which the policy experiences a change in the environment dynamics at 200k steps while being evaluated in the first environment across the full training process. More details of the experiment can be found in Section 3.7.1. Once the policy

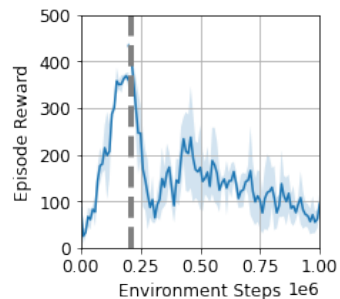


Figure 3.2: Forgetting in MPO: The figure shows a performance drop in the original environment after a switch of environment at 200k steps.

starts training in a new environment, the performance of MPO drops significantly even if the data from the old environment is kept in the replay buffer. We identify this extra challenge in lifelong reinforcement learning besides the catastrophic forgetting issue of the neural network.

The reason behind this drop is related to the issues of applying off-policy algorithms to Offline Reinforcement Learning problems [31]. The objective of offline RL is to learn a policy from a fixed dataset without further exploration. During training, due to the extrapolation error in the Q-function, the policy might select overestimated actions beyond the dataset. This error will be accumulated by bootstrapping during Q-function updates which results in significant overestimation bias of the Q-function. When the agent does not have access to collect more data to correct the overestimation bias, the performance of the policy will drastically degrade. Thus, off-policy algorithms designed with the assumption of active data collection often break under this problem setting. Similarly, in the lifelong learning scenario discussed above, when the agent switches from one environment to another, it is essentially training over the static dataset of the old environment. When the agent cannot collect more data in the old environment, it cannot correct the extrapolation error on those state-action pairs.

Prior work in offline RL proposes to fix the overestimation issue of off-policy algorithms by restricting the policy $\pi(a|s)$ to be closer to the conditional distribution $\mu_B(a|s)$ of the dataset, such as Critic Regularized Regression (CRR) as described in Section 3.3.2. If we apply a similar “conservative” objective in the lifelong learning setting, we find that it is able to reduce the forgetting issue. However, it will instead affect forward transfer due to the conservatism. Figure 3.3 shows an example of running CRR from scratch, which can be viewed as the beginning stage of a lifelong learning experiment. Although CRR has been shown to have strong performance on offline RL benchmarks, it does not have good performance when exploration is needed due to the constraint on the policy. In the experiment, we will further show that tuning the constraint will lead to either forgetting or ineffective forward transfer.

The above examples demonstrate the trade-off between preserving performance

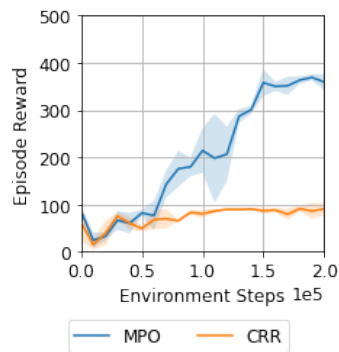


Figure 3.3: CRR is not as efficient as MPO when training from scratch.

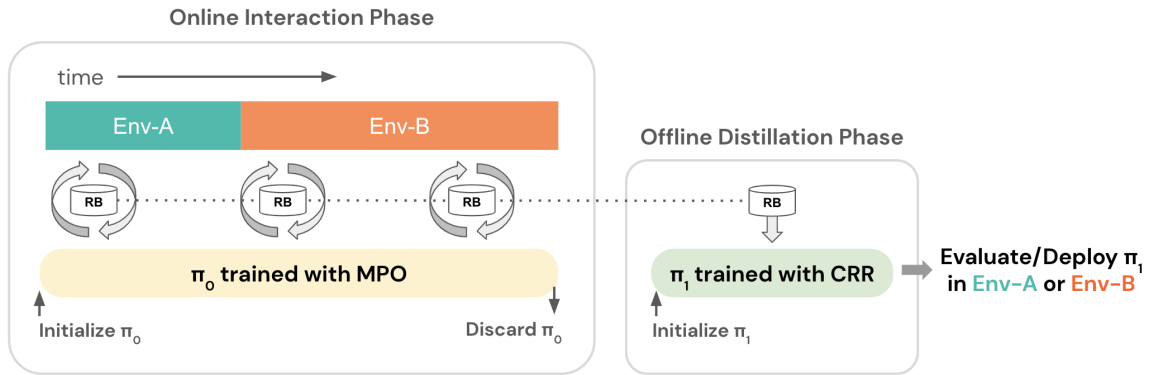


Figure 3.4: We propose the Offline Distillation Pipeline: the agent is trained over the environments sequentially during online interaction phase and runs offline distillation at the end of training before deployment. “RB” means replay buffer in the figure.

in the old environment (backward transfer) and exploring in the new environment effectively (forward transfer). Note that this is different from the “stability-plasticity” dilemma in previous lifelong learning literature in two ways. First, in terms of backward transfer, the issue of forgetting rises from the extrapolation error of Q-function which is specific to off-policy reinforcement learning. Second, in terms of forward transfer, previous work mainly considers the trade-off between past and *recent* experience from the streaming data. The issue we discuss above is a trade-off between past and *future* experience which is specific to reinforcement learning where the performance highly depends on effective data collection.

3.5 Offline Distillation Pipeline

To address this trade-off, we propose the Offline Distillation Pipeline shown in Figure 3.4. During data collection across environment variations, we can use any RL algorithm that maximizes forward transfer without considering forgetting. At the end of training, we “distill” the experience into a single policy by treating the entire dataset as an offline RL dataset. In this work, we use MPO to train the policy for data collection, and use CRR during offline distillation. In this way, the forgetting issue of the off-policy data is handled by the distillation step without affecting exploration. The full algorithm can be found in Appendix B.2.

There are several benefits of this pipeline that are especially important for lifelong

learning of real robots. First, the proposed pipeline does not require task boundaries. The wear and tear of the robot might happen over time and sometimes the change of the environment might not be immediately noticeable. This is different from a common multi-task learning setting where the task switches are well defined (such as learning to stand up and then learning to walk). In our method, the distillation step happens after the agent has experienced the sequence of environments and treats the replay buffer as a single dataset. Second, our method is flexible on the choice of data collection methods since the offline distillation phase only shares the replay buffer dataset with the online interaction phase. For example, the training of multiple robots can happen in parallel or sequentially, and potentially with different choices of algorithms. The Offline Distillation Pipeline can reuse all of these previous experience within the “lifetime” of the platform.

3.6 Imbalanced Experience in Offline Distillation

The second challenge we identify in such a lifelong learning setting is the issue of imbalanced experience with offline data. In the offline distillation phase, we find that when the policy is trained over the combined dataset from multiple environments, the imbalance of the datasets might create an unexpected performance drop. For example, following Figure 3.4, the agent is first trained in Env-A and then switches to Env-B. During the offline distillation phase, we use CRR to train a policy with the combined dataset $\mathcal{D}_A \cup \mathcal{D}_B$ and evaluate the performance in both environments, as formulated in Section 3.3.1. We find that this sometimes results in worse performance in Env-A compared to training on \mathcal{D}_A alone. Although previous work has studied the problem of data imbalance in supervised learning [53, 101], the issue we observe has the extra complexity from the bootstrapping procedure in off-policy RL. We provide evidence to the following hypothesis: Both the imbalanced **quality** and the imbalanced **size** of the combined dataset lead to additional extrapolation error of the Q-function in offline learning which contribute to the performance drop. As we discussed in Section 3.4, extrapolation error of the Q-function plays an important role in the failure cases in offline RL. The imbalanced dataset exacerbates this problem in the following way: if one dataset has a higher average return than the other, it may cause overestimation bias of the Q-function for the “weaker” dataset. At the

same time, if there is a large size imbalance, the policy network will be trained with more data points from one environment than the other. In this way, the policy may create more out-of-distribution actions in the environment that comes with a smaller dataset which makes the extrapolation error worse. Both of these two aspects contribute to the undesirable performance we observe in offline distillation phase. In the experiment, we provide evidence to support this hypothesis and eliminate other potential factors.

To build a robust algorithm for lifelong learning, we need to improve the offline distillation phase to achieve good performance on all of the environments despite imbalanced experience. We prefer a solution that does not rely on task boundaries as discussed before. Our insight is that since both the quality imbalance and the size imbalance eventually result in additional extrapolation error, we can follow the conservative objective in offline RL and make the policy even more conservative to compensate for this issue. As shown in Equation 3.5, the temperature β controls the strength of the KL term in the policy improvement objective in CRR. With a larger temperature, the policy is constrained to be closer to the behavior policy $\mu_{\mathcal{B}}(a|s)$ of the dataset. We find that the imbalanced dataset requires a higher strength of the KL term compared to single dataset training to compensate for the additional extrapolation error. In the experiment, we show that increasing β is a simple yet effective fix to the data imbalance problem. The effectiveness of increasing β can also serve as an evidence that the performance drop is highly related to extrapolation error. Note that increasing β only makes the policy more “conservative” during the distillation phase which will not affect exploration.

3.7 Experiments

3.7.1 Experiment Setup

We study the lifelong learning problem in a simulated bipedal walking task, where the goal is to maximize the forward velocity while avoiding falling. Our experiments involve a small humanoid robot, called OP3¹, that has 20 actuated joints and has been previously used to train walking directly on hardware [8]. All of the experiments

¹<https://emanual.robotis.com/docs/en/platform/op3/introduction/>

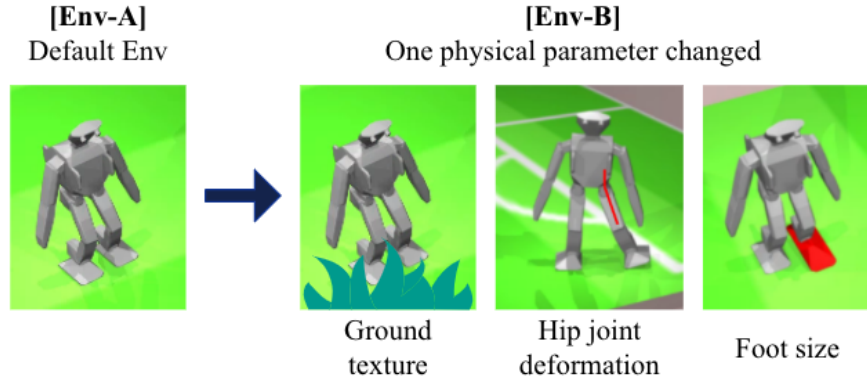


Figure 3.5: Experiment Setup: A bipedal robot walking task with various environment variations.

in this work are conducted in simulation both due to limited access to hardware and for a more controlled experiment setting. All of the results are averaged over 3 random seeds. The shaded area of the curves and the error bars of the bar plots represent \pm one standard deviation across seeds.

The experiments in the section are based on the setup where the robot is trained in Env-A for 0.2M steps, and then trained in Env-B for 1M steps (Figure 3.4). The goal is to achieve good performance at the end of training in both Env-A and Env-B. Additional experiment setups with three environments and parallel sharing can be found in Appendix B.3. To evaluate the generality of the results, we consider different types of changes in the environment including softer ground texture, hip joint deformation and larger foot size (Figure 3.5). The parameters for each change of the environment are chosen to create a clear performance drop when we perform zero-shot transfer of a policy trained in the default environment to the new environment. In the following experiments when there is a switch from Env-A to Env-B, we use the default environment as Env-A, and change one of the physical parameters to create Env-B. When we switch from one environment to another, we always keep the previous policy, Q-function and the replay buffer.

To remove partial observability in non-stationary dynamics, we include the ground truth physical parameters in the observation. This eliminates the possibility that the issue we observe in the lifelong learning pipeline and the imbalanced experience are caused by the partial observability. The results we provide can be served as an upper

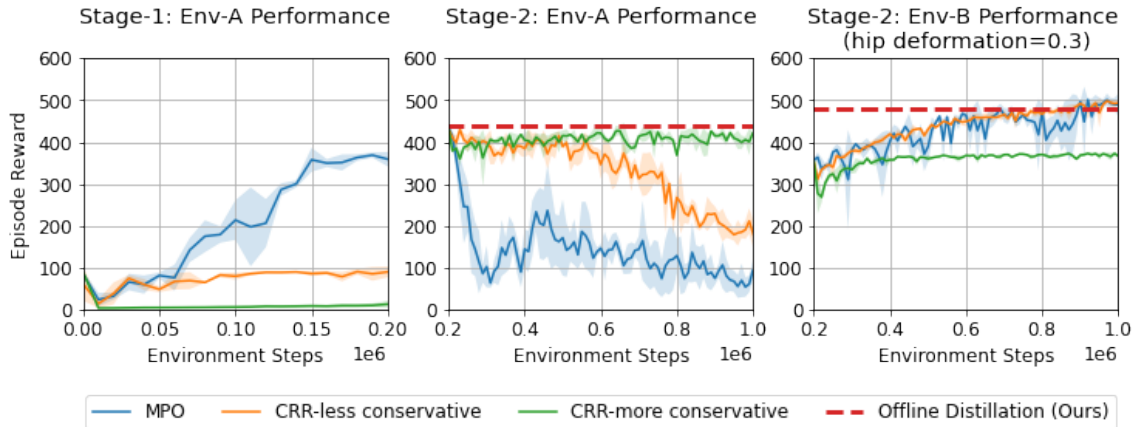


Figure 3.6: Evaluation curves of the lifelong learning experiment across two stages. The Offline Distillation Pipeline effectively breaks the trade-off between forward and backward transfer and achieves better performance than the baselines at the end.

bound on the expected performance without the physical parameters. However, the physical parameters are not explicitly used to denote task boundaries in the proposed method. In a realistic scenario, the partial observability could be handled by memory or system identification methods [39, 143, 150]. In these cases, the variations of the environments might be represented as continuous embeddings which cannot be used as task boundaries. Thus, we avoid relying on the physical parameters in the proposed pipeline as task boundaries.

3.7.2 Offline Distillation for Lifelong Reinforcement Learning

In this section, we demonstrate the trade-off between forward transfer and backward transfer in lifelong reinforcement learning and the effectiveness of the Offline Distillation Pipeline. The policy is trained from scratch in Env-A during Stage-1 and then switched to Env-B during Stage-2 (as illustrated in Figure 3.4). We compare the performance of MPO, CRR with a less conservative objective (with an indicator function which corresponds to $\beta \rightarrow 0$ as discussed in 3.3.2), CRR with a more conservative objective ($\beta = 1$) and the Offline Distillation Pipeline. In the example shown in Figure 3.6, the robot in Env-B has its right hip joints deformed for 0.3 rad. The results for more Env-B variations are included in Appendix B.1.

We first demonstrate the forward transfer problem in Stage-1. As shown on the left figure in Figure 3.6, conservative algorithms such as CRR cannot learn as efficiently as MPO from scratch. Then, we show the backward transfer problem in Stage-2. To compare the performance drop better, we enforce the same starting performance of Stage-2 for all the baselines. This is done by loading the same agent (networks and replay buffer) trained with MPO from Stage-1. All the baselines keep training on the loaded replay buffer with Env-A data while collecting new data in Env-B with the latest policy. In Stage-2, the policy only collects data in Env-B, but we evaluate the performance in both Env-A and Env-B. From the middle figure in Figure 3.6, the performance of MPO drops significantly in Env-A after the switch. This could potentially be explained by the fact that Env-A transitions are “offline” during Stage-2 and thus the extrapolation error starts to accumulate. The forgetting issue also happens in the less conservative CRR, despite being less severe than MPO. The more conservative CRR keeps the performance of Env-A effectively which indicates that the performance drop is indeed related to the extrapolation issue in offline RL. However, as shown in the right figure in Figure 3.6, more conservative CRR does not improve the performance in Env-B as the other baselines.

In summary, these baselines either struggle with backward transfer or forward transfer. As described in Section 3.5, we propose the Offline Distillation Pipeline which distills the data collected by MPO using CRR. In this experiment, we perform the distillation step at the end of Stage-2. The performance is shown as the dotted lines in the figures. Taking the best of both world, our method can achieve better performance than the baselines in both environments. Note that in this experiment we include the results of the proposed method with the data imbalance issue fixed which will be discussed in the next section.

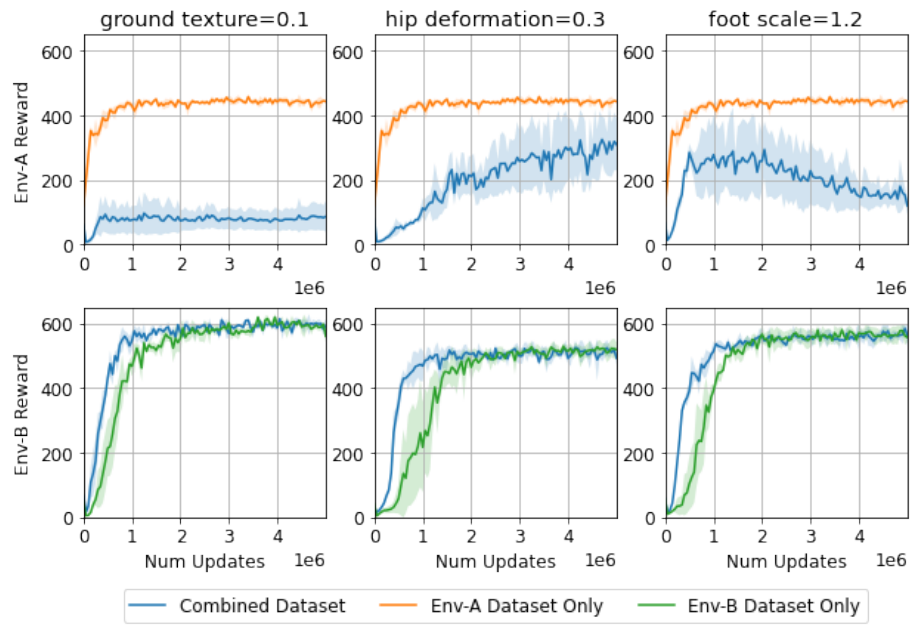
3.7.3 Imbalanced Experience in Offline Distillation

As discussed in Section 3.6, we sometimes observe a performance drop during the distillation phase in the proposed pipeline with imbalanced experience. We will provide experimental evidence for the hypothesis that the decrease in performance is caused by the imbalanced size and quality between the datasets. In this section, we use CRR with the indicator function by default which corresponds to a less

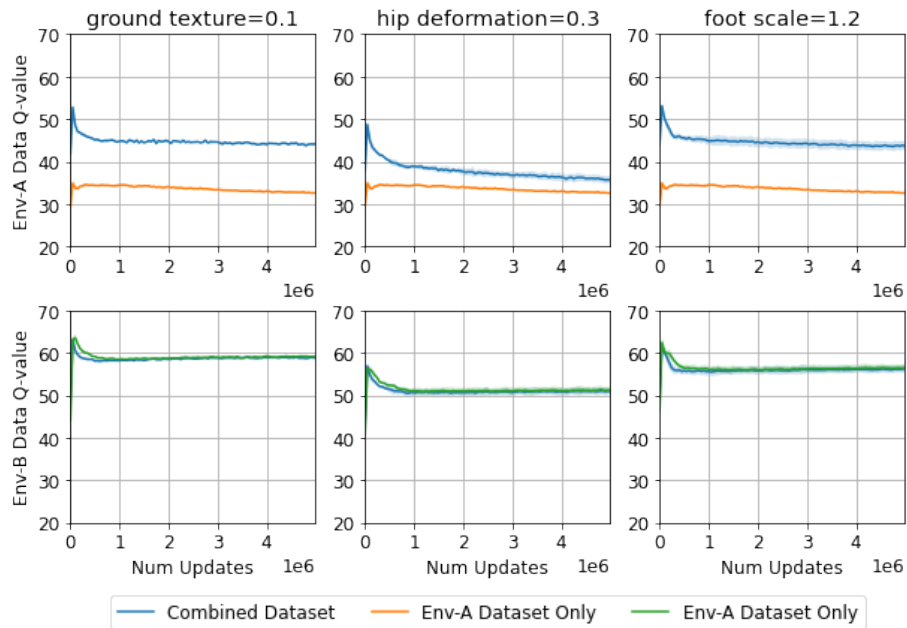
conservative CRR objective as discussed in Section 3.3.2.

The Imbalance Issue. Following the setup in the previous section (Figure 3.4), we run the offline distillation step at the end of Stage-2 with the combination of dataset \mathcal{D}_A in Env-A from Stage-1 and \mathcal{D}_{B_i} in Env-B from Stage-2, both collected by MPO. We apply different environment variations in Env-B to generate \mathcal{D}_{B_i} with different i while keeping \mathcal{D}_A the same in the combined dataset (Figure 3.5). In Figure 3.7a, the blue curves (**Combined Dataset**) show the performance of the distilled policy $\pi_{\mathcal{D}_A \cup \mathcal{D}_{B_i}}$ evaluated in Env-A and Env-B across the CRR training process of the distillation stage. Each column corresponds to a different \mathcal{D}_{B_i} . Note that there is no data collection in this stage and the x-axis here is the number of policy updates in CRR rather than environment steps. As a comparison, we run CRR on \mathcal{D}_A and \mathcal{D}_{B_i} separately with the same hyperparameters (**Env-A Dataset Only**, **Env-B Dataset Only**). Given that there is no partial observability (see Section 3.7.1), we expect $\pi_{\mathcal{D}_A \cup \mathcal{D}_{B_i}}$ to have similar or better performance than $\pi_{\mathcal{D}_A}$ evaluated in Env-A, and $\pi_{\mathcal{D}_{B_i}}$ evaluated in Env- B_i . From the second row of Figure 3.7a, the performance in Env- B_i is similar between $\pi_{\mathcal{D}_A \cup \mathcal{D}_{B_i}}$ and $\pi_{\mathcal{D}_{B_i}}$ at convergence. However, from the first row of Figure 3.7a, the performance of $\pi_{\mathcal{D}_A \cup \mathcal{D}_{B_i}}$ in Env-A is much worse than training with \mathcal{D}_A alone: we observe the blue curves to converge at a lower reward, converge much slower, or becomes unstable during training. Despite being trained on the same \mathcal{D}_A , the distilled policies $\pi_{\mathcal{D}_A \cup \mathcal{D}_{B_i}}$ have very different performance in Env-A due to the fact that they are combined with different \mathcal{D}_{B_i} . Although the performance drop in Env-A does not always happen, it is important to understand when and why the performance degrades to develop a robust lifelong learning pipeline that works for diverse settings. To get more insights of the problem, we also train a behavior cloning policy with the combined dataset. Figure 3.8 includes the final performance of CRR (**Baseline**) and behavior cloning (**BC**) over the combined dataset. Despite the size imbalance, with only supervised learning, BC performs reasonably well in Env-A. The CRR Baseline is much worse than BC in Env-A. This comparison indicates that the performance drop we observe in Env-A is more likely to be rooted in the RL procedure, instead of being a regular data imbalance problem in a supervised setting. We have also tried a few sanity check experiments including increasing batch size, increasing network capacity, or using a mixture of Gaussians as the policy output. None of these can prevent the performance drop in Env-A. In the following sections,

3. Forgetting and Imbalance in Robot Lifelong Learning with Off-policy Data



(a) Reward



(b) Q-values

Figure 3.7: The imbalance issue in Offline Distillation: Training CRR with the combined dataset results in much lower performance in Env-A compared to training on the individual dataset. However, the worse performance corresponds to a higher average Q-value which indicates overestimation.

3. Forgetting and Imbalance in Robot Lifelong Learning with Off-policy Data

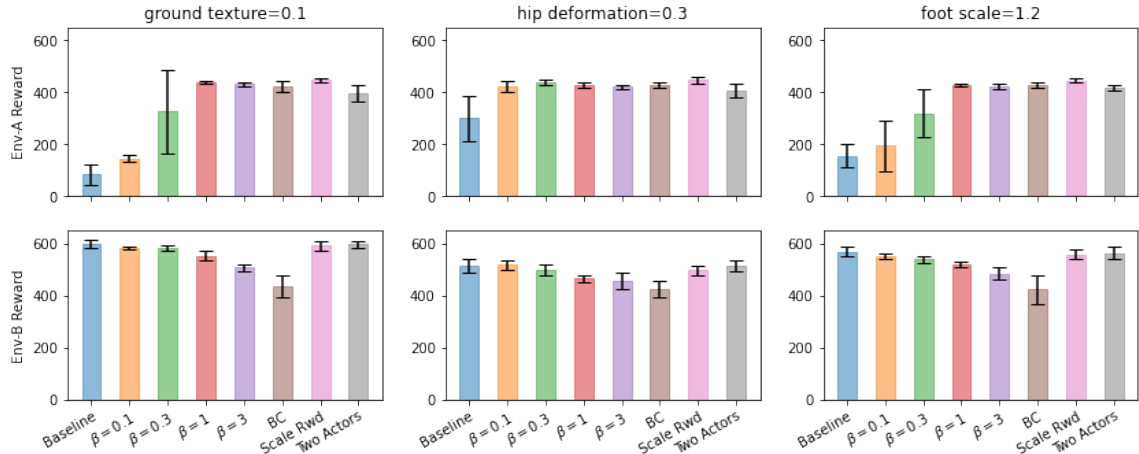


Figure 3.8: To study the imbalance issue, the figure shows the final performance of different variants of the offline distillation step at $5e6$ policy updates. The **Baseline** corresponds to the performance of **Combined Dataset** in Figure 3.7.

we will discuss the most important experiments that can support our hypothesis discussed in Section 3.6.

Overestimation due to the imbalanced quality. Figure 3.7b plots the average estimated Q-values of each method over $(s, a) \sim \mathcal{D}_A$ and $(s, a) \sim \mathcal{D}_{B_i}$ separately. Although $\pi_{\mathcal{D}_A \cup \mathcal{D}_B}$ has a lower performance in Env-A than $\pi_{\mathcal{D}_A}$, the corresponding Q-functions $Q^{\pi_{\mathcal{D}_A \cup \mathcal{D}_{B_i}}}$ produce higher estimated Q-values over Env-A datapoints, which indicates significant overestimation. In contrast, if we compare the Q-values over $(s, a) \sim \mathcal{D}_{B_i}$ on the second row, the curves are similar to each other within each plot. This indicates that $Q^{\pi_{\mathcal{D}_A \cup \mathcal{D}_{B_i}}}$ suffers from overestimation specifically for Env-A data points. Furthermore, we observe that the average Q-value over \mathcal{D}_{B_i} is higher than \mathcal{D}_A for the individual dataset experiments. This is because \mathcal{D}_A is collected by training from scratch, while \mathcal{D}_{B_i} is collected during Stage-2 where the policy is bootstrapped from the previous experience and starts from a higher performance (see Figure 3.6). This observation leads us to the hypothesis that the high value datapoints in Env-B bias the Q-function which leads to overestimation for Env-A datapoints. To verify this hypothesis, we perform an experiment where we scale the reward for all the transitions in \mathcal{D}_B by 0.5, which does not change the optimal solution of the policy. After this change, the distilled policy with the combined dataset works well on both environments (**Scale Rwd** in Figure 3.8) and achieves similar performance as

the individual dataset baselines. However, re-scaling the reward is not an acceptable solution in our problem setting because it requires the knowledge of task boundaries. It only serves as an analysis to demonstrate the imbalance issue due to the quality of the dataset.

Additional fitting error of the actor. We also test the contribution of the fitting error of the actor to the overall extrapolation error. We use separate actor networks for each dataset when training CRR on the combined dataset $\mathcal{D}_A \cup \mathcal{D}_{B_i}$ (**Two Actors** in Figure 3.8): Actor-A and Actor-B are trained with the transitions from Env-A and Env-B respectively, while the critic is shared across two environments. This change also makes the policy work well in both environments despite that the imbalanced reward is not corrected. In the Two Actors experiment, we find that the overestimation over \mathcal{D}_A still exists but has been reduced. Together with the Scale Reward experiment, the results indicate that the overestimation we observe in Figure 3.7b in Env-A is caused by two sources of error: the imbalanced quality creates overestimation; The imbalanced size creates more fitting error of the actor which results in more out-of-distribution actions that may take advantage of the overestimation. Note that using two separated actors also requires task boundaries and only serves as an analysis.

Effectiveness of the temperature. As shown in previous sections, fixing either the imbalanced quality or the fitting error of the actor makes the algorithm stable when evaluated in both Env-A and Env-B. However, we need a solution that does not require task boundaries. As proposed in Section 3.6, increasing the temperature term β in CRR can largely fix this issue. Figure 3.8 includes the performance of CRR with different β . **Baseline** uses an Indicator function as the transformation function which corresponds to very small β . With increased β , the performance in Env-A increases. Although we observe a minor drop in Env-B with high β , the overall performance in both Env-A and Env-B are reasonably satisfactory. To further demonstrate the effectiveness of increasing β , we conduct an experiment where we upsample either \mathcal{D}_A or \mathcal{D}_{B_i} to simulate other compositions of the combined dataset (Figure 3.9). As mentioned in Section 3.7.1, the size ratio of $\mathcal{D}_A : \mathcal{D}_{B_i}$ is 1 : 5 (denoted as **raw**). The performance in Env-A of CRR with the indicator function (**Baseline**) decreases drastically with higher Env-B sampling ratio. Interestingly, when the size ratio is 1 : 1, the policy is still not able to consistently achieve the single dataset performance

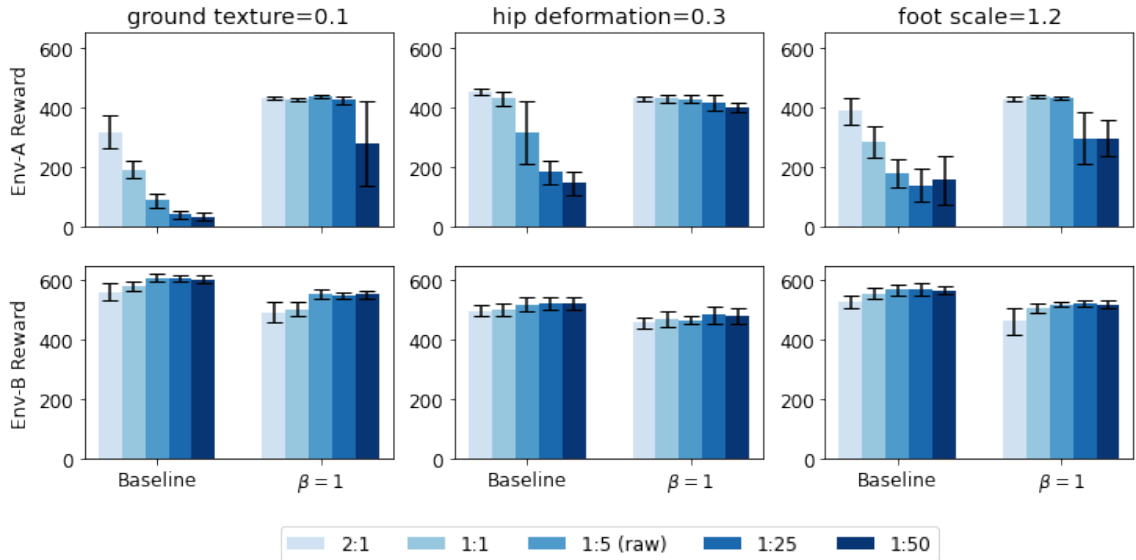


Figure 3.9: Sensitivity analysis of data imbalance: Higher β makes CRR more robust to different ratios of dataset imbalance. The legend indicates the dataset size ratio of Env-A:Env-B corresponding to different colors.

in Env-A (which is expected to be above 400 as shown in Figure 3.6). In contrast, CRR with $\beta = 1$ works well across a wider range of size ratios (which is what we use in Section 3.7.2). As shown in previous work [130], the specific choice of β could be domain-dependent. The more important takeaway from this analysis is that if we observe a performance drop during RL training with an imbalanced dataset, we may consider increasing the conservativeness of the policy to compensate for the additional extrapolation error, such as increasing β in CRR.

3.8 Conclusion

In this work, we investigate the lifelong learning problem of variations in environment dynamics as commonly observed when learning on robot hardware. Our main contributions include identifying and addressing two challenges within such a problem setting: First, we find that there is a trade-off between backward and forward transfer of existing RL algorithms in this problem setting even when we keep all of the transitions in the replay buffer. We connect the problem to offline RL and

propose the Offline Distillation Pipeline to break this trade-off. In the proposed pipeline, the forgetting issue is prevented by distilling the replay buffer data across multiple environments into a universal policy as an offline RL problem. In this way, the solution to the forgetting problem is disentangled from data collection. We empirically verify the effectiveness of the pipeline through a bipedal robot walking task in simulation across various physical changes. Second, we identify an potential issue with imbalanced experience in offline distillation. Through controlled experiments, we demonstrate how the quality imbalance and the increased fitting error of the actor might exacerbate extrapolation error and create a performance drop. We also provide a simple yet effective solution to this issue by increasing the temperature term in CRR.

The insights from this work could potentially be applied in other settings beyond the lifelong learning problem of varying dynamics. For example, the Offline Distillation Pipeline can be used in other lifelong reinforcement learning settings with a different definition of “task” without varying dynamics. The imbalance issue may also happen in other cases of multi-task learning in offline RL, or in single-task RL with sufficient non-stationarity (e.g. due to partial observability). In future work, we hope to see the proposed method being verified and deployed in more settings including having multiple distillation steps across the training procedure or real robot experiments.

3. Forgetting and Imbalance in Robot Lifelong Learning with Off-policy Data

Chapter 4

Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity

4.1 Introduction

Humans have dexterous multi-fingered hands; however, similarly dexterous robot hands are expensive and fragile. Instead, a simple hand can achieve dexterous manipulation by leveraging the environment, known as “Extrinsic Dexterity” [18]. For example, a simple gripper can rotate an object by pushing it against the table [11], or lifting an object by sliding it along a vertical surface [45]. With the exploitation of external resources such as contact surfaces or gravity, even simple grippers can perform skillful maneuvers that are typically studied with a multi-fingered dexterous hand. Different from a common practice of considering the robot and an object of interest in isolation, extrinsic dexterity focuses on a holistic view of considering the interactions among the robot, the object, and the external environment.

Previous work in extrinsic dexterity has demonstrated a variety of tasks such as in-hand reorientation with a simple gripper, prehensile pushing, or shared grasping [11, 18, 45]. However, the underlying approaches come with several limitations such as relying on hand-designed motions or primitives with limited capability of generalizing across different objects, making assumptions about contact locations and contact modes, or requiring specific gripper design [11, 12, 15, 16, 18, 44, 45]. Instead, we

4. Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity

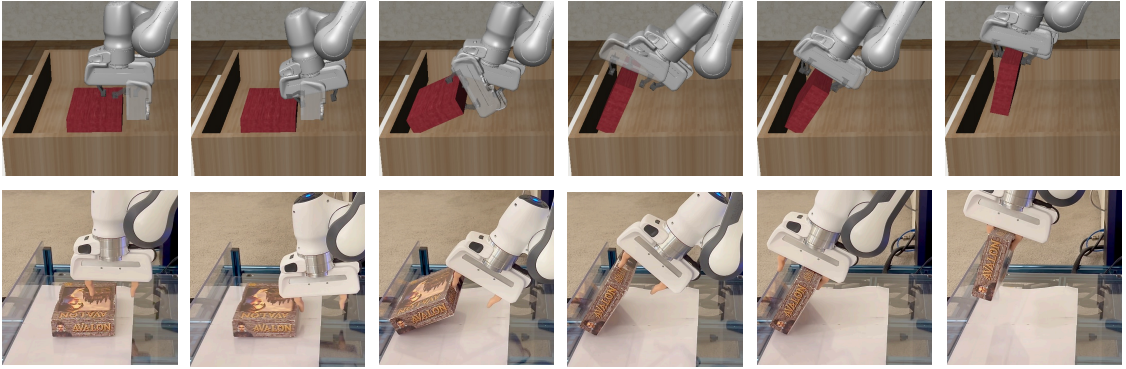


Figure 4.1: We present a system for the “Ocluded Grasping” task with extrinsic dexterity. The goal of this task is to reach an occluded grasp configuration (indicated by a transparent gripper). The figure shows an example of the emergent behavior of the policy and successful sim2real transfer.

build a system with reinforcement learning (RL) to remove these limitations. With RL, the agent can learn a closed-loop policy of how the robot should interact with the object and the environment to solve the task, taking into account both planning and control. In addition, when trained with domain randomization, the policy can learn to be robust to different variations of physics. These properties of RL can enable extrinsic dexterity in a more general setting.

In this work, we study “Ocluded Grasping” as an example of a task that requires extrinsic dexterity. The goal of this task is to grasp an object in poses that are initially occluded. Consider, for example, a robot that needs to grasp a cereal box lying on its side on a table; the desired grasp is not reachable because it is partially occluded by the table (Figure 4.1). To achieve this grasp with a parallel gripper, the robot might rotate the object by pushing it against a vertical wall to expose the desired grasp and then reach it. This task is in contrast with common grasping tasks which focus on reaching an unoccluded grasp in free space with a static or near-static scene [83, 86, 128].

The goal of this work is to build a system for the “Ocluded Grasping” task as an example of the combination of RL and extrinsic dexterity that works on a physical robot. We investigate design choices of such a system and emphasize the simplicity of the method. With model-free RL, we design a reward function that optimizes pre-grasp and grasping motion without the separation of stages as previous work

in pre-grasp [10, 36, 118]. By placing the object in the bin and using a compliant low-level controller, the agent shows emergent extrinsic dexterity behavior without additional reward terms. We also incorporate a set of desired grasps with a training curriculum and a grasp selection procedure during evaluation. We improve the policy with Automatic Domain Randomization [91] over physical parameters which robustify the contact-rich behaviors across noise and environment variations. In the experiments, we provide a comprehensive evaluation of the system in simulation to analyze the importance of each component. The policy is zero-shot transferred to the physical robot and successfully executes similar behaviors to complete the task. The policy achieves a success rate of 78% and shows generalization across various out-of-distribution objects. Our main contribution is the real robot experiment results. Existing work with a simple hand has not shown such behaviors on the real robot with a similar level of complexity in contact events and generalization across objects at the same time.

4.2 Related Work

4.2.1 Extrinsic dexterity

“Extrinsic dexterity” is a type of manipulation skill that enhances the intrinsic capability of a robot hand using external resources including external contacts, gravity, or dynamic motions of the arm [18]. Previous work in extrinsic dexterity has demonstrated complex tasks with a simple gripper including in-hand reorientation [18, 44], prehensile pushing [11, 12], shared grasping [45], and more. Their methods are based on hand-crafted trajectories [18], task-specific motion primitives [23, 44, 45], or planning over contact modes [11, 12, 15, 16] to simplify the problem. They rely on careful assumptions on contacts such as assuming a fixed number of sticking contacts between the fingertips and the object. In this work, we take an alternative approach to use RL to learn a closed-loop policy that considers both planning and control without limitations on contact events. The resulting policy shows more versatile contact switches beyond prior work and can be transferred to a physical robot.

4.2.2 Grasping

Grasping is an important task in robot manipulation and has been studied from various aspects.

Grasp generation: One area of study in grasping is to generate stable grasps using analytical approaches [90, 110] or learning approaches [9, 83, 85, 86, 96]. In this work, we assume that the desired grasp configurations are given which may come from any grasp generation method.

Grasp execution: To execute a grasp following grasp generation, a motion planner is usually used to generate a collision-free path towards the desired grasp configuration [27, 127, 128, 129]. All of these works aim at achieving the unoccluded grasp configurations in static or near-static scenes. Instead, our work focuses on a complementary direction of achieving occluded grasp locations by interacting with the object of interest. Another line of work in grasping uses an end-to-end pipeline without the separation of grasp generation and grasp execution [54, 117]. However, they do not demonstrate performing the occluded grasps studied in this work.

Pre-Grasp manipulation: To deal with occluded grasps, prior work has studied pre-grasps as a preparatory stage of the grasping task. Typical motions for pre-grasps include rotation through planar pushing [10], sliding the object to the edge of the table [36, 58], or rotate the object against the wall [118]. Sun et al. [118] is the most related to our work, but they use a specially designed end-effector to perform the pre-grasp motion and then use a second gripper to grasp. We demonstrate that the full grasping task can be solved with a single gripper without special requirements on the end-effectors. These previous work typically separates pre-grasp motion and grasp execution into two stages and impose restrictions on the transitions of the stages. Instead, we co-optimize pre-grasp and grasp execution throughout the episode without explicit separation of the stages.

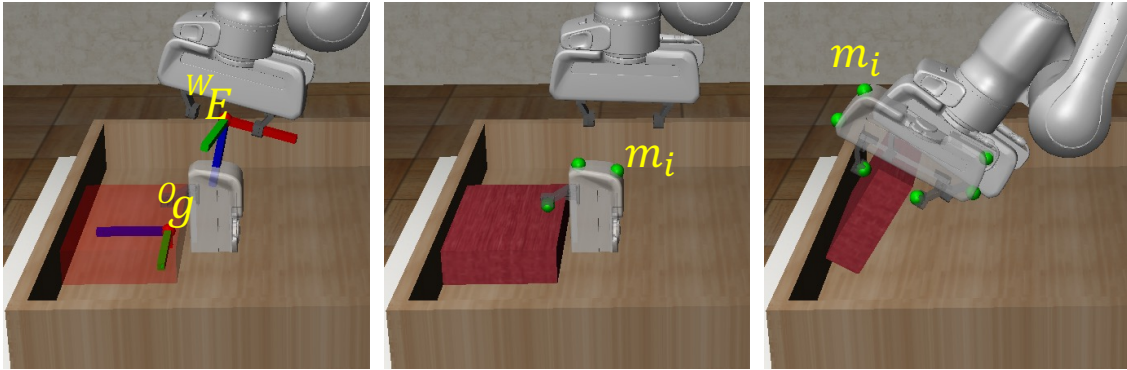


Figure 4.2: Notations: ${}^W E$ denotes the pose of the end-effector in the world frame W . ${}^O g$ denotes the target grasp in the object frame O . Six marker locations m_i in green on the target grasp are used to calculate the occlusion penalty.

4.2.3 Reinforcement Learning for Manipulation

Previous work in RL for manipulation usually treats the object and the robot in isolation from the environment without considering extrinsic dexterity. RL has been applied to dexterous manipulation with a multi-fingered hand and shows contact-rich behaviors [13, 87, 91]. In contrast, with a parallel gripper, prior work focuses on tasks with limited contacts and object motions without utilizing the environment [66, 124]. This is the first work that demonstrates extrinsic dexterity with a simple parallel gripper using RL.

4.3 Task Definition: Occluded Grasping

The goal of a common grasp execution task is to move the end-effector E close to a given desired grasp pose g . The desired grasp might come from any grasp generation method [83, 85, 86] as the input to the grasp execution task. As shown in Figure 4.2, we define an **“Occluded Grasping”** task to be a subset of the grasp execution tasks where the input desired grasp g is initially occluded. To clarify, the term “occluded” in this work is more than visual occlusion. It means the desired grasp intersects with the table and moving the gripper in free space cannot solve this task. The robot has to interact with the object to make the grasp pose reachable. The grasp ${}^O g$ is defined in the object frame O and moves with the object. Formally, the grasp execution

is defined to be successful if the position difference $\Delta T(g, E)$ and the orientation difference $\Delta\theta(g, E)$ are less than the pre-defined thresholds ϵ_T and ϵ_θ respectively at the end of an episode. After successfully reaching a desired grasp pose, the gripper will be closed to complete the grasp. In addition, when the input to the system is a set of grasps $G = \{g_i\}_{i=1}^k$ instead of a single grasp, the agent may select any of the grasp to approach to.

4.4 Learning dexterous grasping with Reinforcement Learning

We build a system that learns a closed-loop policy for the occluded grasping task defined above with model-free RL. In this section, we will discuss important design choices of the system including the design of the RL problem, the extrinsic environment, and the choice of low-level controller. Then we will discuss how to deal with a set of grasps by training with a grasp curriculum and selecting the best grasp during evaluation. We also include Automatic Domain Randomization [91] to improving the generalization of the policy across environment variations.

4.4.1 Preliminaries: Goal-conditioned Reinforcement Learning

We define a Markov Decision Process (MDP) with states $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$, reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and discount factor γ . The state space, action space and the reward function for our task will be discussed in detail in the next section. The goal is to find a policy $\pi(a_t|s_t)$ that maximizes the return $R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$. A Q-function is defined to be the expected return of the policy $Q^\pi(s, a) = \mathbb{E}_\pi[R_t|s_t, a_t]$. In goal-conditioned RL, we define a set of goals $\eta \in \mathcal{G}$ correspond to the reward function $r_\eta(s_t, a_t)$ [107]. To train a policy with a set of goals, both the policy and the Q-function will now take the goal η as input, given by $\pi(a_t|s_t, \eta)$ and $Q^\pi(s_t, a_t, \eta)$. In the occluded grasping task, we use the desired grasps as goals.

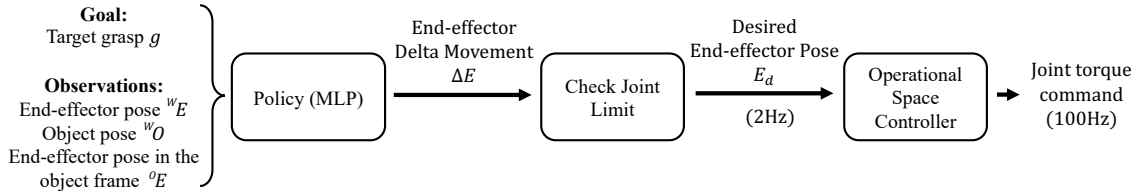


Figure 4.3: Outline of policy execution: Given the observations, the policy outputs an end-effector delta movement (Section 4.4.2) to the low-level controller (Section 4.4.4).

4.4.2 RL Problem Design

Observation and Action Space: The observation that is input to the policy includes a target grasp configuration in the object frame ${}^O g$, the pose of the end-effector in the world frame ${}^W E$ and the object pose in the world frame ${}^W O$. Note that the policy only takes one grasp ${}^O g$ as input but we will discuss how to deal with a set of grasps in Section 4.4.5. For real robot experiments, we use Iterative Closest Point (ICP) for pose estimation of the object which matches a template point cloud of the object to the current point cloud [104]. The action space of the policy is the delta pose of the end-effector ΔE in its local frame which is passed into a low-level controller (Section 4.4.4). An outline of the policy execution is shown in Figure 4.3. More details can be found in Appendix C.2.

Reward: We design the reward function to optimize the pre-grasp motion and grasp execution without separating them into two stages as in previous work [10, 36, 58, 118]:

$$r = \alpha D(g, E) + \beta \sum_i P(m_i) \quad (4.1)$$

$$D(g, E) = \alpha_1 \Delta T(g, E) + \alpha_2 \Delta \theta(g, E) \quad (4.2)$$

where α_1 , α_2 and β are the weights for the reward terms. The first term of Equation 4.1, $D(g, {}^O E)$, is the pose difference between the target grasp and the current end-effector pose, which is to optimize for reaching the grasp. This term is expanded in Equation 4.2 to include the translational and rotational distance, as described in Section 4.3. The second term of Equation 4.1 is the target grasp occlusion penalty which is to penalize the agent if the target grasp configuration is in collision with

the table. This corresponds to a pre-grasp objective. To measure how much the target grasp is occluded by the table, we set six marker points on the target gripper (Figure 4.2) denoted as m_i and compare the height of the markers with the table top. If a marker is below the table top, the height difference will be used as the penalty. Including this occlusion penalty can effectively reduce the local optima where the gripper will reach close to the target grasp (which is occluded) without trying to move the object. Note that we did not impose any reward terms that are explicitly related to extrinsic dexterity. In our system, the use of extrinsic dexterity is an emergent behavior of policy optimization given our objective and environmental setup.

4.4.3 Extrinsic Environment

To exploit the benefits of extrinsic dexterity from object-scene interaction in this task, we construct the scene as having an object in a bin, instead of leaving the object on the table as shown in Figure 4.2. We also make the workspace of the robot large enough such that the robot can move the object to make contacts with the walls (during which the robot itself may also make contact with the wall). In the experiments, we will show that the policy will learn to utilize the wall to rotate the object. Without the wall, it is not able to find a strategy that can perform the task with the parallel gripper.

4.4.4 Choice of Low-level Controller

The choice of low-level controller is important for this task due to the fact that we expect the agent to use extrinsic dexterity which involves rich contacts among the gripper, the object and the bin. We choose Operation Space Control (OSC) as the lower-level controller to execute the policy output which operates at a higher frequency (100Hz) than the RL policy (2Hz) [55] (Figure 4.3). Given a desired pose of the end-effector, OSC first calculates the corresponding force and torque at the end-effector to minimize the pose error according to a PD controller with gain K_p and K_d . Then, the desired force and torque of the end-effector will be converted into desired joint torques according to the model of the robot. We choose relatively low gains so that the controller becomes compliant in the end-effector space. There are two benefits of a compliant OSC in such a contact-rich manipulation task with extrinsic

dexterity. First, being compliant in end-effector space allows safe execution of the motions without smashing the gripper on the objects or the bin. Limiting the delta pose and selecting proper gains K_p , K_d will limit the final force and torque output of the end-effector. If we use a controller that is compliant in the joint configuration space instead, we will not have direct control over the maximum force the end-effector might have on the object and the bin. Second, as shown in Martín-Martín et al. [75], using OSC as the low-level controller might speed up RL training and improve sim2real transfer for contact-rich manipulation.

4.4.5 Multi-grasp Training and Grasp Selection

In this section, we consider the scenario in which a set of desired grasp configurations are given instead of just a single one. During training, given a set of grasps G_{train} , we aim at covering as many grasp configurations as possible. As we will show in the experiments, reaching different grasps might require a significantly different behavior. Learning directly over a diverse set of goals might create difficulties for policy learning [32, 140]. We use an automatic curriculum following OpenAI et al. [91] to gradually expand the set of grasps to be trained with. We start the training with just a single fixed grasp; after the policy has achieved a success rate larger than a threshold, it will be trained on a slightly larger set with grasps close to the initial grasp location.

During testing, if a set of grasps G_{test} is provided, we can select the best grasp within the set to improve the performance of the grasping task, following previous work in integrated grasp and motion planning [27, 127, 128]. With value-based model-free RL algorithms such as Soft Actor Critic [34], the policy is trained together with a Q-function (defined in Section 4.4.1). We propose to select the grasp that maximizes the learned Q-function for the given observation and action: $g^* = \arg \max_{g \sim G_{test}} Q(s_t, a_t, g)$. The selection can be performed for each timestep t , or at the beginning of an episode when $t = t_0$. We include both implementations in the experiments. This can select the grasp that is most easily reached which depends both on the environmental configuration as well as how well the policy has learned to achieve different grasp configurations.

4.4.6 Improving Policy Generalization

To improve generalization across environment variations, we train the policy with Automatic Domain Randomization (ADR) [91]. Similar to the multi-grasp curriculum, the policy is first trained on a single environment with a single object, and gradually expands the range of randomization automatically according to its performance. This significantly reduces the effort of tuning the range of randomization. We randomize different variations of the environment properties such as object size, density, and friction coefficient. We also randomize the parameters of the controller. More descriptions on the ADR procedure can be found in Appendix C.3.

4.5 Experiments

We build the simulation environment for this task using Robosuite [153] and the MuJoCo simulator [125] as shown in Figures 4.1 and 4.2. The environment contains a Franka Emika Panda robot with a parallel gripper and an object in the bin in front of it. We focus on grasping large flat objects from the side since they cannot be grasped with a top-down motion. The policy is trained in simulation with Soft Actor Critic [34]. In this section, we include the results in simulation to discuss each component of the proposed system. We then evaluate zero-shot sim2real transfer on a physical Panda robot across different objects. Implementation details can be found in Appendix C.2.

4.5.1 Training Curves and Ablations

We first evaluate our method by training the policies with a single desired grasp in the default environment without ADR. Figure 4.4(left) shows the training curve of the proposed method and the ablations. The policy trained with the complete system can reach a success rate of 100% before 4000 episodes which corresponds to 160000 environment steps. To evaluate the importance of extrinsic dexterity, we remove the walls of the bin. The resulting policies have 0% success rate and push the object outside of the table. For ablations on the reward function, we remove the occlusion penalty (the second term of Equation 4.1) and also try a sparse reward. Without the

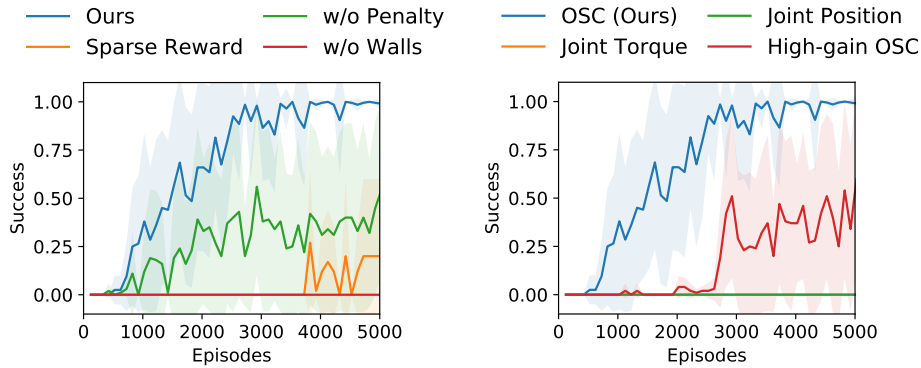


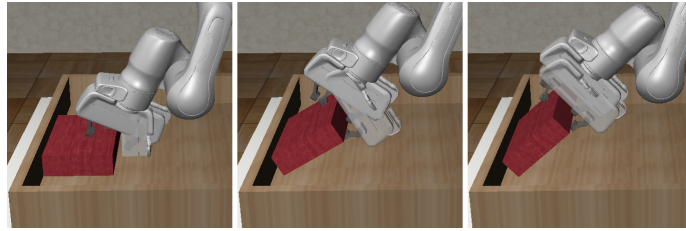
Figure 4.4: Training curves and ablations: Left: ablations on the reward function and the wall. Right: ablations on the controller.

occlusion penalty, the policy is more likely to get stuck at a local optima (an example shown in Figure 4.5a) and thus the success rate becomes lower. With the alternative of a $\{-1, 0\}$ sparse reward, the policy learns much slower. We also experiment with different low-level controllers. Both joint torque and joint position control lead to worse performance which indicates the importance of using end-effector coordinates. With a less compliant controller by increasing the gains of OSC, the success rate becomes lower which demonstrates the importance of compliance for contact-rich tasks in addition to the safety considerations.

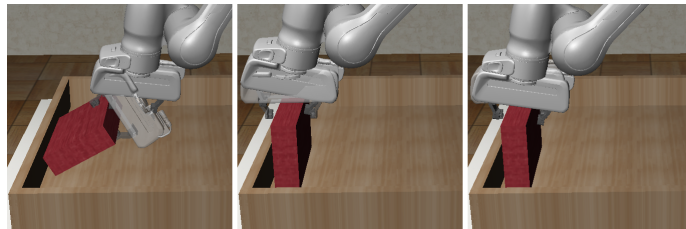
4.5.2 Emergent Behaviors

Figure 4.1 shows a typical strategy of a successful policy which involves multiple stages of contact switches. The gripper first moves close to the object and makes contact on the side of the object with the top finger. It then pushes the object against the wall to rotate it. During this stage, the gripper usually maintains a fixed or rolling contact with the object, but sliding also occurs. The object might have sliding or sticking contacts with the wall and the ground. After the gripper has rotated a bit further and the bottom fingertip is below the object, the gripper will let the object drop on the bottom finger. After that, the gripper will try to match the desired pose more precisely. At this point, the policy has executed the grasp successfully and it is ready to close the gripper. This type of learned contact-rich behaviors with a

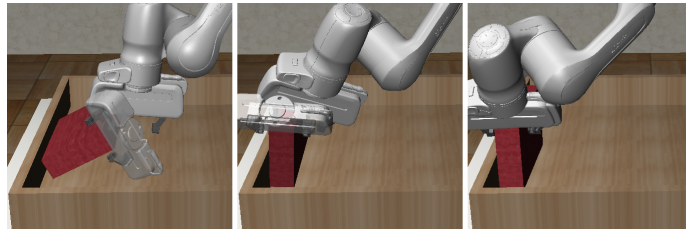
4. Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity



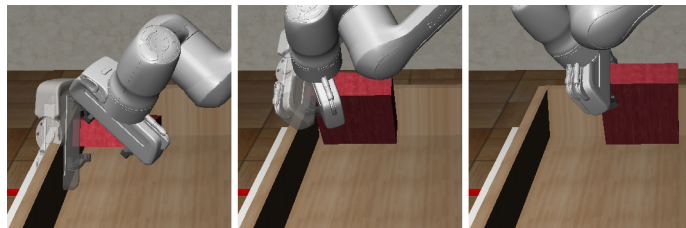
(a) **Local optima:** An example of local optima where the gripper uses the bottom finger to lift the object (instead of the top) and then fails to move the object between its two fingers to prepare for the grasp.



(b) **Standing object:** One of the successful strategies is to flip the object until it stands on the side and then reach the grasp.



(c) **MultiGrasp-Front:** When the desired grasp is at the corner, the policy flips the object by pushing it on the side and then move close to the grasp.



(d) **MultiGrasp-Side:** When the grasp is on the side, the policy can use another side of the wall to rotate the object and reach the desired grasp.

Figure 4.5: Visualizations of the policies in different scenarios.

simple gripper has not been shown in previous work. In Section 4.5.5, we will further demonstrate that it can be transferred to a physical robot.

One of the key decisions in this strategy is to use the top finger to rotate the object instead of the bottom finger. One might suppose an alternative approach which is to use the bottom finger to scoop the object against the wall and then directly roll the finger underneath the object to reach the grasp. However, this strategy is not physically feasible on the parallel gripper due to the limited degree of freedom of the finger. We observe that the policies that follow this strategy during exploration usually get stuck at a local optima without successfully reaching the grasp (Figure 4.5a). Another successful strategy is to flip the object to stand on its side and then move to the grasp (Figure 4.5b). This strategy relies on the fact that the object remains stable after the rotation. We will show in the real-robot experiments that for a non-box object, the object may lie on the wall to maintain stability.

4.5.3 Multi-grasp Experiments

Multi-grasp Training: Going beyond a single desired grasp, we generate the grasp configurations around the side of the object and parameterized the grasps into a continuous grasp ID in the range of $[0, 4]$ (Figure 4.6). We train two types of multi-grasp policies with curriculum: *MultiGrasp-Front* which starts from grasp ID=1.5 and *MultiGrasp-Side* which starts from grasp ID=2.5. As a baseline, we train a policy by uniformly sampling from the entire set of grasps without curriculum (*No curriculum*). Figure 4.6 shows the performance of these policies evaluated across all grasp IDs. Without curriculum, the agent has difficulties in reaching any of the grasps. With the automatic curriculum, both *MultiGrasp-Front* and *MultiGrasp-Side* expand from a single grasp to most of the grasps on one side of the object. Figures 4.5c and 4.5d include qualitative examples of the behaviors which shows that it may require a completely different behavior for different grasps.

Grasp Selection: We compare grasp selection methods with *MultiGrasp-Front* and *MultiGrasp-Side*. We sample 50 grasps from the training range of the policy at the beginning of each episode. The grasp selection methods will choose a grasp from this set as the input to the policy. We evaluate the following grasp selection options:

4. Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity

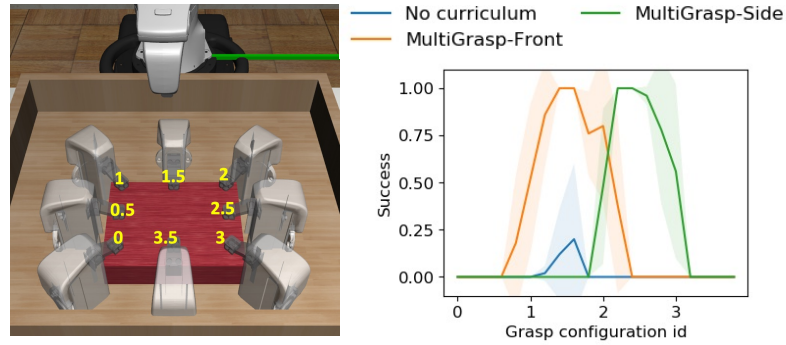


Figure 4.6: **Left:** Grasp configurations. **Right:** MultiGrasp Training results with and without curriculum.

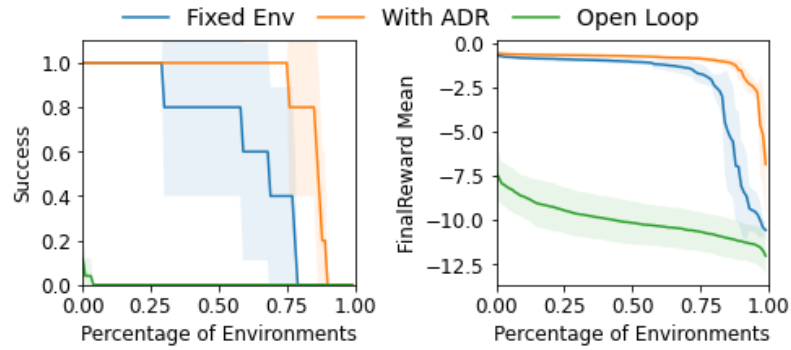


Figure 4.7: Evaluation on the generalization of the policies by sampling 100 environments.

Table 4.1: Comparison of grasp selection methods: Side grasp policies achieve better performance when using the Q-function to select the grasp.

	MultiGrasp Front	MultiGrasp Side
ArgmaxQ	1.00 ± 0.00	1.00 ± 0.00
ArgmaxQ- t_0	1.00 ± 0.00	1.00 ± 0.00
PoseDiff	1.00 ± 0.00	0.96 ± 0.08
PoseDiff- t_0	1.00 ± 0.00	0.50 ± 0.43
Uniform	0.54 ± 0.16	0.90 ± 0.06

ArgmaxQ selects the grasp that corresponds to the highest Q-value. *PoseDiff* selects the grasp according to the closest distance to the current gripper pose according to Equation 4.2 (with the same weights as the reward function). Both *ArgmaxQ* and *ArgmaxQ* select a grasp for each timestep. Alternatively, *ArgmaxQ-t₀* and *PoseDiff-t₀* only selects a grasp during the first timestep of the episode. *Uniform* samples a grasp from the set uniformly. The results are summarized in Table 4.1. For *MultiGrasp-Side*, using the Q-function for grasp selection is better than the other approaches. Since the policy has a more complicated maneuver to reach the side (Figure 4.5d), the Q-function can capture the difficulty of the goal better than pose difference.

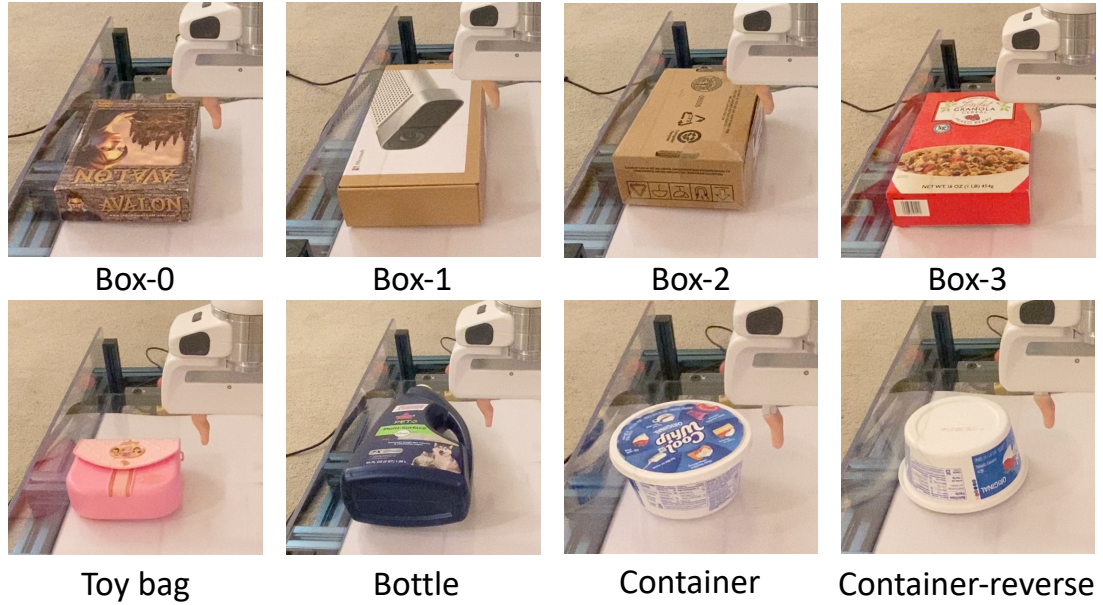
4.5.4 Policy Generalization

In this section, we evaluate the generalization of the policy across environment variations: open loop trajectories (*Open Loop*), policies trained over a fixed environment (*Fixed Env*) and policies trained with ADR (*With ADR*). The open loop trajectories are obtained by rolling out the *Fixed Env* policies in the default environment. We sample 100 environments from the range covered by the ADR policies (Appendix C.3) and plot the percentage of environments that are above a certain performance metric (Figure 4.7). The closed-loop policies are much better than open-loop trajectories. With ADR, the generalization can be improved even further. Sensitivity analysis on single physical parameters can be found in Appendix C.1.

4.5.5 Real-robot experiments

We execute the single grasp policies on the real robot with zero-shot sim2real transfer over 10 test cases with different dimensions, densities, surface frictions, and sizes as shown in Figure 4.8. For non-box objects, the poses are defined with respect to the bounding boxes. The bounding boxes are obtained by running Principle Component Analysis (PCA) on the scanned object point cloud. More details of the real robot experiments can be found in Appendix C.4. Note that most of the objects are out-of-distribution. We evaluate 10 episodes for each test case and summarize the results in Figure 4.8. The success is measured by being able to close the gripper and lift the object at the end of the episode. We first compare the policies with and without Automatic Domain Randomization, denoted as **w ADR** and **w/o ADR**

4. Learning to Grasp the Ungraspable with Emergent Extrinsic Dexterity



Object-ID	Success w/o ADR	Success w/ ADR	Success Init-pose
Box-0 (128g)	9/10	9/10	7/10
Box-0 (237g)	6/10	10/10	7/10
Box-0 (345g)	3/10	4/10	5/10
Box-1	5/10	8/10	8/10
Box-2	2/10	9/10	9/10
Box-3	0/10	7/10	9/10
Toy Bag	7/10	7/10	9/10
Bottle	0/10	8/10	1/10
Container	0/10	10/10	1/10
Container-rev	0/10	6/10	0/10
Average	33%	78%	56%

Figure 4.8: We evaluate the policy on the real robot with various test objects. The policy trained in simulation on box-shape objects can generalize to the real robot and other shapes. With ADR, the policy achieves 45% better success rate.

respectively. Quantitatively, the policy with ADR achieves a success rate of 78% while the policy without ADR achieves 33%. Interestingly, the policy with ADR achieves 24/30 successes over the bottle, the Cool Whip container, and the container with a reversed initial pose. This demonstrates that although the policy is only trained with boxes in simulation, it can also generalize to other shapes to some extent when we represent the object with its bounding box. However, when the object with an out-of-distribution shape has a very different transition dynamics, the policy could fail. Qualitatively, both policies being evaluated exhibit similar strategies as discussed in Section 4.5.2. In fact, a single policy network may execute either the dropping strategy (Figure 4.1) or the standing strategy (Figure 4.5b) depending on the current state. We also include additional results when the initial object location is not close to the wall, denoted as **Init-pose** in Figure 4.8. We finetune the **w/ADR** policy to expand further over the range of initial object locations. The success rate remains similar for most objects, but this setting becomes more challenging for non-box shapes. Videos of the full real robot evaluation including failure cases and recovery behaviors can be found on the website ¹. These real robot results are valuable to the field of manipulation because it is beyond what has been shown with a simple hand considering the combined complexity of contact events, object motion and object generalization.

4.6 Limitations

One limitation of this work is that the policy is trained with box-shape objects. Although it may generalize to other shapes to some extent as shown in the experiments, the policy might be improved by including other shapes during training. In addition, the pose of the object alone may not be sufficient to generalize to novel objects; using a better representation of the shape such as a point cloud or key-points could improve generalization across shapes. However, these changes would also increase the training complexity. Another limitation is that we assume a reasonably accurate robot and gripper model, in terms of geometries, kinematic and dynamic parameters. It would be interesting to explore how to extend the method to transfer across robots and grippers.

¹<https://sites.google.com/view/grasp-ungraspable>

4.7 Conclusion

In this work, we study the “Occluded Grasping” task where the robot with a parallel gripper aims to reach a grasp configuration using extrinsic dexterity. We present a system that learns a closed-loop policy for this task with reinforcement learning. In the experiments, we demonstrate the importance of each component of the system. We also show that the policies can be executed on the real robot and generalize to various objects. One potential extension of our work is to train the policy with a wide variety of object shapes which may require image-based or point cloud-based policies. Also, the pipeline can potentially be extended to other extrinsic dexterity tasks.

Despite the simplicity of the proposed method, we would like to emphasize the following takeaways from this work: First, we provide a concrete example that a simple gripper can do much more than pick-and-place while being cheaper and easier to maintain than a dexterous hand, following previous work in extrinsic dexterity. We envision more future work in this direction in manipulation. Second, RL can be a good option to generate policies with emergent extrinsic dexterity, and sim2real transfer works reasonably well with our proposed system. Our work takes a step towards deploying contact-rich policies with a simple gripper in the real world.

Chapter 5

HACMan: Learning Hybrid Actor-Critic Maps for 6D Non-Prehensile Manipulation

5.1 Introduction

The ability to manipulate objects in ways beyond grasping is a critical aspect of human dexterity. Non-prehensile manipulation, such as pushing, flipping, toppling, and sliding objects, is essential for a wide variety of tasks where objects are difficult to grasp or where workspaces are cluttered or confined. However, non-prehensile manipulation remains challenging for robots; previous work has only shown results with limited object generalization [16, 43] or limited motion complexity, such as planar pushing or manipulating articulated objects with limited degrees of freedom [81, 136, 138]. We propose a method that generalizes across object geometries while showing versatile interactions for complex non-prehensile manipulation tasks.

We present **Hybrid Actor-Critic Maps for Manipulation** (HACMan), a reinforcement learning (RL) approach for non-prehensile manipulation from point cloud observations. The first technical contribution of HACMan is to propose an **object-centric action representation** that is **temporally-abstracted** and **spatially-grounded**. The agent selects a contact location and a set of motion parameters determining the

5. HACMan: Learning Hybrid Actor-Critic Maps for 6D Non-Prehensile Manipulation

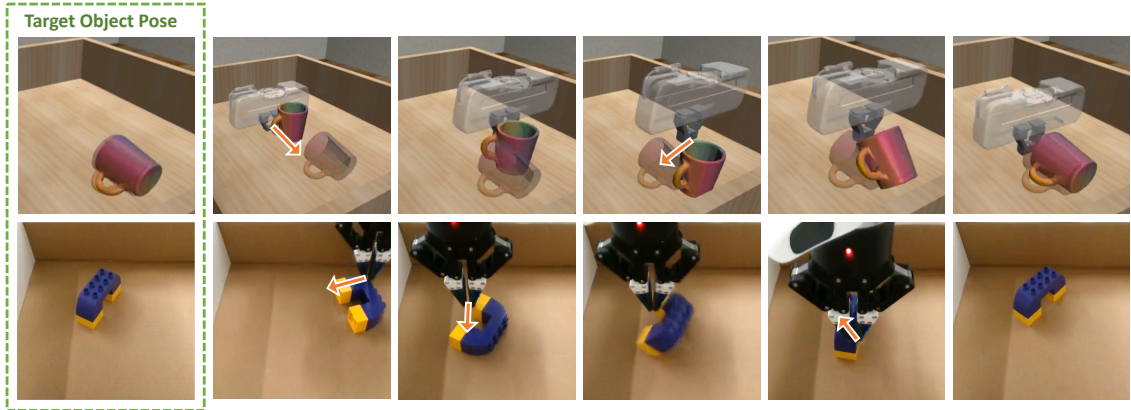


Figure 5.1: We propose HACMan (**H**ybrid **A**ctor-**C**ritic **M**aps for **M**anipulation), which allows non-prehensile manipulation of unseen objects into arbitrary stable poses. With HACMan, the robot learns to push, tilt, and flip the object to reach the target pose, which is shown in the first column and in the top row with transparency. The policy allows for dynamic object motions with complex contact events in both simulation (top) and in the real world (bottom). The performance of the policy is best understood from the videos on the website: <https://hacman-2023.github.io>.

trajectory it should take after making contact. The contact location is selected from the observed object point cloud which provides spatial grounding. At the same time, the robot decisions become more temporally-abstracted because we focus on only learning the contact-rich portions of the action.

The second technical contribution of HACMan is to incorporate the proposed action representation in an actor-critic RL framework. Since the contact location is defined over a discrete action space (selecting a contact point among the points in the object point cloud) and the motion parameters (defining the trajectory after contact) are defined over a continuous action space, our action representation is in a hybrid discrete-continuous action space. In HACMan, the actor network outputs *per-point* continuous motion parameters and the critic network predicts *per-point* Q-values over the object point cloud. Different from common continuous action space RL algorithms [30, 70], the per-point Q-values are used both to update the actor and also to compute the probability for selecting the contact location. We modify the update rule of an existing off-policy RL algorithm to incorporate such a hybrid action space.

We apply HACMan to a 6D object pose alignment task with randomized initial object poses, randomized 6D goal poses, and diverse object geometries (Fig. 5.1). In simulation, our policy generalizes to unseen objects without a performance drop, obtaining an 89% success rate on unseen objects. In addition, HACMan achieves a training success rate more than three times higher than the best baseline with an alternative action representation. We also perform real robot experiments with zero-shot sim2real transfer, in which the learned policy performs dynamic object interactions over unseen objects of diverse shapes with non-planar goals. Our contributions include:

- We propose a novel object-centric action representation based on 3D spatial action maps to learn complex non-prehensile interactions. We also modify an existing off-policy RL algorithm to incorporate such a hybrid discrete-continuous action space.
- The proposed action representation demonstrates substantive improvements of performance over the baselines and shows strong generalization to unseen objects.
- The learned policy showcases complex contact-rich and dynamic manipulation skills, including pushing, tilting, and flipping, shown both in simulation and with a real robot.

5.2 Related Work

Non-prehensile manipulation. Non-prehensile manipulation is defined as manipulating objects without grasping them [76]. Many non-prehensile manipulation tasks involve complex contact events among the robot, the object, and the environment, which lead to significant challenges in state-estimation, planning and control [16, 43, 82, 138]. Recent work has applied learning-based methods in non-prehensile manipulation, but they are limited in terms of either skill complexity [68, 81, 131, 136] or object generalization [68, 149]. In contrast, our work shows 6D object manipulation involving more complex object interactions while also generalizing to a large variety of unseen object geometries.

Visual Reinforcement Learning with Point Clouds. Recent research has explored various ways of incorporating point clouds into RL [71, 99]. To overcome the optimization difficulties, previous work has tried pre-training the feature extractor with an auxiliary loss [46], initializing the RL policy with behavior cloning [129], or using student-teacher training [13, 14] (see detailed discussion in Appendix D.6). Our method does not require these additional training procedures due to the benefits of the proposed action representation. In the experiments, we show that the baselines following the most relevant previous work [13, 14, 99] struggles when the task becomes more complex.

Spatial action maps. Similar to our method, recent work has explored spatial action maps that are densely coupled with visual input instead of compressing it into a global embedding, based on images [25, 136, 145], point clouds [81, 109, 119], or voxels [111]. Unlike previous works with spatial action maps that consider one-shot decision making (similar to a bandit problem) [81, 131, 136] or rely on expert demonstrations with ground truth actions [109, 111, 145], our method reasons over multi-step sequences with no expert demonstrations. For example, Xu et al. [136] only chooses a single contact location followed by an action sequence, rather than a sequence of contact interactions. Unlike previous work in spatial action maps that uses DQN with discrete actions [25, 47, 131, 144], our hybrid discrete-continuous action space allows the robot to perform actions without discretization. Furthermore, we demonstrate the benefit of spatial action representations when applied to a 6D non-prehensile manipulation task, which is more challenging than the pick-and-place and articulated object manipulation tasks in previous work.

RL with hybrid discrete-continuous action spaces. Most RL algorithms focus on either a discrete action space [79] or a continuous action space [30, 34, 70]. However, certain applications are defined over a hybrid action space where the agent selects a discrete action and a continuous parameter for the action [38, 89, 135]. Unlike previous work, our hybrid action space uses a spatial action representation in which the discrete actions are defined over a map of visual inputs. The closest to our work is Feldman et al. [25] in terms of applying RL to spatial action maps, but they only consider a finite horizon of 2. We include formal definitions of the policies over the hybrid action space and modify the loss functions and exploration accordingly.

5.3 Preliminaries

A Markov Decision Process (MDP) models a sequential stochastic process. An MDP can be represented as a tuple (S, A, P, r) , where S is the state space; A is the action space; $P(s_{t+1}|s_t, a_t)$ is the transition probability function, which models the probability of reaching state s_{t+1} from state s_t by taking action a_t ; $r(s_t, a_t, s_{t+1})$ is the immediate reward at time t . The objective is to maximize the return R_t , defined as the cumulative discounted reward $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$. Given a policy π , the Q-function is defined as $Q^\pi(s, a) = \mathbb{E}_\pi[R_t | s_t = s, a_t = a]$.

HACMan is built on top of Q-learning-based off-policy algorithms with a continuous action space [30, 34, 38]. In these algorithms, we define a deterministic policy π_θ parameterized by θ and a Q-function Q_ϕ parameterized by ϕ . Note that since the policy is deterministic, we use epsilon-greedy during exploration. Given a dataset D with transitions (s_t, a_t, s_{t+1}) , the Q-function loss is defined according to the Bellman residual:

$$L(\phi) = \mathbb{E}_{s_t, a_t, s_{t+1} \sim D} [(Q_\phi(s_t, a_t) - y_t)^2], \quad (5.1)$$

where y_t is defined as:

$$y_t = r_t + \gamma Q_\phi(s_{t+1}, \pi_\theta(s_{t+1})). \quad (5.2)$$

The policy loss for π_θ is defined to maximize the Q-function:

$$J(\theta) = -\mathbb{E}_{s_t \sim D} [Q^{\pi_\theta}(s_t, a_t) |_{a_t = \pi_\theta(s_t)}]. \quad (5.3)$$

5.4 Problem Statement and Assumptions

We focus on the task of 6D object pose alignment with non-prehensile manipulation. The objective of the robot is to perform a sequence of non-prehensile actions (i.e. pushing, flipping) to move an object on the table into a target goal pose. We assume that the goals are stable object poses on the table. The robot policy observes the point cloud of the scene from depth cameras, denoted as \mathcal{X} . We assume that the point cloud observation is segmented between the background and the object to be manipulated. Thus, the full point cloud \mathcal{X} consists of the object point cloud \mathcal{X}^{obj} and the background point cloud \mathcal{X}^b . The feature for each point is a 4-dimensional

vector, including a 1-dimensional segmentation mask and a 3-dimensional goal flow vector (will be defined in Section 5.5.3).

5.5 Method

5.5.1 Action Representation

We propose an object-centric action space that consists of two parts: a **contact location** a^{loc} on an object and **motion parameters** a^m which define how the robot moves to interact with the object after contact. As shown in Fig. 5.2, to execute an action, the end-effector will first move to a location in free space near location a^{loc} , after which it will interact with the object using the motion parameters a^m . After the interaction, the end-effector will move away from the object, a new observation is obtained, and the next action can be taken.

Specifically, given the object point cloud $\mathcal{X}^{obj} = \{x_i \mid i = 1 \dots N\}$, where $x_i \in \mathbb{R}^3$ are the point locations, the contact location a^{loc} is chosen from among the points in \mathcal{X}^{obj} . Thus, a^{loc} is defined over a discrete action space of dimension N . We assume a collision-free motion planner to move the gripper to the contact location a^{loc} (see Appendix D.1.6 for

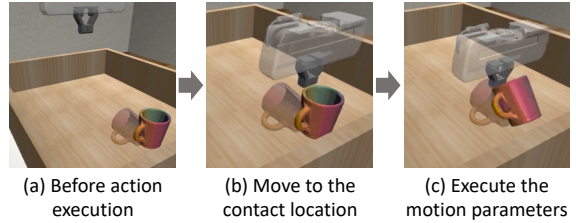


Figure 5.2: **Illustration of our action space.**

details). In contrast, the motion parameters a^m , which define how the gripper interacts with the object after contact, are defined in a continuous action space. Furthermore, we define a^m as the end-effector delta position movement from the contact position, hence $a^m \in \mathbb{R}^3$. Our experiments show that translation-only movements are sufficient to enable complex 6D object manipulation in our task. We also include additional experiments on extending motion parameters to enable rotations in Appendix D.3.4.

The proposed action representation has two benefits compared to previous work. First, it is **temporally-abstracted**. We “abstract” a sequence of lower-level gripper movements of approaching the contact and executing the motion parameters into one action decision step in the RL problem definition. Compared to the common action

space of end-effector delta movements [84, 97, 139, 149], the agent with our action space can avoid wasting time learning how to move in free space and instead focus on learning contact-rich interactions. Second, it is **spatially-grounded** since the agent selects a contact location from the observed object point cloud.

5.5.2 Hybrid RL Algorithm

The proposed action space is a hybrid discrete-continuous action space: the contact location a^{loc} is discrete while the motion parameters a^m are continuous. We propose a way to adapt existing off-policy algorithms designed for continuous action spaces [30, 34] to this hybrid action space. First, consider the simpler case of an action space that only has the continuous motion parameters a^m . In this case, we can directly apply existing off-policy algorithms as described in Section 5.3. Given the observation s (which is the point cloud \mathcal{X} in our task), we can train an actor to output the motion parameters $\pi_\theta(s) = a^m$. Similarly, the critic $Q_\phi(s, a^m)$ outputs the Q-value given the observation s and the motion parameters a^m ; the Q-value can be used to update the actor according to Eqn. 5.3.

To additionally predict the contact location a^{loc} , we also need the policy to select a point among the discrete set of points in the object point cloud \mathcal{X}^{obj} . Our insight is that we can embed such a discrete component of the action space into the critic by training the critic to output a per-point Q-value Q_i for each point x_i over the entire point cloud. The Q-value at each point on the object represents the estimated return after selecting this point as the contact location. These Q-values can thus be used not only to update the actor, but also to select the contact location as the point with the highest Q-value. Additionally, we train the actor to also output per-point motion parameters a_i^m for each point x_i . If point x_i is selected as the contact location a^{loc} , then the motion parameters at this point a_i^m will be used as the gripper motion after contact.

The overall architecture is shown in Fig. 5.3. The actor π_θ receives as input the point cloud observation and outputs per-point motion parameters $\pi_\theta(\mathcal{X}) = \{a_i^m = \pi_{\theta,i}(\mathcal{X}) \mid i = 1 \dots N\}$. We call this per-point output an **“Actor Map”**. The critic also receives as input the point cloud observation. It first calculates per-point features $f(\mathcal{X}) = \{f_i \mid i = 1 \dots N\}$. The critic then concatenates each per-point feature f_i

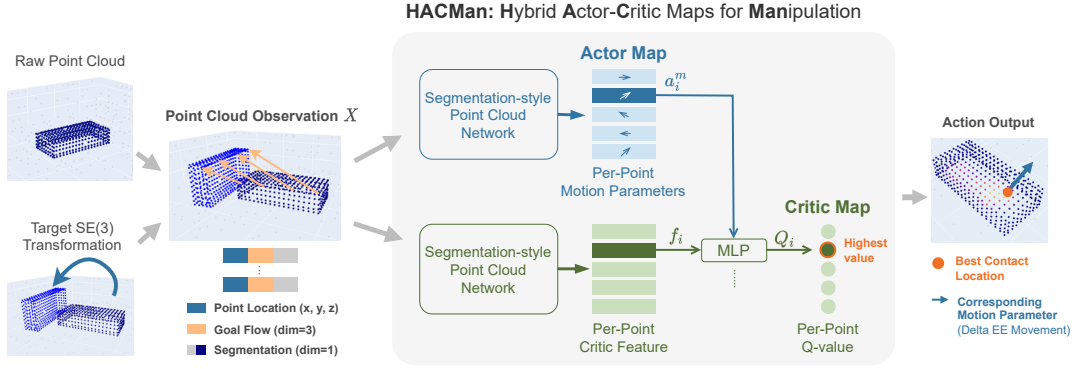


Figure 5.3: **An overview of the proposed method.** The point cloud observation includes the location of the points and point features. The goal is represented as per-point flow of the object points. The actor takes the observation as input and outputs an **Actor Map** of per-point motion parameters. The Actor Map is concatenated with the per-point critic features to generate the **Critic Map** of per-point Q-values. Finally, we choose the best contact location according to the highest value in the Critic Map and find the corresponding motion parameters in the Actor Map.

(Section 5.4) with the corresponding per-point motion parameter a_i^m and inputs the concatenated vector to an MLP. The output of the MLP is a per-point Q-value: $Q_i = Q_\phi(f_i, a_i^m)$, which scores the action of moving the gripper to location x_i and executing motion parameters a_i^m . We call this per-point output a “**Critic Map**”. In this way, the critic is able to reason jointly about the contact location (via the feature f_i) as well as the motion parameters a_i^m . In our implementation, both the actor and the critic use segmentation-style PointNet++ architecture [98] (Appendix D.2).

At inference time, we can select the point x_i within the object points \mathcal{X}^{obj} with the highest Q-value Q_i as the contact location and use the corresponding motion parameters a_i^m . For exploration during training, we define a policy π^{loc} which selects the contact location based on a softmax of the Q-values over all of the object points. The probability of a point x_i being selected as the contact location is thus given as:

$$\pi^{loc}(x_i | s) = \pi^{loc}(x_i | \mathcal{X}) = \frac{\exp(Q_i/\beta)}{\sum_{k=1, \dots, N} \exp(Q_k/\beta)}. \quad (5.4)$$

β is the temperature of the softmax which controls the exploration of the contact location. Note that the background points \mathcal{X}^b are included in the observation

$s = \mathcal{X} = \{\mathcal{X}^b, \mathcal{X}^{obj}\}$, but are excluded when choosing the contact location. We modify the update rules of the off-policy algorithm for this hybrid policy. Given $s = \mathcal{X}$, we first define the per-point loss for updating the actor $\pi_{\theta,i}(s)$ at location x_i according to Eqn. 5.3:

$$J_i(\theta) = -Q_\phi(f_i, a_i^m) = -Q_\phi(f_i, \pi_{\theta,i}(s)). \quad (5.5)$$

f_i is the feature corresponding to point x_i . The total objective of the actor is then computed as an expectation over contact locations:

$$J(\theta) = \mathbb{E}_{x_i \sim \pi^{loc}}[J_i(\theta)] = \sum_i \pi^{loc}(x_i | s) \cdot J_i(\theta). \quad (5.6)$$

$\pi^{loc}(x_i | s)$ is the probability of sampling contact location x_i , defined in Eq. 5.4. The difference between Eqn. 5.6 and the regular actor loss in Eqn. 5.3 is that we use the probability of the discrete action to weight the loss for the continuous action. To take into account π_{loc} during the critic update, the Q-target y_t from Eqn. 5.2 is modified to be:

$$y = r_t + \gamma \mathbb{E}_{x_i \sim \pi^{loc}}[Q_\phi(f_i(s_{t+1}), \pi_{\theta,i}(s_{t+1}))]. \quad (5.7)$$

5.5.3 Representing the Goal as Per-Point Goal Flow

As described in Section 5.4, the objective of our task is to move an object to a given goal pose. Instead of concatenating the goal point cloud to the observed point cloud [13, 14], we represent the goal as per-point ‘‘goal flow’’: Suppose that point x_i in the initial point cloud corresponds to point x'_i in the goal point cloud; then the goal flow is given by $\Delta x_i = x'_i - x_i$. The goal flow Δx_i is a 3D vector which is included as the feature of the point cloud observation (concatenated with a segmentation label, resulting in a 4-dimensional feature vector). This flow representation of goal is also used to calculate the reward and success rate for the pose alignment task (Appendix D.1.3). In the real robot experiments, we estimate the goal flow using point cloud registration (Appendix D.4). Our ablation experiments suggest that utilizing the flow representation of the goal drastically enhances performance compared to directly concatenating the goal point cloud (Appendix D.3.1).

5.6 Experiment Setup

We evaluate our method on the 6D object pose alignment task as described in Section 5.4. The objective is to perform a sequence of non-prehensile actions (i.e. pushing, flipping) to move the object to a given goal pose. In this section, we describe the task setup in simulation, used for training and simulation evaluation (see Section 5.8 for real robot experiments).

Task Setup. The simulation environment is built on top of Robosuite [154] with MuJoCo [125]. We include 44 objects with diverse geometries from Liu et al. [72]. Details and visualizations of the object models are included in Appendix D.1.1. The object dataset is split into three mutually exclusive sets: training set (32 objects), unseen instances (7 objects) and unseen categories (5 objects). The 7 unseen instances consist of objects from categories included in the training set, whereas the 5 objects in “unseen categories” consist of novel object categories. An episode is considered a success if the average distance between the corresponding points of the object and the goal is less than 3 cm. More details on our simulation environment setup can be found in Appendix D.1.

Task Variants. To analyze the limitations of different methods, we design the object pose alignment task with varying levels of difficulty. We consider three types of object datasets: An **All Objects** dataset that includes the full object dataset, a **Cylindrical Objects** dataset consisting of only cylindrical objects, and a **Single Object** dataset consists of just a single cube. We also try different task configurations: In the **Planar goals** experiments, the object starts from a *fixed* initial pose at the center of the bin, and the goal pose is a randomized *planar translation* of the starting pose. In the **6D goals** experiments, both the object initial pose and goal pose are *randomized* $SE(3)$ stable poses, not limited to planar transformations. This task requires $SE(3)$ object movement to achieve the goal which imposes challenges in spatial reasoning. These different task variations are used to show at what level of difficulty each of the baseline methods stop being able to complete the task.

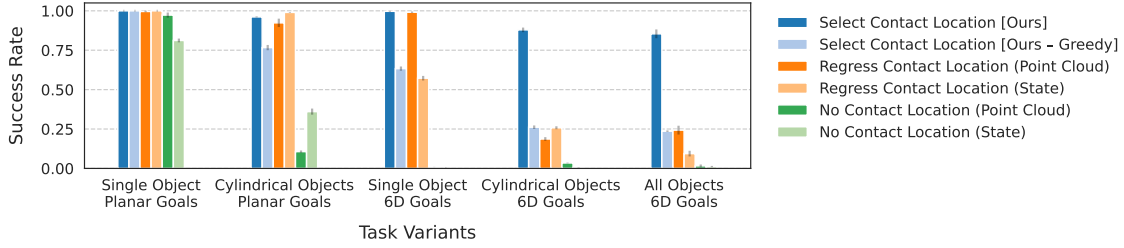


Figure 5.4: **Baselines and Ablations.** Our approach outperforms the baselines and the ablations, with a larger margin for more challenging tasks on the right. Success rates for simple tasks - pushing a single object to an in-plane goal - are high for all methods, but only HACMan achieves high success rates for 6D alignment of diverse objects.

Table 5.1: Features of the proposed action representation compared to the baselines.

	Spatially Grounded?	Temporally Abstracted?
Select Contact Location [Ours]	✓	✓
Regress Contact Location	×	✓
No Contact Location [13, 14, 84, 99, 139, 149]	×	×

5.7 Simulation Results

In this section, we demonstrate the effectiveness of HACMan compared to the baselines and ablations. Fig. 5.4 summarizes the performance of each method after being trained with the same number of environment interactions. The training curves, tables, and additional results can be found in Appendix D.3. Implementation details of all methods are included in Appendix D.2.

Effect of action representations. We compare our method with two alternative action representations, summarized in Table 5.1. In **Regress Contact Location**, the policy directly regresses to a contact location and motion parameters, instead of choosing a contact point from the point cloud as in HACMan. The **No Contact Location** baseline directly regresses to a delta end-effector movement at each timestep. For every action, the robot continues from its position from the previous action, instead of first moving the gripper to a selected contact location. This is the most common action space in manipulation [13, 14, 84, 99, 139, 149]. As input for these two baselines, we use either point cloud observations or ground-truth state, establishing



Figure 5.5: **Qualitative results for the object pose alignment task.** HACMan shows complex non-prehensile behaviors that move the object to the goal pose (shown as the transparent object).

four baselines in total. The baseline that regresses motion parameters from point cloud observations is a common action representation used in prior work in RL from point clouds such as Qin et al. [99]. The baseline that regresses motion parameters from ground-truth state is the most common approach in prior work, such as in Zhou and Held [149] as well as the teacher policies in Chen et al. [13, 14] (see Appendix D.6 for more discussion).

As shown in Fig. 5.4, these baseline action representations struggle with the more complex task variants. For the most challenging task variant “All Objects + 6D goals,” our method achieves a success rate 61% better than the best baseline (see Table D.3 for numbers). As mentioned in Section 2.4, the proposed action representation benefits from being *spatially-grounded* and *temporally-abstracted*. The comparison against the baselines demonstrates the importance of each of these two features (Table 5.1). The “Regress Contact Location” baseline still benefits from being temporally-abstracted because the gripper starts from a location chosen by the policy at each timestep; however, this action representation is not spatially-grounded because it regresses to a contact location which might not be on the object surface, unlike our approach which selects the contact location among the points in the point cloud observation. Thus, the “Regress Contact Location” baseline suffers from training difficulties with more diverse objects (last two variants in Fig. 5.4). The “No Contact Location” baseline is neither spatially-grounded nor temporally-abstracted; it follows the usual approach from prior work [13, 14, 84, 99, 139, 149] of regressing an end-effector delta motion at each timestep. While this is the most common action space in prior work, it has close to zero performance with 6D goals.

Table 5.2: **Generalization to unseen objects.**

Object Set Split	Success Rate	# of Objects
Train	$0.833 \pm .018$	32
Train (Common Categories)	$0.887 \pm .024$	13
Unseen Instance (Common Categories)	$0.891 \pm .033$	7
Unseen Category	$0.827 \pm .047$	5

Effect of Multi-step Reasoning. To test the necessity of multi-step reasoning for the pose alignment task, we experiment with a “**Greedy**” version of HACMan by setting the discount factor γ in the RL algorithm to $\gamma = 0$. This forces the algorithm to optimize for greedy actions for each step. Using RL for multi-step reasoning is one of the important differences between our method and previous work such as Where2Act [81] and UMPNet [136] which optimize for one-step contact locations. Fig. 5.4 indicates that greedy actions might work for planar goals, but suffer from poor performance for 6D goals that requires multi-step non-greedy interactions. For example, the last row in Fig. 5.5 shows an example of our method pushing the object away from the goal position to prepare for flipping it to the correct orientation, demonstrating non-greedy behavior. In contrast, we find that the greedy ablation often results in local optima of only trying to match the object position but not its orientation.

Generalization to unseen objects. The evaluation of our method over unseen objects with 6D goals is summarized in Table 5.2. Our method generalizes well to unseen object instances and unseen categories without a performance drop. When we increase the maximum episode length from 10 steps to 30 steps, our method achieves 95.1% success on unseen categories (Appendix D.3.7). Table 5.2 shows that, comparing the same set of object categories (“common categories”), the success rates of the training instances are similar to the unseen instances. The differences in geometry comparing the training objects and the unseen objects are visualized in Appendix D.1.1.

Goal-conditioned Object Affordance and Multimodality. We visualize the Critic Map, which computes the score of each contact location of the object (Fig. 5.6). The Critic Maps capture goal-conditioned object affordances which describe how the object can be moved to achieve the goal. Fig. 5.6(a) and (b) are two scenarios of performing translation object motions for different object heights: For a thin object,

5. HACMan: Learning Hybrid Actor-Critic Maps for 6D Non-Prehensile Manipulation

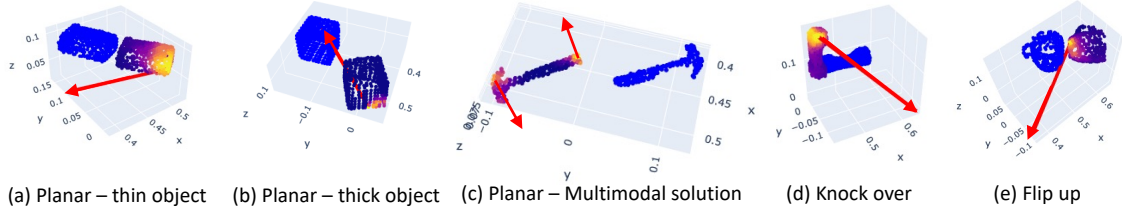


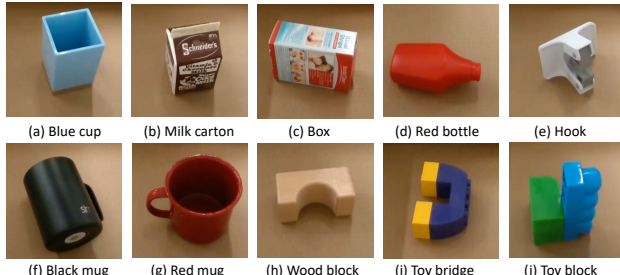
Figure 5.6: **Goal-conditioned Critic Maps.** Blue: goal point cloud. Color map: observed object point cloud. Lighter colors indicate higher Critic Map scores. Red arrows: motion parameters at a selected location. The policy uses different contact locations based on object geometries and goals.

the Critic Map highlights the region of the top of the object (dragging) or from the back side of the object (pushing). For a thick object, it prefers to push from the bottom to avoid the object falling over. In Fig. 5.6(c), the hammer needs to be rotated by 180 degrees. The Critic Map predicts a multimodal solution of pushing from either end of the object. Fig. 5.6(d) and (e) show out-of-plane motions of the object of knocking over and flipping up the objects.

Additional Results. We include additional experiment results in Appendix D.3, including additional ablations, extending the motion parameters, cluttered scenes, longer training steps, longer episode lengths, and success rate breakdown for each object category.

5.8 Real robot experiments

In the real robot experiments, we aim to evaluate the ability of the trained policy to generalize to novel objects and execute dynamic motions in the real world. We evaluate the policy with a diverse set of objects with different shapes, surface frictions, and densities (Fig. 5.7). We use random initial poses and random 6D goals (referred to in the previous section as “All Objects + 6D Goals”). For example, the red mug (Fig. 5.7(g)) has goal poses of being upright on the table or lying on the side. An episode is considered a success if the average distance of corresponding points between the object and the goal is smaller than 3 cm; we also mark an episode as a failure if there is a failure in the point cloud registration between the observation and the goal. Implementation details of the real robot experiments can be found in Appendix D.4.



Object Name	Planar Goals	Non-planar Goals	Total
(a) Blue cup	4/7	4/13	8/20
(b) Milk bottle	6/7	10/13	16/20
(c) Box	2/5	10/15	12/20
(d) Red bottle	4/7	0/13	4/20
(e) Hook	5/8	5/12	10/20
(f) Black mug	4/7	0/13	4/20
(g) Red mug	5/7	3/13	8/20
(h) Wood block	6/7	6/13	12/20
(i) Toy bridge	9/10	5/10	14/20
(j) Toy block	2/2	10/18	12/20
Total	47/67	53/133	100/200
Percentage	70%	40%	50%

Figure 5.7: **Real robot experiments.** HACMan achieves a 50% success rate over unseen objects with different geometries and physical properties, with 6D goal poses.

Fig. 5.7 (right) summarizes the quantitative results of the real robot experiments. We run the evaluations without manual reset, which may create uneven numbers of planar versus out-of-plane goals. It achieves 70% success rate on planar goals and 40% success rate on non-planar goals. Non-planar goals are more difficult than the planar goals because they require dynamic motions to interact with the object. A small error in the action may result in large changes in the object movement. Videos of the real robot experiments can be found on the website: <https://hacman-2023.github.io>. The real robot experiments demonstrate that the policy is able to generalize to novel objects in the real world, despite the sim2real gap of the simulator physics and inaccuracies of point cloud registration for estimating the goal transformation. More discussion can be found in Appendix D.4.

5.9 Limitations

Since the contact location in our action space is defined over the point cloud observation, our method requires relatively accurate depth readings and camera calibration. Further, the contact location is currently limited to the observed part of the object. In addition, for this goal-conditioned task, we represent the goal as per-point flow (Section 5.5.3) which relies on point cloud registration algorithms. Inaccuracies in the registration algorithm sometimes lead to failure cases in real robot experiments. More discussion of the failure cases can be found in Appendix D.4.3. In addition, finetuning the policy on the real robot can potentially improve the success rate further.

5.10 Conclusion

We propose to learn Hybrid Actor-Critic Maps with reinforcement learning for non-prehensile manipulation. The learned policy shows complex object interactions and strong generalization across unseen object categories. Our method achieves a significantly higher success rate than alternative action representations, with a larger performance gap for more difficult task variants. We hope the proposed method and the experimental results can pave the way for future work on more skillful robot manipulation over diverse objects.

One thing to note is that the RL policy also has the potential to discover extrinsic dexterity behaviors in the object pose alignment task, such as pushing the objects against the wall, following the idea from Chapter 4. Currently, the policy does not show such extrinsic dexterity behavior of using the walls mainly because the initial object locations are not close to the walls, and most of the objects we consider in this chapter do not require extrinsic dexterity to be rotated by the gripper. In the future, if we train the policies with a broader range of initial object locations and more diverse objects, the policies may discover strategies involving extrinsic dexterity.

5.11 Extensions to HACMan

5.11.1 HACLeg: Visual Manipulation with Legs

Animals have the ability to use their arms and legs for both locomotion and manipulation. We envision quadruped robots to have the same versatility. We present a system as an extension to HACMan that empowers a quadruped robot to perform object interactions with its legs, drawing inspiration from non-prehensile manipulation, named Hybrid Actor Critic Maps for Leg Manipulation (HACLeg).

The proposed system has two main components: a manipulation module based on HACMan and a locomotion module implemented based on impedance control and Model Predictive Control (MPC). The manipulation module follows HACMan. Given a point cloud observation of the object, the module outputs the best contact location and a set of motion parameters. The locomotion module serves to coordinate leg movement and body pose adjustments to complement the manipulation module.

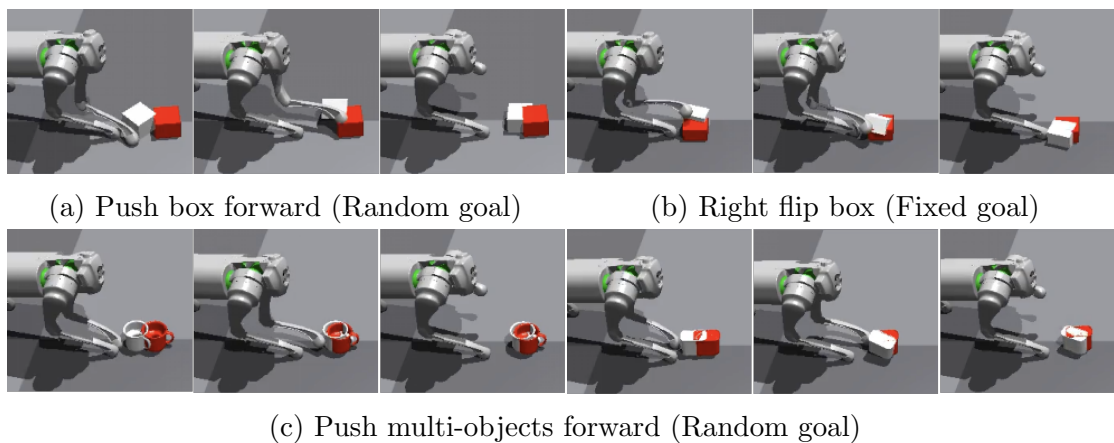


Figure 5.8: HACLeg demonstrates non-prehensile manipulation using leg to align the 6D pose of the object (white) to a given target pose (red).

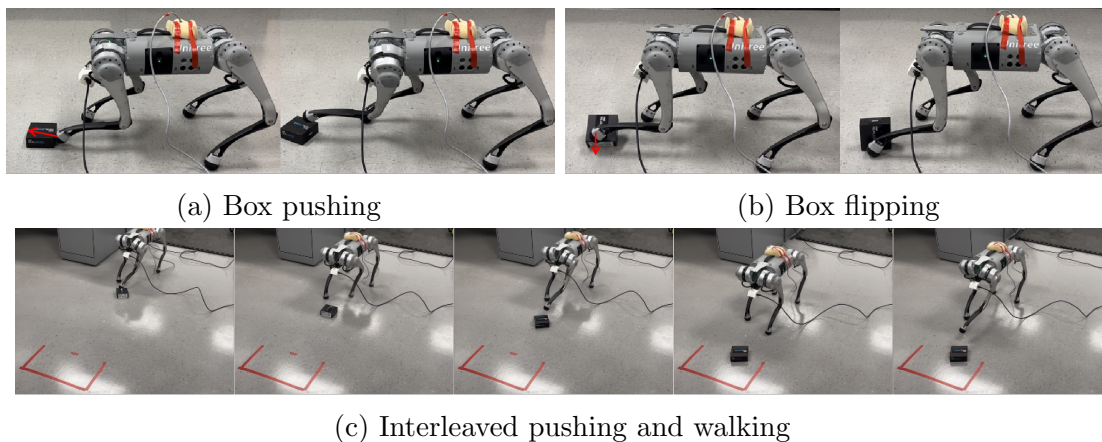


Figure 5.9: In the real robot experiments, HACLeg enables object interactions such as (a) pushing, (b) flipping, or (c) moving the object to a distant goal through repeatedly pushing and walking.

During object interaction, the locomotion module converts the high-level action from HACMan into the low-level robot torque commands. Specifically, we use impedance control to control the leg that interacts with the object, and use MPC to control the other three legs to maintain the stability of the robot. Between manipulation actions, the locomotion module adjusts the robot’s base pose if the object is out of immediate reach. This adjustment enables the robot to effectively push objects toward distant goals, broadening its operational range.

We evaluate the proposed system with the object pose alignment task similar to Section 5.6. In simulation, we experiment with three task variants, including pushing a box forward, flipping a box to the side, and pushing objects with diverse geometries. As shown in Figure 5.8, the policies are able to manipulation the objects to desired goal locations effectively. In the real-world experiment, we evaluate the feasibility of zero-shot sim2real transfer of the manipulation module and the performance of the controller in the locomotion system. As shown in Figure 5.9, the robot is able to perform object pose alignment while maintaining balance using the other three legs. Across 10 episodes, the proposed system achieves a 60% success rate for forward pushing and a 50% success rate for box flipping. HACLeg showcase enhanced manipulation skill and precision beyond previous work related to manipulation with legs [17, 51, 52, 115, 116]. We also demonstrate that the robot is able to push the object to a distant goal by repeatedly pushing the object and walking forward (Figure 5.9c).

5.11.2 HACMan++: A Spatially-grounded Skill Library for Manipulation

In previous sections, we show the effectiveness of HACMan for non-prehensile manipulation. However, the concept of *spatially-grounded* and *temporally-abstracted* action space can be applied to a broader range of manipulation tasks. For example, when the robot needs to grasp an object, it can ground the decision of the grasp location on the object point cloud [120]; when the robot needs to place an object, it can ground the decision of the place location on the table [145]. We extend HACMan to incorporate more manipulation primitives to solve a wider variety of manipulation tasks that may involve both prehensile and non-prehensile motions.

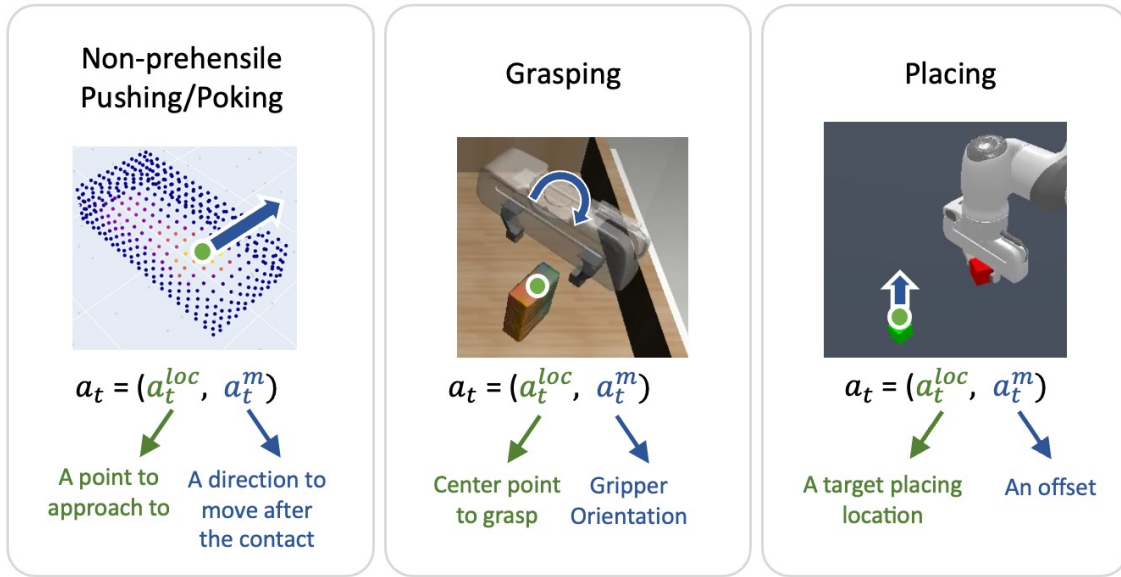


Figure 5.10: **HACMan++ Primitives**. We consider a set of primitives in HACMan++. Each primitive is defined using a grounded location and a vector of motion parameters.

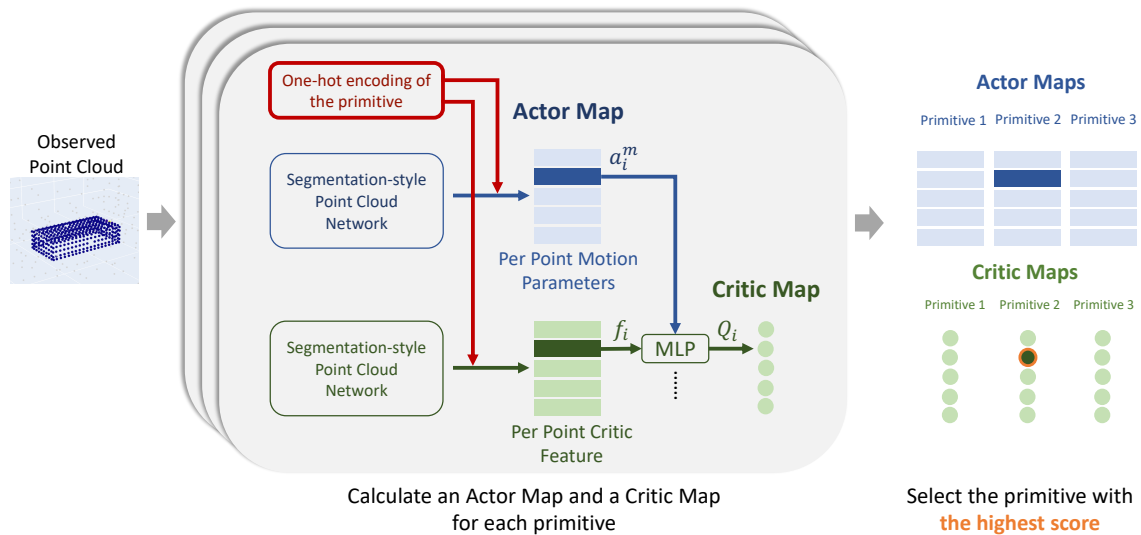


Figure 5.11: **Overview of HACMan++**. In HACMan++, we extend HACMan to incorporate multiple manipulation primitives, such as picking, placing, and poking. We compute the actor and critic maps for each primitives separately. Each primitive is represented as a one-hot encoding as an additional input to the actor and the critic network. The policy output is the type of primitive, a contact location, and a set of motion parameters, selected according to the highest score from the critic maps.

We propose HACMan++, a spatially-grounded skill library for manipulation. Examples of the primitive definitions are shown in Figure 5.10. From Section 2.4, the action space of HACMan consists of a contact location and a set of motion parameters, which describes a non-prehensile poking motion. In HACMan++, we also include a grasping primitive, where the action space consists of a grasp point grounded on the object point cloud, and a gripper orientation which is a continuous parameter. We also define a placing primitive, where the action space consists of a target placing location grounded on the background point cloud and an offset vector which allows more flexibility in the placing location. All of these primitives are defined by a grounded location (discrete) and a vector of motion parameters (continuous), which can fit into the HACMan training framework. Given these primitives, we extend the action space of HACMan to include primitive selection. Based on the current observation, the robot needs to decide which type of primitive to execute, together with the location and motion parameters that define the motion of the primitive.

The modified network architecture is shown in Figure 5.11. We first compute the actor and critic maps for each primitive. We concatenate the one-hot encoding of each primitive with the per-point features before feed them into the MLPs for the actor or the critic. From the critic maps of all primitives, we determine the primitive and location associated with the highest Q-value. The selected primitive, point location, and the corresponding motion parameters become the final output of the policy.

The proposed framework HACMan++ has two novelties compared to HACMan: First, it demonstrates the generality of the idea of spatially-grounded and temporally-abstracted action space beyond non-prehensile skills. Second, training with the skill library with RL may result in sequential reasoning of the skills. As we will show in the experiments, the robot may learn to poke the object first to facilitate grasping (similar idea as Chapter 4.5). This may allows the robots to perform a broader range of manipulation tasks. The proposed method is also related to previous work Dalal et al. [19] and Nasiriany et al. [88]. These work also use a set of manipulation primitives in RL. However, the skills are only temporally-abstracted but not spatially-grounded. In addition, they select the primitive based on policy output while we select the primitive according to the Q-values of different primitives.

In our experiments, we evaluate HACMan++ using a modified object pose alignment task and two tasks from the Maniskill2 benchmark [33]. In the modified

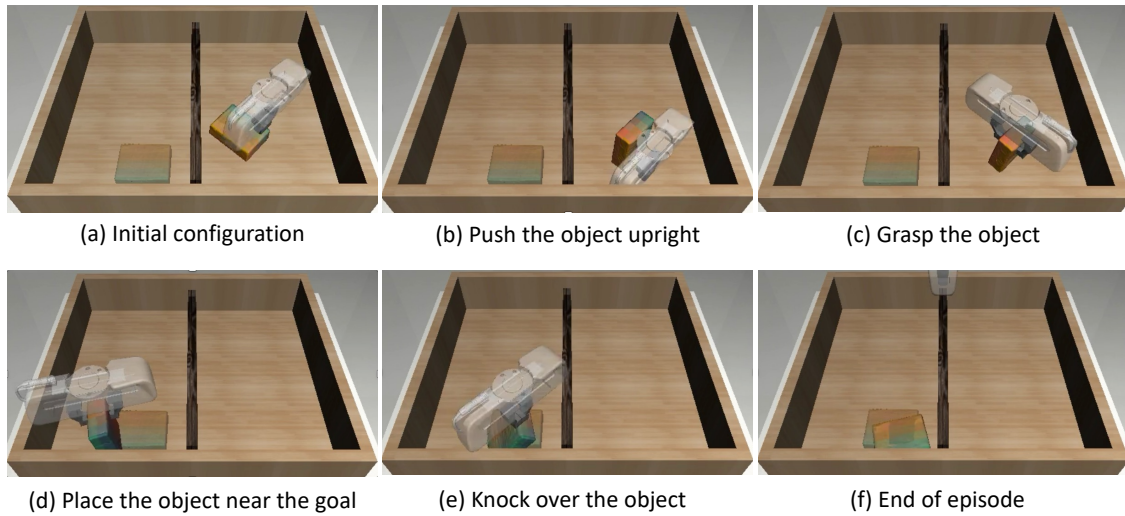


Figure 5.12: **An example of the modified object pose alignment task.** The goal of this task is to place the object at a target 6D pose in a different bin. The policy trained with HACMan++ learns to chain a sequence of prehensile and non-prehensile skills to achieve this task.

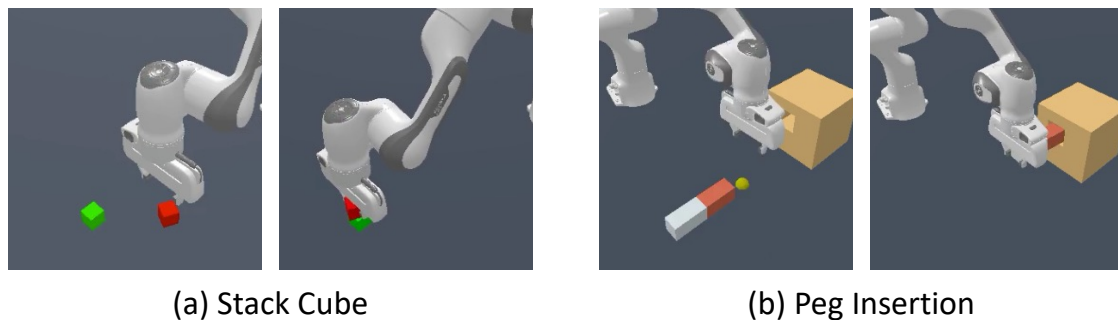


Figure 5.13: **HACMan++ policies for the Maniskill2 benchmark.** We also evaluate HACMan++ on the Maniskill2 benchmark, demonstrating the generality of the method in a wider range of manipulation tasks.

object pose alignment task shown in Figure 5.12, the robot needs to transfer the object from one bin to another which requires both prehensile and non-prehensile skills. The policy trained with HACMan++ exhibits complex sequential reasoning of using appropriate skills at various stages of the task. In the example shown in Figure 5.12, the object in the bin is initially too large for direct top-down picking. Thus, the policy first tilts the object upright and then grasps it from on the thinner side. The policy also learns the correct gripper orientation when the robot tries to grasp the object. Subsequently, the policy places the object near the goal by grounding the placing location on the background point cloud. In the final action, the policy knocks over the object to achieve the desired 6D pose. We also evaluate HACMan++ using two tasks from the manipulation benchmark Maniskill2 [33], cube stacking and peg insertion, as shown in Figure 5.13. The policy learns to solve the task efficiently by selecting appropriate primitives along with the corresponding locations and parameters. These experiments further underscore the flexibility and generality of HACMan framework in manipulation.

Chapter 6

Conclusions

6.1 Summary

In this thesis, we have explored various challenges and approaches to advance robot dexterity using reinforcement learning. Firstly, we proposed an offline RL algorithm “Policy in Latent Action Space (PLAS)” and introduced the “Offline Distillation Pipeline” to reuse robot data more efficiently in reinforcement learning. Secondly, we improved the robot’s dexterity beyond its hardware limitations with emergent “Extrinsic Dexterity” learned using RL by rethinking the environment of the MDP. Lastly, we have introduced “Hybrid Actor-Critic Map for Manipulation (HACMan)” to improve the generalization capability of complex motor skills to unseen objects by proposing a better action space. We demonstrate that re-framing the action space and the environment of the MDP can be an effective way of addressing the challenges in applying RL to manipulation. These advancements pave the way for robots to become more capable of learning complex and generalizable motor skills that can be used to perform a broader range of tasks in the real world.

In the future, the different methods proposed in this thesis could be combined together to build a better robotics system. For example, we may apply the idea of extrinsic dexterity from Chapter 4 in the object pose alignment task from Chapter 5. Currently, the results we presented for the object pose alignment task in Chapter 5 did not utilize extrinsic dexterity for two reasons: First, the robot can rotate most of the objects we considered without using the walls. Second, the randomized initial

object locations are not close to the walls. Suppose we train the policies with objects that require extrinsic dexterity to manipulate and with a broader range of initial object locations. In that case, we might see emergent extrinsic dexterity behaviors in the object pose alignment task. In addition, we may apply the action space proposed in HACMan from Chapter 5 to the Occluded Grasping task studied in Chapter 4 to make the RL policy generalize to more diverse objects. We may also use the offline RL algorithm from Chapter 2 and Chapter 3 to reuse real robot data for all tasks we studied in this thesis.

6.2 Future Directions

Throughout this thesis, our primary focus was on improving the dexterity and generalizability of a *single* robot to solve a *single* task at a time, which can be considered as an effort to build a better *specialist* robot. Moving forward, we are also excited about exploring a scalable and efficient way to build a *generalist* robot, capable of performing many tasks while achieving a similar level of dexterity and generalization as we showed in this thesis. Additionally, it would also be interesting to explore the possibility of scaling from a single robot to multiple robots with different embodiments.

From one task to many tasks. We need to consider several aspects to build a generalist robot that can perform many tasks. The most basic aspect is that we need methods that are general enough to be applied to different tasks. In this thesis, we have taken a step towards this goal. For instance, we have demonstrated the applicability of PLAS (Chapter 2) and HACMan (Chapter 5) to many tasks in the experiments. The Offline Distillation Pipeline (Chapter 3) and the concept of emergent Extrinsic Dexterity (Chapter 4) also have the potential to be applied to more tasks. To take one step further, we may consider how to share the data across different tasks to improve overall data efficiency. For example, training HACMan across various tasks with a shared network may enable positive transfer among the tasks. This research direction is related to many previous work in Multi-task RL [122, 139, 140] and Offline RL [64, 142]. In addition, we need to scale up the way we define the tasks. Typically, we need to tune the reward functions in RL to obtain desired robot behaviors when we develop a new task. In the future, it would

be interesting to explore how to efficiently specify a new task, such as by using visual language models as a reward function [21].

From one robot to many robots. A related future direction is to build a robot brain that can solve tasks across different robot embodiments such as different designs of robot arms, multi-fingered hands, wheeled robot, and quadruped robots. From one perspective, different robots can be considered as different “tasks” in robot learning. Thus, scaling robot learning to many robots also involve the challenges mentioned above. However, there might be additional challenges and opportunities. For example, a universal action representation can play a more important role in cross-embodiment training. In HACMan from Chapter 5, we propose an object-centric and robot-agnostic action representation that was applied to both a robot arm and a quadruped robot. Similar embodiment-agnostic representations can be especially beneficial when we scale the problems to multi-robot training.

6. Conclusions

Appendix A

Appendix to PLAS (Chapter 2)

A.1 Implementation Details

The implementation of our algorithm is based on the original implementation of BCQ: <https://github.com/sfujim/BCQ>. We train the CVAE first and then train the policy using the fixed decoder. The latent policy is a deterministic policy with tanh activation at the output. The output is then scaled by a hyperparameter max latent action. More discussions on the max latent action is in Appendix A.3. The perturbation layer is not used by default. We include the discussion on the effect of the perturbation layer in Appendix A.4.

Hyper-parameters for MuJoCo datasets: The actor, the critic, and the CVAE are optimized using Adam. The actor learning rate is $1e-4$ and the critic learning rate is $1e-3$. The CVAE learning rate is $1e-4$. Both the encoder and the decoder have two hidden layers (750, 750) by default. For datasets smaller than $1e6$ transitions such as the medium-replay datasets, we use (128, 128) to prevent overfitting. We train the CVAE for $5e5$ timesteps with batch size 100. The latent policy, the critic, and the perturbation layer have two hidden layers (400, 300). We use $\tau = 0.005$ for the soft target update. $\lambda = 1$ is used to calculate the Q-value target. The policy is trained for $5e5$ timesteps with batch size 100.

Hyper-parameters for the robot experiment: The actor and critic learning rates are set to $3e-4$ and the CVAE learning rate is $1e-4$, with Adam as the optimizer. All of the networks have two hidden layers of size 64, including the actor, the critic,

the encoder, and the decoder. The smaller network sizes are to prevent overfitting. The CVAE is trained for 15000 iterations. For soft target update we use $\tau = 0.005$. We use $\lambda = 0.75$ for clipped double Q learning and use batch size of 256. The max latent action is set to 2.0.

A.2 D4RL Results

To benchmark the performance of our algorithm, we include the full results for the d4rl MuJoCo datasets here as a reference. The numbers for the baselines are from the d4rl paper [29]. The results are averaged over 3 seeds. “PLAS” refers to the latent policy without the perturbation layer. “PLAS+P” refers to the latent policy + perturbation layer. For max latent action, hopper-medium-replay and halfcheetah-medium-expert use 0.5 and all the other locomotion datasets use 2 for both “PLAS” and “PLAS+P”. For the max perturbation, we report the best result from Appendix A.4. Our method consistently achieves good performance especially on medium-expert and medium-replay datasets. The other baselines work well on a part of the datasets and fail on the others.

In the current version of the d4rl dataset, hopper-medium-expert is actually a combination of the medium-replay and the expert datasets instead of the medium and the expert datasets. We have verified that the results given in their paper also correspond to the medium-replay + expert dataset. In Table A.1 and Table A.2 below, we use hopper-medium-expert(a) to refer to the results on this dataset. In addition, we generate the actual hopper-medium-expert by concatenating the medium and the expert datasets, referred to as hopper-medium-expert(b) in the table. The results from Figure 2.4 and the other experiments in the appendix are all based on hopper-medium-expert(b).

A.3 Sensitivity Analysis: Max Latent Action

The max latent action limits the range of output for the latent policy to ensure that the output has a high probability under the latent variable prior of the CVAE. As mentioned in Section 2.4.1, if the output of the latent policy has a high probability

Table A.1: D4rl Benchmark Results: Average Reward

Dataset	BEAR	BRAC-v	BCQ	PLAS (Ours)	PLAS+P (Ours)
walker2d-medium-expert	1842.7	4926.6	2640.3	4113.2	4465.0
hopper-medium-expert-(a)	3113.5	5.1	3588.5	3593.7	3062.5
hopper-medium-expert-(b)	2648.4	2245.7	2021.7	3592.4	3518.5
halfcheetah-medium-expert	6349.6	4926.6	7750.8	11716.9	12051.4
walker2d-medium-replay	883.8	44.5	688.7	1387.9	658.4
hopper-medium-replay	1076.8	-0.8	1057.8	888.4	1669.6
halfcheetah-medium-replay	4517.9	5640.6	4463.9	5172.6	5397.4
walker2d-medium	2717.0	3725.8	2441.0	2047.0	3072.4
hopper-medium	1674.5	990.4	1752.4	1050.4	1182.1
halfcheetah-medium	4897.0	5473.8	4767.9	4602.6	4964.6
walker2d-random	336.3	87.4	228.0	104.0	311.6
hopper-random	349.9	376.3	323.9	320.5	412.2
halfcheetah-random	2831.4	3590.1	-1.3	2922	3235.8

Table A.2: D4rl Benchmark Results: Normalized Score

Dataset	BEAR	BRAC-v	BCQ	PLAS (Ours)	PLAS+P (Ours)
walker2d-medium-expert	40.1	81.6	57.5	89.6	97.2
hopper-medium-expert-(a)	96.3	0.8	110.9	111.0	94.7
hopper-medium-expert-(b)	82.0	69.6	62.7	111.0	108.7
halfcheetah-medium-expert	53.4	41.9	64.7	96.6	99.3
walker2d-medium-replay	19.2	0.9	15	30.2	14.3
hopper-medium-replay	33.7	0.6	33.1	27.9	51.9
halfcheetah-medium-replay	38.6	47.7	38.2	43.9	45.7
walker2d-medium	59.1	81.1	53.1	44.6	66.9
hopper-medium	52.1	31.1	54.5	32.9	36.9
halfcheetah-medium	41.7	46.3	40.7	39.3	42.2
walker2d-random	7.3	1.9	4.9	3.1	6.8
hopper-random	11.4	12.2	10.6	10.5	13.3
halfcheetah-random	25.1	31.2	2.2	25.8	28.3

Table A.3: D4rl Results on More Datasets: Average Reward. For these datasets, we searched over 0.5, 1, 2 for max latent action and report the best results.

Dataset	BC	SAC-off	BEAR	BRAC-v	BCQ	PLAS (Ours)
maze2d-umaze	29.0	145.6	28.6	1.7	41.5	102.6
maze2d-medium	93.2	82.0	89.8	102.4	35.0	109.6
maze2d-large	20.1	1.5	19.0	115.2	23.2	334.6
antmaze-umaze	0.7	0.0	0.7	0.7	0.6	0.7
antmaze-umaze-diverse	0.6	0.0	0.6	0.7	0.7	0.5
antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	0.2
antmaze-medium-diverse	0.0	0.0	0.1	0.0	0.0	0.0
antmaze-large-play	0.0	0.0	0.0	0.0	0.0	0.0
antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	0.0
pen-human	1121.9	284.8	66.3	114.7	2149.0	2101.0
hammer-human	-82.4	-214.2	-242.0	-243.8	-210.5	324.7
door-human	-41.7	57.2	-66.4	-66.4	-56.6	73.3
relocate-human	-5.6	-4.5	-18.9	-19.7	-8.6	7.1
pen-cloned	1791.8	797.6	885.4	22.2	1407.8	1558.0
hammer-cloned	-175.1	-244.1	-241.1	-236.9	-224.4	-142.9
door-cloned	-60.7	-56.3	-60.9	-59.0	-56.3	41.2
relocate-cloned	-10.1	-16.1	-17.6	-19.4	-17.5	-16.7
pen-expert	2633.7	277.4	3253.1	6.4	3521.3	3693.3
hammer-expert	16140.8	3019.5	16359.7	-241.1	13731.5	16333.5
door-expert	969.4	163.8	2980.1	-66.6	2850.7	3004.0
relocate-expert	4289.3	-18.2	4173.8	-21.4	1759.6	4528.5
kitchen-complete	1.4	0.6	0.0	0.0	0.3	1.4
kitchen-partial	1.4	0.0	0.5	0.0	0.8	1.8
kitchen-mixed	1.9	0.1	1.9	0.0	0.3	1.6

Table A.4: D4rl Results on More Datasets: Normalized Score

Dataset	BC	SAC-off	BEAR	BRAC-v	BCQ	PLAS (Ours)
maze2d-umaze	3.8	88.2	3.4	-16.0	12.8	57.0
maze2d-medium	30.3	26.1	29.0	33.8	8.3	36.5
maze2d-large	5.0	-1.9	4.6	40.6	6.2	122.7
antmaze-umaze	65.0	0.0	73.0	70.0	78.9	70.7
antmaze-umaze-diverse	55.0	0.0	61.0	70.0	55.0	45.3
antmaze-medium-play	0.0	0.0	0.0	0.0	0.0	16.0
antmaze-medium-diverse	0.0	0.0	8.0	0.0	0.0	0.7
antmaze-large-play	0.0	0.0	0.0	0.0	6.7	0.7
antmaze-large-diverse	0.0	0.0	0.0	0.0	2.2	0.3
pen-human	34.4	6.3	-1.0	0.6	68.9	67.3
hammer-human	1.5	0.5	0.3	0.2	0.5	4.6
door-human	0.5	3.9	-0.3	-0.3	0.0	4.4
relocate-human	0.0	0.0	-0.3	-0.3	-0.1	0.3
pen-cloned	56.9	23.5	26.5	-2.5	44.0	49.0
hammer-cloned	0.8	0.2	0.3	0.3	0.4	1.0
door-cloned	-0.1	0.0	-0.1	-0.1	0.0	3.3
relocate-cloned	-0.1	-0.2	-0.3	-0.3	-0.3	-0.2
pen-expert	85.1	6.1	105.9	-3.0	114.9	120.7
hammer-expert	125.6	25.2	127.3	0.3	107.2	127.1
door-expert	34.9	7.5	103.4	-0.3	99.0	104.2
relocate-expert	101.3	-0.3	98.6	-0.4	41.6	106.9
kitchen-complete	33.8	15.0	0.0	0.0	8.1	34.8
kitchen-partial	33.8	0.0	13.1	0.0	18.9	43.9
kitchen-mixed	47.5	2.5	47.2	0.0	8.1	40.8

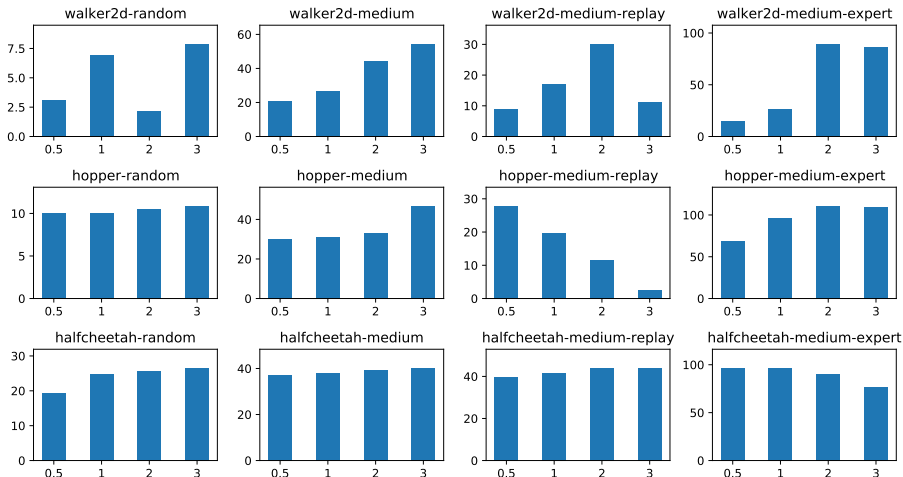


Figure A.1: Sensitivity analysis on the max latent action for the latent policy: X-axis is the max latent action value. Y-axis is the normalized score.

under the distribution of the latent variable prior, then the decoded output has a high probability to be within the distribution of the behavior policy. Larger max latent action may result in out-of-distribution actions. On the other hand, smaller max latent action will make the action selection more restrictive. We evaluated the effect of the max latent action from $\{0.5, 1, 2, 3\}$ over the MuJoCo datasets in d4rl as shown in Figure A.1. In hopper-medium-replay and halfcheetah-medium-expert, 0.5 works the best. In most cases, 2 works well. Thus, we use 0.5 for hopper-medium-replay and halfcheetah-medium-expert and 2 by default for all the other environments for simplicity. Note that all the experiments for walker2d-random are unstable, thus the comparison across different parameters might not be valuable since we only average across 3 seeds.

A.4 Ablation Study: Perturbation Layer

We provide a full comparison of the perturbation layer on MuJoCo datasets in this section. We summarize the results with max perturbation $\epsilon \in \{0, 0.01, 0.05, 0.1, 0.2, 0.5\}$ in Figure A.2. $\epsilon = 0$ is only using the Latent Policy without the perturbation layer. As mentioned above, the walker2d-random experiments are not stable, thus the comparison might not be valuable. In most cases, the addition of the perturbation layer sometimes improves the performance, but not significant. With a large ϵ higher

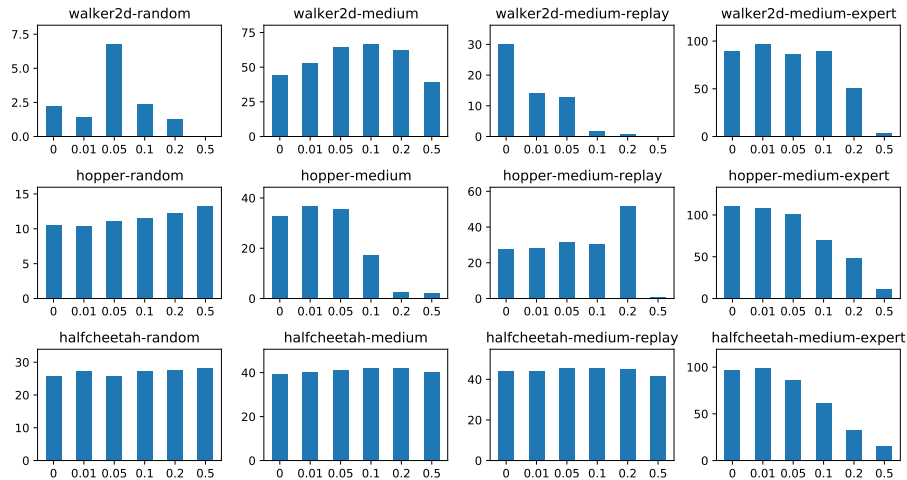


Figure A.2: Ablation study on the perturbation layer: X-axis is the max perturbation. Y-axis is the normalized score.

than a certain value, the performance usually drops. Thus, we make the perturbation layer an optional component in our method.

A.5 Empirical Analysis on MMD Constraint

To understand the limitation of using sampled MMD constraint to limit out-of-distribution actions, we simulate the MMD loss in different scenarios. In the first experiment, we construct a one-dimensional behavior policy sampling from $N(0, 1)$ and an agent policy sampling from $N(0, x)$, where x is a variable. In Figure A.3 below, we plot the MMD loss for this agent policy with different x as the x-axis with various kernel parameters. Ideally, the loss should be smaller than a threshold for any $x \leq 1$ to allow the agent policy to select the best action within the support with a higher probability. However, as shown in Figure A.3, this is only roughly satisfied with the Gaussian kernel and large sigma. Sampled MMD constraint aims to match the entire support of two distributions and could be overly restrictive.

In Figure A.4, we further demonstrate the limitation of MMD constraint on multimodal distributions. We assume a behavior policy sampling uniformly from $[-2, -1] \cup [1, 2]$ and an agent policy sampling from $N(x, 0.5)$. We vary the mean value x in the x-axis of the figures. In this case, we expect the minimum loss to happen at $x = -1.5$ and $x = 1.5$ to prevent out of distribution actions. However,

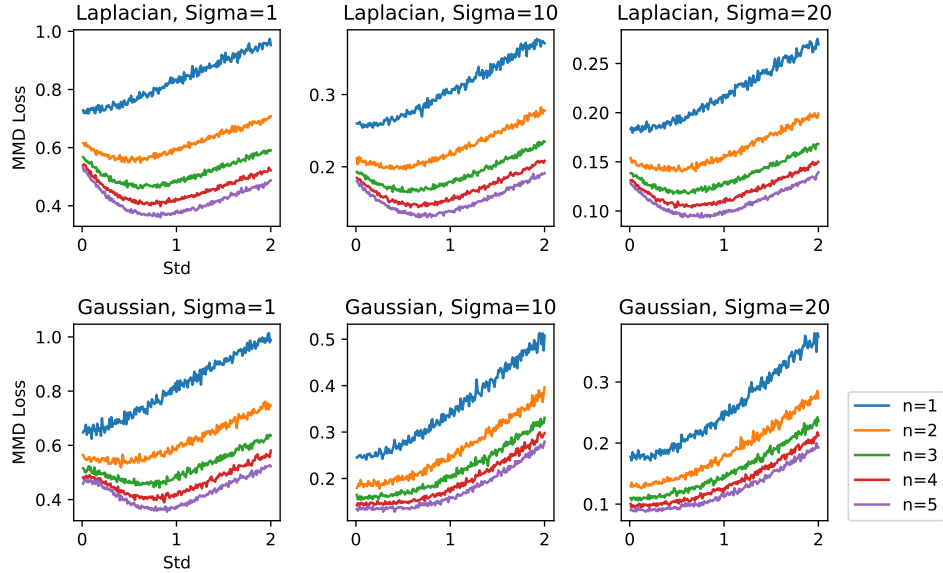


Figure A.3: Simulated MMD loss with $N(0,1)$ as the behavior policy.

the simulation results show that this is not the case for any of the curves. With large sigma, the minimum MMD loss occurs at $x = 0$, which lies in the “hole” of the behavior policy distribution.

A.6 Robot Experiment

For the real robot experiments, we use a Sawyer robot equipped with a WSG 32 gripper and a WSG DSA tactile sensor finger. The physical setup involves a cloth with one corner clamped on a fixture. The task is to slide along the cloth as far as possible, ideally until the other corner is reached.

The movement of the end-effector is constrained to a vertical plane from the fixture. The environment’s action space consists of the incremental movement in horizontal (x) and vertical (z) directions for a single time step. The observation space of the environment consists of the tactile sensor readings, end-effector force, and end-effector pose (z position and angle). The observations are thus in the form of a 89-d vector. The action space consists of horizontal and vertical delta position actions.

The reward and the terminal conditions are designed to encourage large movement in the x-direction without losing the cloth. For each timestep, if the gripper is

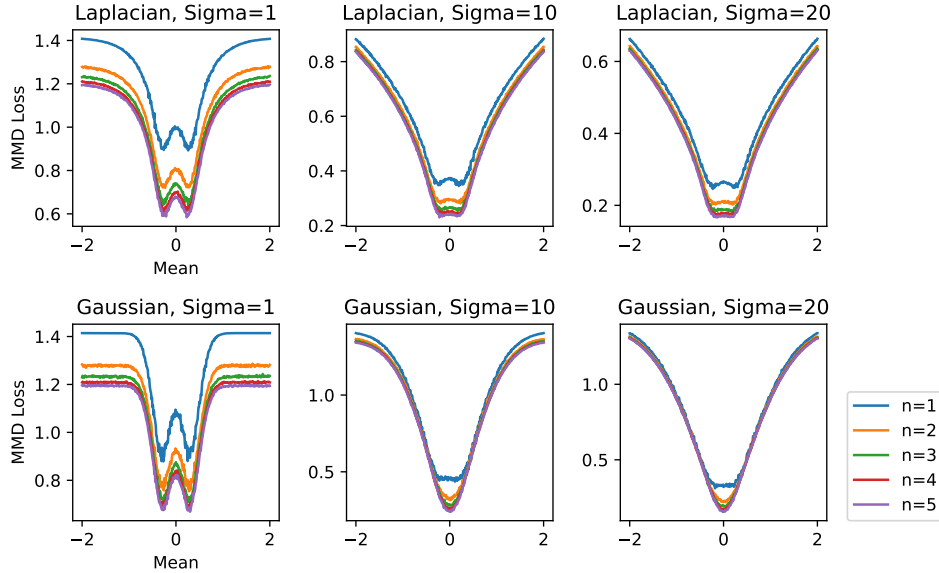


Figure A.4: Simulated MMD loss with $N(0,1)$ as the behavior policy.

sliding along the cloth, it receives a reward equal to the horizontal action. This is to encourage faster sliding. However, if the edge is lost from the gripper, the reward will be zero for that timestep and the episode ends. This failure condition is detected based on the gripper width adjustment procedure discussed below. In addition, the maximum episode length is 70 timesteps.

We use a hard-coded procedure to adjust the gripper width. The overall objective is to get clearer tactile readings of the cloth by grasping tightly while allowing the gripper to slide easily without too much friction. The adjustment is based on the coverage and the mean value of the tactile readings as well as the end-effector force readings. When the gripper width is at the minimum value and there is still no tactile reading or force reading, we consider it as a failure and the episode ends.

Appendix B

Appendix to Robot Lifelong Learning (Chapter 3)

B.1 Additional results on the offline distillation pipeline

In Figure 3.6, we present the results of the two-stage lifelong learning experiments when Env-A is the default environment and Env-B has the hip joint of the robot deformed by 0.3 rad. In this section, we include more results across more environment variations. To demonstrate the difficulty of data collection with conservative algorithms, Figure B.1 shows the performance of each algorithm when they are trained from scratch, which corresponds to the first stage of the lifelong learning setup discussed in Section 3.7.2. MPO performs significantly better than both versions of CRR. Figure B.2 shows the performance during Stage-2 where all of the algorithms loaded an agent which is pretrained in Env-A (the default environment) for 0.2M steps. The Offline Distillation Pipeline can achieve the best performance consistently across different Env-B variations.

B.2 Algorithm

A summary of the algorithm is included in Algorithm 2.

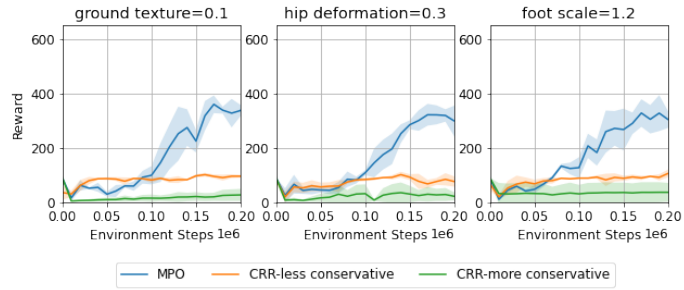


Figure B.1: Comparison of off-policy algorithms for training from scratch which corresponds to the beginning stage of a lifelong learning experiment. This is an extension of Figure 3.6 Stage-1 result.

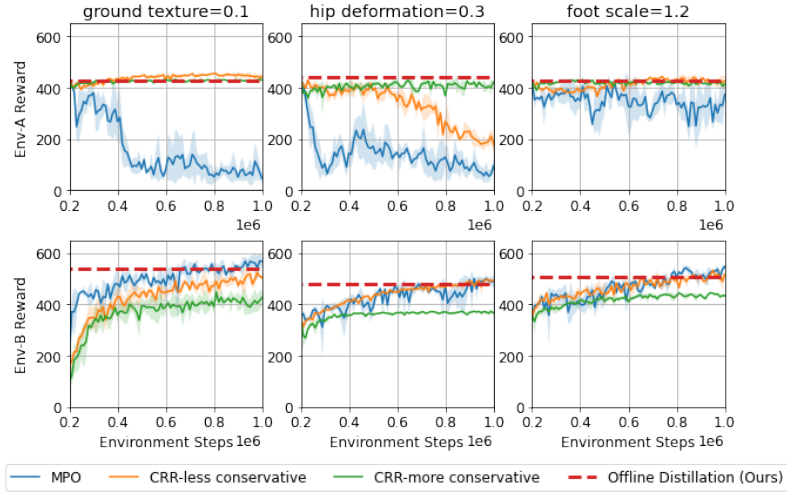


Figure B.2: Comparison of off-policy algorithms during Stage-2 of the lifelong learning experiment. This is an extension of Figure 3.6 Stage-2 results.

Algorithm 2: Offline Distillation Pipeline

Input: Maximum data collection step N . Maximum offline training step M .
 A training environment with non-stationary dynamics P_t . A testing environment with dynamics P_{test} within the support of the training distribution.

// Online Interaction Phase
 Initialize policy π_0 and Q-function Q_0 . Initialize an empty replay buffer D .
for $t \leftarrow 1$ **to** M **do**
 | Collect data with the current dynamics P_t and save the transition (s_t, a_t, s_{t+1}) into the replay buffer D .
 | Sample a batch of transitions $\{(s_i, a_i, s_{i+1})\}_i$ from D .
 | Update Q_0 with MPO Policy Evaluation Step according to Equation 3.1.
 | Update π_0 with MPO Policy Improvement Step according to Equation 3.2.
end

// Offline Distillation Phase
 Initialize policy π_1 and Q-function Q_1 .
for $t \leftarrow 1$ **to** M **do**
 | Sample a batch of transitions $\{(s_i, a_i, s_{i+1})\}_i$ from D .
 | Update Q_1 with CRR Policy Evaluation Step according to Equation 3.1.
 | Update π_1 with CRR Policy Improvement Step according to Equation 3.4.
end

// Deployment
 Execute the distilled policy π_1 in the testing environment P_{test} .

B.3 Additional results with three environments and parallel sharing

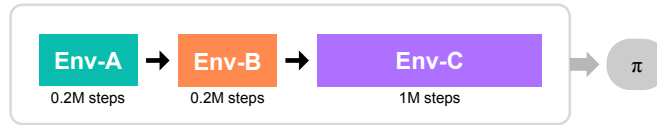
In Section 3.7, we focus on the experiment setup with two environments: switching from Env-A to Env-B during training, and evaluate on both Env-A and Env-B during evaluation. To show the generality of the discussed problems and the proposed solutions, we include experiments with three environments in this section with both sequential training and parallel training. For each experiment setup, we include three different combinations of environment variations as listed in Table B.1.

	Env-Combination-1	Env-Combination-2	Env-Combination-3
Env-A	Default	Default	Default
Env-B	hip deformation = 0.2	hip deformation = 0.3	foot scale = 1.2
Env-C	hip deformation = 0.2 foot scale = 1.6	hip deformation = 0.3 foot scale = 1.6	foot scale = 1.2 ground texture = 0.1

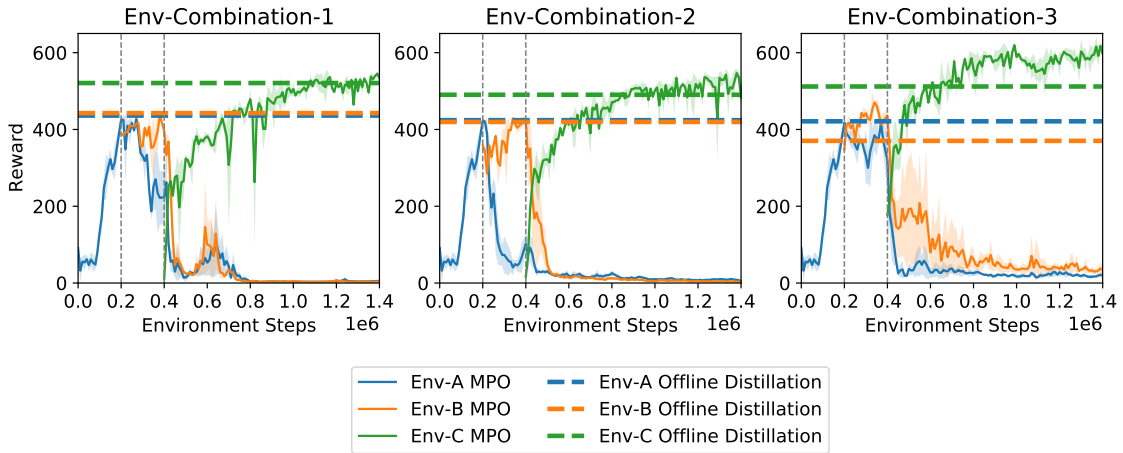
Table B.1: Combinations of different environment variations in the three environment experiments.

B.3.1 Forgetting and the effectiveness of Offline Distillation

In this section, we demonstrate the forgetting issue and the effectiveness of the Offline Distillation Pipeline with a three-stage experiment. As shown in Figure B.3a, the agent first collects data in Env-A for 0.2M steps, switches to Env-B for 0.2M steps, and eventually experiences Env-C for 1M steps. During this online interaction phase, the agent is trained with MPO and keeps all the history data in the replay buffer. We plot the evaluation performance for all three environments across this procedure in Figure B.3b. Similar to the observations from Section 3.7.2, the agent experiences significant performance drop on the previous environments after the switches. At the end of the online interaction phase, we run Offline Distillation on the entire dataset indicated as the dotted lines in Figure B.3b. With Offline Distillation, we can recover a policy that works on all the previous environments.



(a) Experiment Setup: The agent experiences three stages across the online interaction phase.



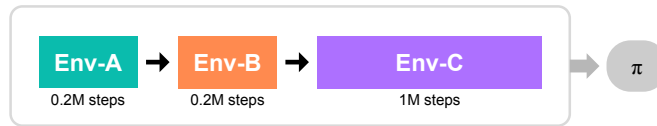
(b) Reward curves

Figure B.3: Lifelong learning experiments with three stages. The policy trained with MPO during online interaction experiences significant forgetting on previous environments. With Offline Distillation at the end, the policy can recover the performance effectively over all the previous environments.

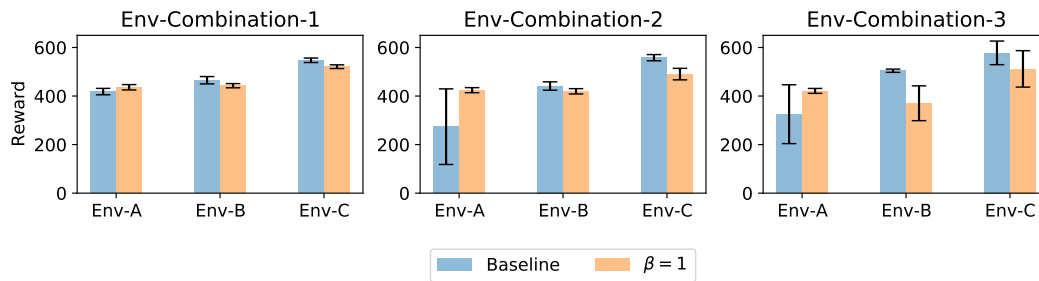
B.3.2 Imbalance experience in offline distillation

We construct additional experiment setups to demonstrate the imbalance issue in Offline Distillation and verify the proposed solution of using a more conservative transformation function. First, we follow the setup from the previous section where the agent experiences three environments consecutively (Figure B.4a). We compare the results of Offline Distillation with an indicator function (Baseline) and an exponential function with $\beta = 1$. From Figure B.4b, the baseline has a significant performance drop in Env-A for Env-Combination-2 and Env-Combination-3, similar to the results from Section 3.7.3. With the exponential function with $\beta = 1$, CRR training during the distillation phase is more reliable across all the environments.

We evaluate two additional experiment setup with parallel training. In Figure B.5a, the agent is first trained on Env-A. After 0.2M steps, the agent is copied into two different agents (including the policy, the Q-function, and the replay buffer). Two copies of the agents continues the training on Env-B and Env-C independently. At the end of the online interaction phase, we run Offline Distillation on the entire dataset. In Figure B.6a, one agent is trained on Env-A and then switches to Env-B. Another agent is trained from scratch on Env-C. The offline distillation is performed on the combined dataset of these three environments. For both experiment setups, the baseline often experiences performance drop in Env-A. For the setup in Figure B.6b, Env-C might also suffer from a performance drop since the dataset is trained from scratch and it is relatively small. As we discussed in Section 3.6, both size and quality contributes to the performance drop of imbalanced dataset. For both setups, CRR with exponential function with $\beta = 1$ can largely fix this issue.

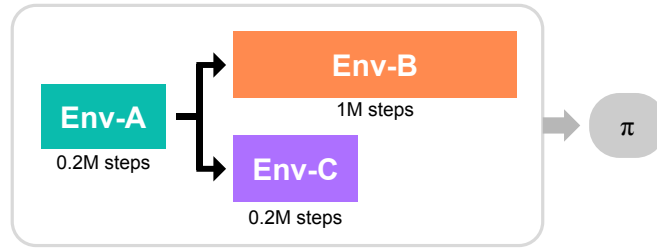


(a) Experiment Setup: The agent experiences three stages across the online interaction phase.

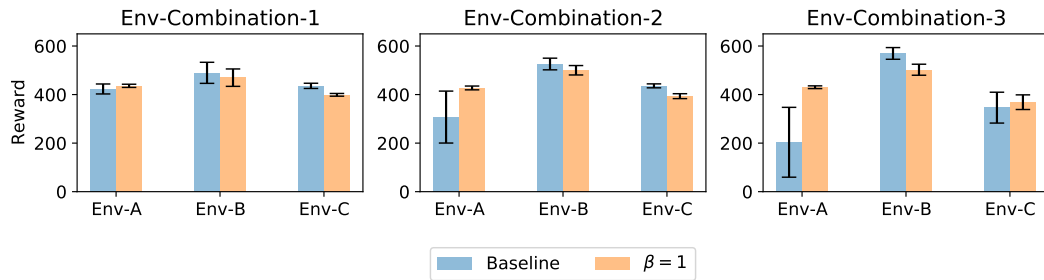


(b) Final performance of Offline Distillation.

Figure B.4: Evaluation of Offline Distillation with different transformation functions with a three-stage experiment setup.

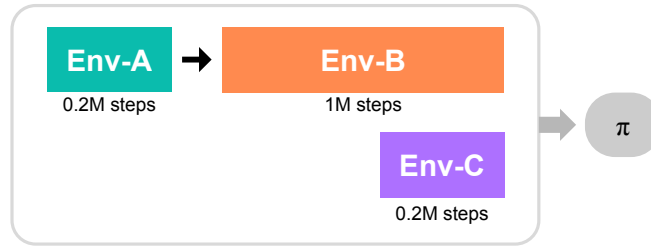


(a) Experiment Setup: Parallel training during the online interaction phase (type-A).

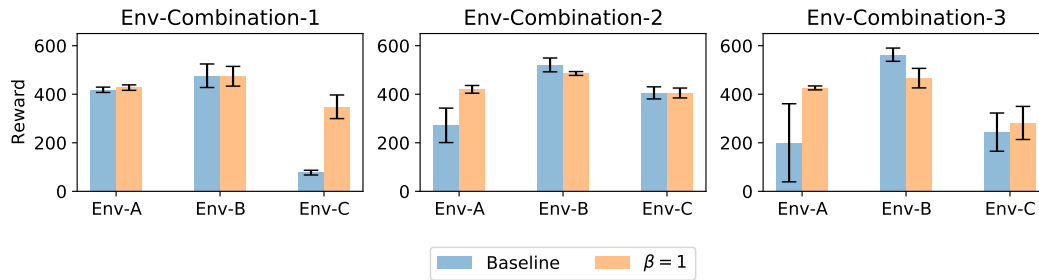


(b) Final performance of Offline Distillation.

Figure B.5: Evaluation of Offline Distillation with different transformation functions with a parallel training experiment setup with three environments (type-A).



(a) Experiment Setup: Parallel training during the online interaction phase (type-B).



(b) Final performance of Offline Distillation.

Figure B.6: Evaluation of Offline Distillation with different transformation functions with a parallel training experiment setup with three environments (type-B).

Appendix C

Appendix to Grasp the Ungraspable (Chapter 4)

C.1 Additional Results

C.1.1 Sensitivity analysis on physical parameters

In addition to the evaluation on policy generalization in Section 4.5.4, we modify the important physical parameters one at a time to understand the sensitivity of the policy performance to these parameters (Figure C.1). We compare the same baselines as Section 4.5.4: policies trained over a fixed environment (*Fixed Env*), policies trained with ADR (*With ADR*) and open-loop trajectories generated by rolling out the fixed env policy in the default environment (*Open Loop*). The ranges of parameters are chosen to create a performance drop for all the baselines as a stress test. Similar to what we observe in Section 4.5.4, the policy can cover a wider range of physical parameters with closed-loop execution and with ADR.

C.1.2 Sensitivity analysis on object pose estimation noise

The proposed system takes the 6D object pose as policy input. In the real world, object pose estimation might be noisy. In this section, we evaluate the policies trained with ADR with different levels of pose estimation noise for each dimension of the 6D

C. Appendix to Grasp the Ungraspable (Chapter 4)

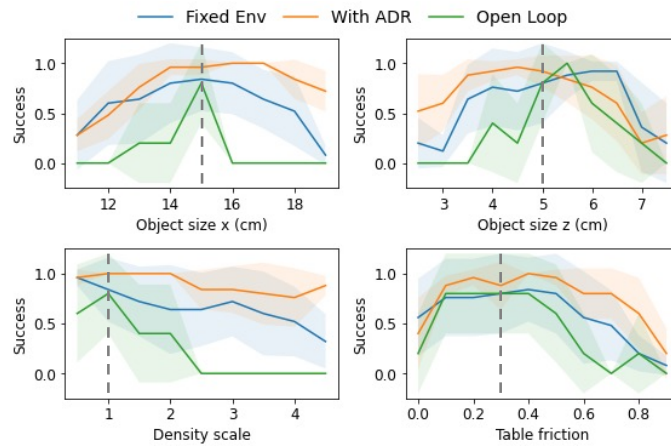


Figure C.1: We evaluate the generalization of policies by changing one parameter at a time. The dashed lines indicate the default values of these parameters in the fixed environment.

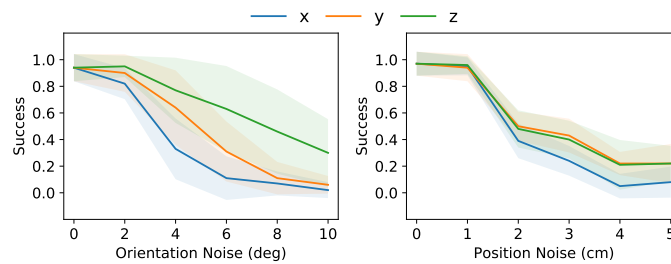


Figure C.2: We evaluate the sensitivity of the ADR policies on object pose estimation noise.

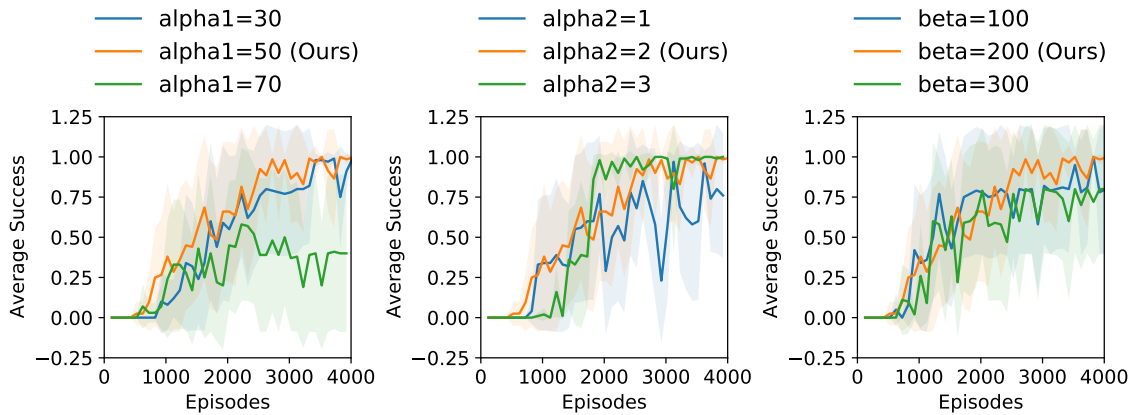


Figure C.3: Training curves with different reward weights. For each plot, we train the policies by changing one of the weight terms to three different values.

object pose (Figure C.2). During evaluation, for each timestep across the episode, we sample a scalar noise from a Gaussian distribution $\mathcal{N}(\mu = 0, \sigma = x)$ and add it to one dimension of the object pose. The standard deviations $\sigma = x$ are shown as the x-axis in the plots. The shaded area indicates the standard deviation of the success rates across seeds.

C.1.3 Reward term weights

The reward function shown in Equation 4.1 is composed of three terms with weights α_1 , α_2 and β . α_1 and α_2 weight the translation and rotation error between the target grasp and the current end-effector. β weights the target grasp occlusion penalty which is to penalize the agent if the target grasp configuration is in collision with the table. We use $\alpha_1 = 50$, $\alpha_2 = 2$, $\beta = 200$ in all the experiments. In this section, we train the policies with different weight values to see how much reward tuning is required to achieve reasonable performance for the occluded grasping task. Figure C.3 shows that the policy is not too sensitive in most of the case we tested except that a higher $\alpha_1 = 70$ leads to a 50% drop in performance.

C.2 Implementation Details

C.2.1 Simulation environment

We build the simulation environment with Robosuite [153] which uses the MuJoCo simulator [125]. Each episode has a length of 40 timesteps which corresponds to 20 seconds of real time execution. At the beginning of each episode, we set the robot arm to an initial joint configuration with Gaussian noise in the joint angles of 0.02 rad. We use a box-shaped object in the simulation environment. The dimensions of the box are randomized in the ADR experiments. One important note on the simulator environment is the parameters of the MuJoCo solver. We notice that MuJoCo sometimes creates unrealistic contacts with the default solver. We reduce the simulation solver timestep from the default value of 0.002 to 0.001 and set the “noslip iterations” to 20 which significantly improved simulation quality on contacts.

C.2.2 Grasp configurations

In this work, we focus on grasping large objects from the side because this is a task that may demonstrate the benefits of extrinsic dexterity. For single grasp experiments, a default grasp location is shown in Figure 4.1. In multi-grasp experiments, the grasps are sampled from a distribution shown in Figure 4.6. The grasps are sampled along the side of the box and they are 2 cm away from the edges. These grasp configurations are supposed to be the input to our proposed system, and could be replaced by other grasp generation methods.

C.2.3 Success rate calculation

In simulation, the success of the task is computed as $\mathbb{1}(\Delta T < 3\text{ cm}) \cdot \mathbb{1}(\Delta\theta < 10\text{ deg})$ at the end of an episode. As defined in Section 4.3, ΔT is the position difference between the end-effector and the target grasp and $\Delta\theta$ is the orientation difference. The success is defined in this way because we focus on reaching the desired grasp. One alternative is to evaluate the final grasping success by closing the gripper and lift the object. However, this will increase the simulation time during training. To confirm that the pose difference is a good proxy for the final grasping success, we evaluated a trained policy and verified that if the robot closes the gripper at the end of a successful episode according to the pose difference metric, it is able to lift the object 100% of the time.

For the real robot experiments, we evaluate success by closing the gripper and lifting the object; if the object was successfully lifted, we will mark it as a successful episode.

C.2.4 Observation and action space

As mentioned in Section 4.4.2, the observation includes a target grasp configuration in the object frame Og , the pose of the end-effector in the world frame WE and the object pose in the world frame WO . One implementation detail is that we also include the pose of the end-effector in the object frame ${}^OE = ({}^WO)^{-1}({}^WE)$ because we found that it sometimes speeds up learning. Each pose is represented as a 3D translation vector and a 4D quaternion representation of the rotation.

The action space of the policy is the delta pose of the end-effector ΔE in its local frame represented by a vector of translation $p \in \mathbb{R}^3$ and a 3D vector of rotation $q \in SO(3)$ with axis-angle representation. An outline of the policy execution pipeline is shown in Figure C.4. ΔE is then passed into a collision check function to form a desired pose E_d which will be sent to a low-level controller.

C.2.5 Low-level controller

Handling joint limit: Although we may use nullspace in the operational space controller to avoid reaching joint limit, in practice, certain desired end-effector poses still reach joint limits that cannot be avoided by nullspace. Thus, we handling the joint limit in the following way. If the corresponding joint configuration of the desired pose is going to reach joint limits, we will overwrite the policy action to output the desired pose of the previous timestep to the low-level controller. In detail, we use the Jacobian J to estimate the joint configuration of the desired pose:

$$\theta_{joints}^{t+1} = \theta_{joints}^t + J^{-1} \cdot \Delta E \quad (\text{C.1})$$

where θ_{joints} are the joint angles and ΔE is the output of the policy. If any joint in θ_{joints}^{t+1} is close to the limit, the low-level controller will use the previous desired pose E_d instead.

Parameters of the Operational Space Controller: We use $K_p = 300$ for position error, $K_p = 30$ for orientation error, and $K_d = \sqrt{K_p}$. These values are chosen by making sure that the real robot is compliant enough to safely collide with the object and the bin without damage. In Figure D.5, the baseline of “High-gain OSC” uses $K_p = 600$ for position error, $K_p = 60$ for orientation error, and $K_d = \sqrt{K_p}$. This baseline with less compliance is not only slower to train in the simulation, but also not safe to execute on the real robot for our task which involves rich contacts and relies on environment constraints. During our initial experiments, with the high-gain OSC, the robot deforms the object and the bin surface.

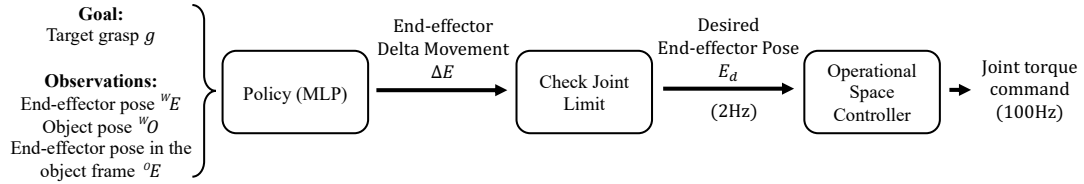


Figure C.4: Outline of policy execution: Given the observation, the policy outputs an end-effector delta movement. If the desired pose is within the joint limit of the robot, it will be sent to the low-level controller which operates at a higher frequency.

C.2.6 Multi-Grasp Training with Curriculum

Here are more details on multi-grasp training. When the success rate of policy on a boundary case of the training range is above 0.8, it will expand the range of grasps by 0.25 (See Figure 4.6 for parameterizations of the grasp configurations). For example, if the policy is currently training with grasps $[1, 2]$, and the success rate evaluated at grasp ID 1 is above 0.8, the new training range will be $[0.75, 2]$. This is following a similar procedure as Automatic Domain Randomization, but randomizing goals instead of simulation parameters.

C.2.7 RL Training

We use Soft Actor Critic [34] to train the RL policy with the implementation from rlkit (<https://github.com/rail-berkeley/rlkit>). Hyperparameters for SAC training are included in Table C.1. Since the task is conditioned on the target grasp as a goal, we use Hindsight Experience Replay [7] for all the experiments with 60% original goals and 40% of the goals sampled from the same rollout. We compare the policies across 5 random seeds of each method and plot the average performance with standard deviation across seeds. We use 10 episodes for each evaluation setting.

C.3 Automatic Domain Randomization

As discussed in Section 4.4.6, we use Automatic Domain Randomization [91] to improve policy generalization across environment variations. In ADR, the policy is first trained with an environment with very little randomization, and then we gradually

Table C.1: Hyperparameters for RL training.

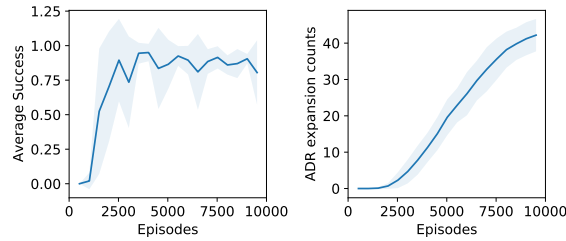
Hyperparameters	Values
Optimizer	Adam
Learning rate - Policy	1e-3
Learning rate - Q-function	5e-4
Networks	[512, 512, 512] MLP
Batch size	256
Nonlinearity	ReLU
Soft target update (τ)	0.005
Replay buffer size	1e6
Discount factor (γ)	0.99
HER rollout goals	40%

expand the variations based on the evaluation performance. For a set of environment parameters λ_i , each λ_i is sampled from a uniform distribution $\lambda_i \sim U(\phi_i^L, \phi_i^H)$ at the beginning of each episode. During training, the policy will be evaluated at these boundary values $\lambda_i = \phi_i^L$ or $\lambda_i = \phi_i^H$. If the performance is higher than a threshold, the boundary value will be expanded by an increment Δ . For example, if the performance at $\lambda_i = \phi_i^H$ is higher than the threshold, the training distribution becomes $\lambda_i \sim U(\phi_i^L, \phi_i^H + \Delta)$ in the next iteration. Compared to directly training the policy with the entire variations, Automatic Domain Randomization can reduce the need of manually tuning a suitable range of variations for each environment parameter.

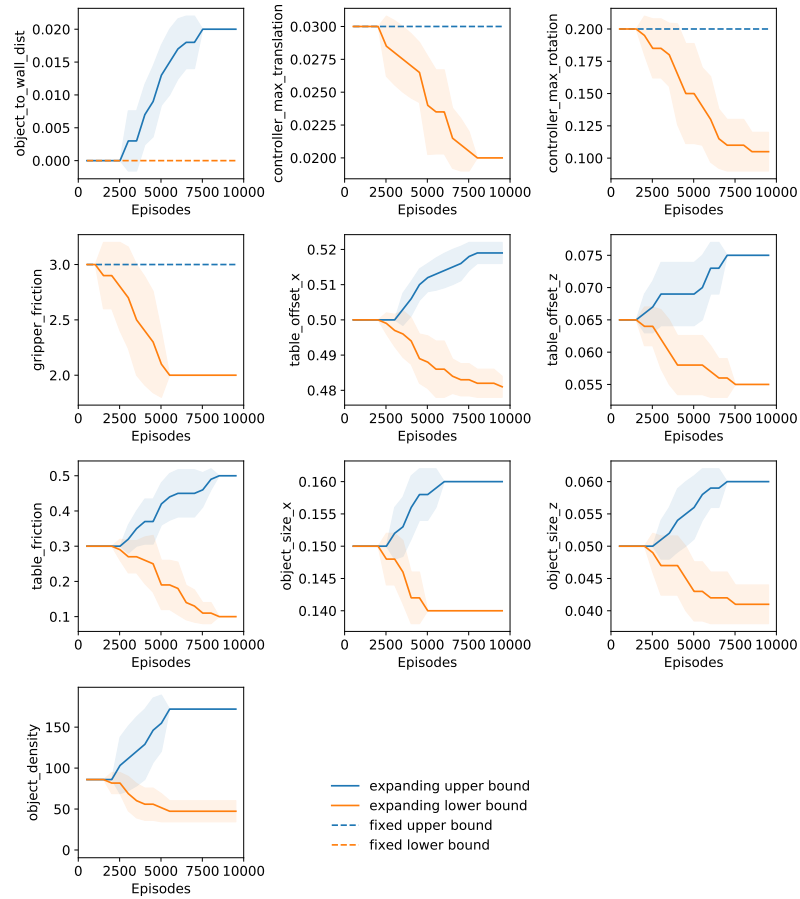
Table C.2 summarized the simulation parameters in the experiment. All the parameters are uniformly sampled from these ranges at the beginning of each episode. The ranges of the parameters start from a single initial value and gradually expand to a wider range according to the pre-specific increment step $+\Delta$ on the upper bound and the decrement step $-\Delta$ at the lower bound.

We include the training plots of the ADR policies in Figure C.5. Dashed lines in Figure C.5 indicate fixed parameter boundaries where we do not intent to expand. The final ranges are used when we sample 100 environments for evaluation in Section 4.5.4.

C. Appendix to Grasp the Ungraspable (Chapter 4)



(a) Overall training performance of the ADR policies: Success rate over the entire training range (left) and total number of expanded parameter boundaries.



(b) Training progress of individual ADR parameters. Each plot for the physical parameters has two curves indicating the upper and lower bound of the expanded training range. Dashed lines indicate fixed parameter boundaries where we do not intent to expand.

Figure C.5: Training curves for the ADR policies.

Table C.2: Simulation parameters in Automatic Domain Randomization

	Initial Value	$+\Delta$	$-\Delta$	Final Range
Object size x (m)	0.15	0.01	-0.01	[0.14, 0.16]
Object size z (m)	0.05	0.01	-0.01	[0.04, 0.06]
Table friction	0.3	0.1	-0.1	[0.1, 0.5]
Gripper friction	3	/	-1	[2, 3]
Object Density (g/m^3)	86	86	43	[43, 172]
Action translation scale (m)	0.03	/	-0.005	[0.02, 0.03]
Action rotation scale (rad)	0.2	/	-0.05	[0.1, 0.2]
Initial distance to wall (m)	0	0.01	/	[0, 0.02]
Table offset x (m)	0.5	0.01	-0.01	[0.48, 0.52]
Table offset z (m)	0.07	0.01	0.01	[0.055, 0.075]

C.4 Real robot experiment

In this section, we include more details and discussion for the real robot experiments. Quantitative results can be found on the website ¹ where we include all the videos for the real robot experiments, video examples of failure cases, recovery behaviors and ICP results.

C.4.1 Implementation details

The robot setup is shown in Figure C.6. The code for controlling the real robot is built on top of FrankaPy [148]. The policies are trained in the simulator and zero-shot transferred to a physical Franka Emika Panda robot. For the real robot experiments, we train a policy in the XZ plane from the side view to reduce the sim2real gap of the policy, since the motion is mostly in the XZ plane for the side grasp.

Sim2Real gap of the low-level controller: We observe a noticeable sim2real gap on the low-level controller when deploying the policy. The same command of moving the end-effector to a certain pose in free space may not have the same resulting movement. This is a combination of two factors: First, there is a significant discrepancy between the robot model in simulation and the real robot. The real robot has more damping and friction on the joints. Second, we use a compliant controller for this task, which is more susceptible to the noise in the system. As a result, the real robot

¹<https://sites.google.com/view/grasp-ungraspable>

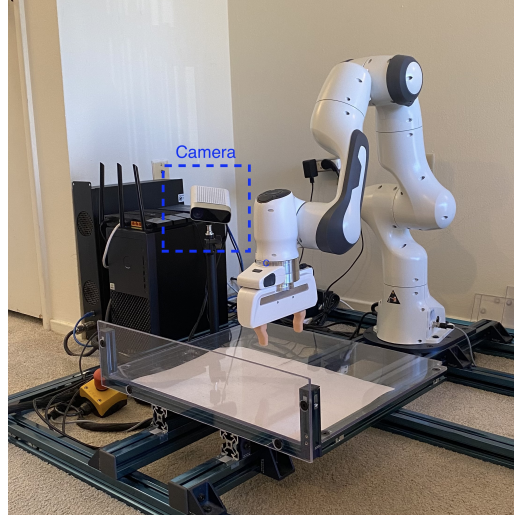


Figure C.6: Robot setup. We use one Azure Kinect camera for object pose estimation.

always executes a smaller delta movement than the simulator. To compensate for this sim2real gap, we slightly increase the action scale and reduce the policy execution rate from 2Hz to 1Hz. Both of these changes will allow the real robot to compensate for the smaller movement caused by the damping and friction of its joints. Note that the sim2real gap still exists after these changes. However, the remaining gap could be further compensated by using a closed-loop policy. During our experiments, we first use the default object Box-0 to tune the controller until we observe several successes in a row. After that, we keep the same controller setting for the entire evaluation process.

Defining object pose: The pose of a box can simply be defined at the center of its volume and with the axes defined parallel to the edges. For non-box object, we define the pose to be the center of its bounding box. We scan the non-box objects into point clouds with the Qlone app on the phone (Figure C.7 top row). To obtain the bounding box, we first run Principle Component Analysis of the scanned object to get the principle axes. Then, we take the min and max values along the axes to get the dimension of the bounding box. The axes are then aligned to global axes based on the initial pose (Figure C.7 middle row).

Pose estimation with Iterative Closest Point: To get the pose of the object as the observation of the policy, we use Iterative Closest Point (ICP) which matches

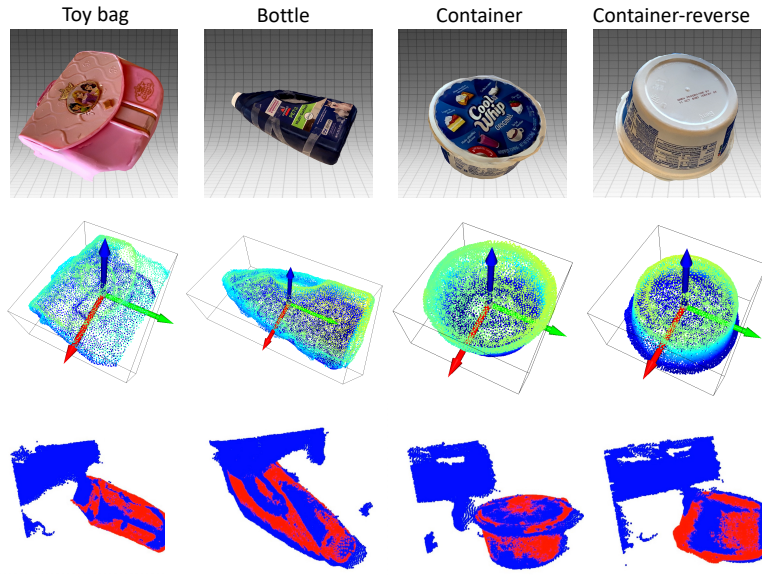


Figure C.7: Illustrations of pose estimation pipeline for the non-box objects. The top row shows the scanned object model. The middle row shows bounding box calculation and pose definition. The last row shows an example of ICP.

the current point cloud to a template point cloud of the object [104]. We use the implementation from Open3D. For box objects, we simply create a box shape template with measured size. For non-box objects, we use the scanned object point clouds as mentioned above. Figure C.8 shows an example of the results from ICP across an episode. Figure C.7 includes examples of ICP results for non-box objects potentially with partial point cloud. More visualizations of ICP results can be found on the website.

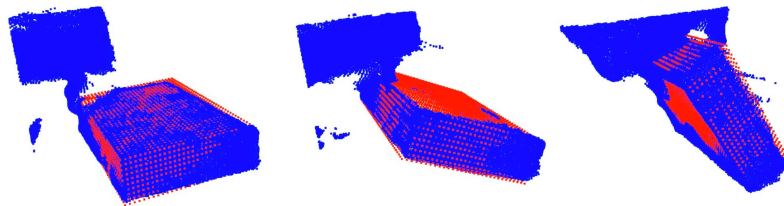


Figure C.8: Examples of pose estimation with ICP during an episode. The blue points are the observed point cloud from the camera. The red points are the object template that matches to the observed point cloud using ICP.

Table C.3: Real robot evaluations with more object information. We highlight the out-of-distribution aspect of the object properties in bold.

Object-ID	Box?	Surface Material	Bounding Box Dimension (cm)	Weight (g)	Success w/o ADR	Success w/ ADR
Box-0	Yes	Cardboard	(15.0, 20.0, 5.0)	128	9/10	9/10
Box-0 + 4 erasers	Yes	Cardboard	(15.0, 20.0, 5.0)	237	6/10	10/10
Box-0 + 8 erasers	Yes	Cardboard	(15.0, 20.0, 5.0)	345	3/10	4/10
Box-1	Yes	Cardboard	(15.4, 29.2, 5.8)	130	5/10	8/10
Box-2	Yes	Cardboard	(15.3, 22.2, 7.4)	113	2/10	9/10
Box-3	Yes	Cardboard with tape	(16.5, 24.5, 5.2)	50	0/10	7/10
Toy Bag	Almost	Silicone	(16.6, 14.5, 7.1)	203	8/10	7/10
Bottle	No	Plastic	(16.3, 28.8, 9.0)	112	0/10	8/10
Container	No	Plastic	(14.7, 14.7, 8.1)	61	0/10	10/10
Container-reverse	No	Plastic	(14.7, 14.7, 8.1)	61	0/10	6/10
Average					33%	78%

C.4.2 More information on the objects

To emphasize the diversity of the objects and demonstrate the generalization capability of the policy, we include more descriptions on the objects in this section. In Table C.3, we highlight the object properties that are out of the ADR training distribution in bold. Box-0 is the default object that we used to calibrate the simulator and to tune the low-level controller.

The policy trained with ADR generalizes across physical properties such as weight and surface friction. We stress test Box-0 with additional weights by putting four or eight erasers inside of the box. The erasers can move in the box during execution, which is not modeled in simulation. Although we do not have access to the true friction coefficient between the object and the table, the difference in friction results in qualitatively different behavior of the object even among the cardboard boxes. For example, Box-3 has tape on its surface which has much higher friction than the others cardboard boxes. It tends to stick to the wall during execution. The toy bag has a similar cross section as the box but the material is very different.

We also evaluate the policies with objects that are not similar to a box shape including a bottle and a container. Due to the difference in shape, both objects result in different dynamics during execution. In addition, with the same container object, starting it from different initial poses will also lead to different object pose distribution. Videos can be found on the website. Nonetheless, the policy trained with ADR shows reasonable generalization across these non-box objects.

C.4.3 Failure cases

In this section, we include discussions on the failure cases of the evaluation. We categorize the failure cases into the following categories and discuss the potential reasons:

A failure case that happens before the initial contact:

- **Missing initial contact:** The robot is not able to reach the initial contact of the object to rotate it. This is mostly due to the noise in pose estimation and the variations in object dimension.

Failure cases that happen during the rotation:

- **Object drops during rotation:** The object drops to the table during rotation. One potential reason for this failure case is that the finger slips on the object during rotation. Another potential reason for this failure case is the insufficient rotation of the low-level controller due to the sim2real gap (See Section C.4 - Sim2Real gap of the low-level controller). In the “dropping” strategy, the policy is supposed to rotate object and then let it drop on the bottom finger. Before the dropping happens, the gripper needs to be rotated until the bottom finger is below the object. Otherwise, the bottom finger will not be able to catch the object and the object directly drops to the table.
- **Repeated rotation:** The robot repeatedly rotates and drops the object. This is different from the previous failure case because the robot moves down with the object at the same time. Our hypothesis for this failure case is that the policy gets stuck in a loop in the MDP.
- **Joint limit:** The robot hits a joint limit and the policy gets stuck at the joint limit.

Failure cases that happen after the rotation:

- **Unexpected object dynamics:** When the robot rotates the object, the object might move in unexpected ways. This mostly happens for the non-box objects.
- **Stop reaching:** Following the “standing” strategy, the policy successfully rotates the object to a stable pose on the side of the object. However, it cannot reach the final grasping pose. The gripper tries to move down to reach the pose but it collides with the object due to the unexpected object dimension.

- **Timeout:** Since we use a fixed episode length during evaluation, sometimes the policy does not have enough time to finish the task although it is very close to a success. This happens when the policy spends time to recover from some failed attempts at the beginning of the episode.

Videos on these failure cases can be found on the website. We summarize the counts of the failure cases in Table C.4 and Table C.5. The most common failure case for the policy trained with ADR is the repeated rotation. For the policy trained without ADR, the most common failure case is missing the initial contact. Comparing the percentage of the failure cases between Table C.4 and Table C.5, we observe that percentage of missing the initial contact and the percentage of stopping the reaching motion drops drastically when the policy is trained with ADR because the policy is more robust to variations in shape and dimensions.

Table C.4: Failure cases for **Policy w/ ADR** during real robot evaluation. The most common failures include dropping the object during rotation, repeated rotation, and unexpected object dynamics.

	Initial contact	Object drops	Repeated rotation	Joint limit	Unexpected dynamics	Stop reaching	Timeout
Box-0	0	0	1	0	0	0	0
Box-0 + 4 erasers	0	0	0	0	0	0	0
Box-0 + 8 erasers	0	3	3	0	0	0	0
Box-1	0	1	1	0	0	0	0
Box-2	0	0	0	1	0	0	0
Box-3	0	0	2	0	0	0	0
Toy Bag	0	2	1	0	0	0	0
Bottle	1	0	0	0	1	0	0
Container	0	0	0	0	0	0	0
Container-reverse	0	0	0	0	4	0	0
Total	1	6	8	1	5	0	0
Percentage	4.8%	28.5%	38.1%	4.8%	23.8%	0.0%	0.0%

Table C.5: Failure cases for **Policy w/o ADR** during real robot evaluation. The most common failures include missing the initial contact, repeated rotation and unexpected object dynamics.

	Initial contact	Object drops	Repeated rotation	Joint limit	Unexpected dynamics	Stop reaching	Timeout
Box-0	0	0	0	0	0	0	1
Box-0 + 4 erasers	1	0	2	0	0	0	1
Box-0 + 8 erasers	0	1	3	0	0	2	1
Box-1	2	0	2	0	1	0	0
Box-2	1	0	1	0	0	6	0
Box-3	10	0	0	0	0	0	0
Toy Bag	0	2	0	0	0	0	1
Bottle	6	0	0	0	4	0	0
Container	0	0	2	0	8	0	0
Container-reverse	1	1	6	0	1	0	1
Total	21	4	16	0	14	8	5
Percentage	30.9%	5.9%	23.5%	0.0%	20.6%	11.8%	7.3%

Appendix D

Appendix to HACMan (Chapter 5)

D.1 Simulation Environment

D.1.1 Object dataset preprocessing

We use the object models from Liu et al. [72]. Before importing the object models to MuJoco, we perform convex decomposition using V-HACD (<https://github.com/kmammou/v-hacd>) and generate watertight meshes using Manifold (<https://github.com/hjwdzh/Manifold>). The objects are first scaled to 10 cm according to the maximum lengths along x, y, and z axis. The object sizes are randomized with an additional scale within $[0.8, 1.2]$ for the “All Objects” task variants.

We filter out a part of the objects in the original dataset due to simulation artifacts such as wall penetration and unstable contact behaviors. For example, some of the long and thin objects can be pushed into the walls and bounce back like springs. Some of the objects cannot remain stable on the table. The filtering procedure proceeds as follows: 1) we drop an object with an arbitrary quaternion and translation for 100 times; 2) we calculate the percentage of rollouts where the objects are still unstable after 80 simulation steps; 3) we filter out objects with larger than 10% instability rate. We also filter out flat objects because they are hard to flip. Flat objects are defined as objects for which the ratio between the second smallest dimension to the smallest dimension is larger than 1.5. After filtering, we are left with 44 objects. We split the 44 objects into three datasets: train (32 objects), unseen instances (7 objects), and

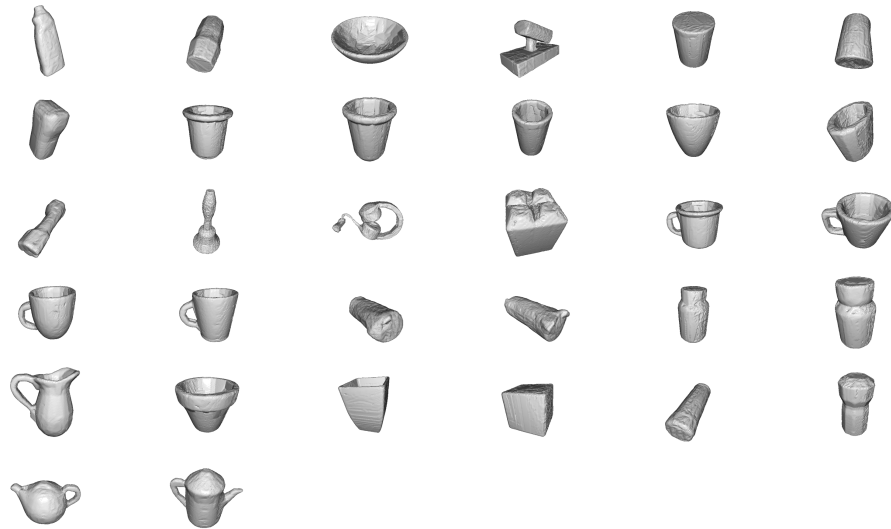


Figure D.1: **Training objects.** 32 objects used in training.

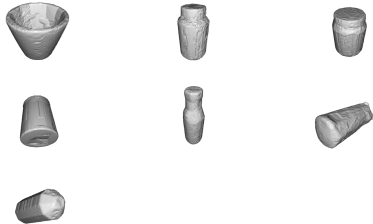


Figure D.2: **Evaluation objects (unseen instance).** 7 objects used in unseen instance evaluations. These instances are from the same categories as the training objects.

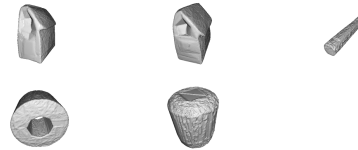


Figure D.3: **Evaluation objects (unseen category).** 5 objects used in unseen category evaluations. They come from 4 randomly chosen categories.

unseen categories (5 objects). The object models of the three datasets are visualized in Fig. D.1, Fig. D.2 and Fig. D.3 respectively. The **Cylindrical Objects** used in the experiments is a subset of the **All Objects** dataset. **Cylindrical Objects** consist of 9 train objects, 3 unseen instance objects, and 4 unseen category objects.

D.1.2 Collecting goal poses

To collect stable goal poses, we sample an SE(3) object pose in the air above the center of bin, drop the object in the bin, and then wait until it becomes stable to record the pose. We collect 100 goal poses for each object. At the beginning of each episode, a goal is sampled from the list of stable poses. Furthermore, we randomize the location of the sampled stable goal pose within the bin.

D.1.3 Representing the goal as per-point goal flow

As mentioned in Section 5.5.3, we represent the goal as the “goal flow” of each object point from the current point cloud to the corresponding point in the transformed goal point cloud. In other words, suppose that point x_i in the initial point cloud corresponds to point x'_i in the goal point cloud; then the goal flow is given by $\Delta x_i = x'_i - x_i$. The goal flow Δx_i is a 3D vector which is concatenated to the other features of the input point cloud to represent the goal. In the ablations in Appendix D.3.1, we show that such a representation of the goal significantly improves training, compared to other goal representations such as concatenating the goal point cloud with the observed point cloud.

In order to compute the flow to the goal, we need to estimate correspondences between the observation and the goal. In simulation, we calculate the goal flow based on the ground truth correspondences, based on the known object pose and goal pose. In the real robot experiments, we estimate the correspondences using point cloud registration methods (see Appendix D.4 for details).

Further, for training the RL algorithm, we need some measure of the distance between the initial pose and the goal pose as the reward. Rather than computing a weighted average of the translation and rotation distance (which requires a weighting hyperparameter), we instead define the reward at each timestep r_t as the negative of the average goal flow: $r_t = -\frac{1}{N} \sum_{i=1}^N \|\Delta x_i\|$, in which $\|\cdot\|$ denotes the L2 distance

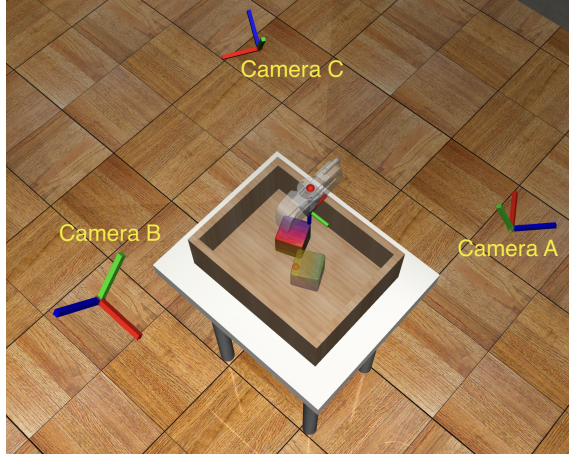


Figure D.4: Camera locations in simulation.

and Δx_i is the “goal flow” as defined above. This computation is similar to the “matching score” [40] or “PLoss” [133] used in previous work, except here we use it as a reward function.

D.1.4 Success rate definition

An episode is marked as a success when the average distance of the corresponding points between the object and the goal is smaller than 3 cm. More specifically, this is calculated by the average norm of the per-point goal flow vectors as described in Appendix D.1.3. The episode terminates when it reaches a success. If the episode does not reach a success within 10 steps, it is marked as a failure. We include an additional experiment on longer episode length in Appendix D.3.7.

D.1.5 Observation

The observation space includes a point cloud of the entire scene \mathcal{X} . It contains background points \mathcal{X}^b and object points \mathcal{X}^{obj} . Note that we move the gripper to a reset pose after every action before taking the next observation. Thus, the gripper is not observed in the point cloud. To get the point cloud, we set three cameras around the bin (Fig. D.4). The depth readings from the cameras are converted to a set of point locations in the robot base frame and combined.

The object points are then downsampled with a voxel size of $0.005 \text{ m} \times 0.005 \text{ m} \times 0.005 \text{ m}$ and the background points are downsampled with a voxel size of $0.02 \text{ m} \times 0.02 \text{ m} \times 0.02 \text{ m}$. We empirically find that using a slightly denser object point cloud may increase the performance. More specifically, using a $0.005 \text{ m} \times 0.005 \text{ m} \times 0.005 \text{ m}$ voxel downsample is slightly better than $0.01 \text{ m} \times 0.01 \text{ m} \times 0.01 \text{ m}$. We suspect that the policy can perform more precise manipulation of the object with a denser point cloud.

After downsampling, we estimate the normals of the object points using Open3D. The estimated normals will be used during action execution (discussed in the next section).

As mentioned in Section 5.5.3 and Appendix D.1.3, the feature of each point contains the goal flow and the segmentation mask (foreground vs background). The goal flow of the object point is calculated according to Section D.1.3. The goal flow of the background point is set to zero. We obtain the segmentation labels of the object points and the background points from Robosuite[154] during simulation. Details of obtaining segmentation labels in real robot experiments are discussed in Appendix D.4.

D.1.6 Action

As mentioned in Section 2.4, the proposed method uses an action space with a contact location selected from the object points and a set of motion parameters. We discuss the implementation details of executing such an action in the simulation environment in this section. Note that we use a floating gripper as the robot in simulation since we only focus on gripper interactions with the objects.

Once the policy selects a point on the object point cloud, we obtain the corresponding location and estimated normal of the point as described in the previous section. The robot first moves to a “pre-contact” location which is 2 cm away from the contact location along the surface normal. In simulation, this is implemented by directly setting the gripper to the desired pose. In real experiments, we adopt a workaround solution discussed in Appendix D.4. If the gripper encounters a collision at the desired pose, we mark this action as failure and skip the remaining action execution procedure. After reaching the pre-contact location, the gripper will approach to the

desired contact location using a low-level controller.

After that, the robot will execute the motion parameters which is the end-effector delta position command that was output by the policy. For the delta actions, we use an action scale of 2 cm. The delta action is executed with an action repeat of 3. We use Operation Space Controller with relatively low gains to allow compliant contact-rich motions with the object. Note that we only consider translation commands (3 dimensions) without rotation in the main experiments because it leads to sufficiently complex object motion for our task. Appendix D.3.4 discusses the effect of including rotation in the gripper movements.

The gripper may not exactly reach the desired location in both sim and real, due to the compliant low-level controller and the gripper geometry. We consider this imperfect execution as a part of the environment dynamics. We do not enforce assumptions such as keeping the contact while executing the motion parameter or avoiding other contact points. Avoiding such assumptions on contacts is a strength of the proposed method compared to some of the classical methods [16, 43].

D.2 Algorithm and Training Details

D.2.1 HACMan (Ours)

HACMan is implemented as a modification on top of TD3 [30] based on the implementation from Stable-Baselines3 (<https://github.com/DLR-RM/stable-baselines3>). We use PointNet++ segmentation-style backbones for both the actor and the critic using the implementation from PyG (<https://pytorch-geometric.readthedocs.io>). Weights are not shared between the actor and the critic. Hyperparameters are included in Table D.1. The actor and the critic use the same network size and the same learning rate. To improve the stability of policy training, we clamp the target Q-values according to an estimated upper and lower bound of the return for the task. The location policy temperature β is described in Eqn. 5.4.

Table D.1: Hyperparameters.

Hyperparameters	Values
Initial timesteps	10000
Batch size	64
Discount factor (γ)	0.99
Critic update freq per env step	2
Actor update freq per env step	0.5
Target update freq per env step	0.5
Learning rate	0.0001
MLP size	[128, 128, 128]
Critic clamping	[-20, 0]
Location policy temperature (β)	0.1

D.2.2 Baselines

The baselines share the same code framework as HACMan. We discuss their differences with HACMan in this section.

Regress Contact Location. Unlike HACMan, this baseline does not use the object surface for contact point selection. Instead, it directly predicts a location (3 dimensions) and a motion parameter (3 dimensions, represented as a delta end-effector movement). For each action execution, the end-effector moves to the selected location, moves according to the motion parameters, and then resets to the default pose. To improve the performance of this baseline, we project the contact location output to be within the bounding box of the object. Thus, in this baseline, for a location output of the policy, a value of 0 corresponds to the center of the object along a specific dimension, while 1 and -1 represent the maximum and minimum boundaries of the bounding box along that dimension, respectively. Since the location output is no longer a point selected from the object surface, we can no longer use the surface normal vector to determine the approach direction as in HACMan. Instead, this baseline always approaches the location from the top at a height equal to the maximum side length of the object bounding box.

No Contact Location. This baseline does not use the idea of a contact point. Instead, the policy only predicts a motion parameter (3 dimensions, represented as a delta end-effector movement). For each action execution, the end-effector moves according to the motion parameter starting from where it ends after the previous action, without resetting to the default pose. To reduce the exploration difficulties,

we make two additional changes: 1) we always start the end-effector right above the object (at a height equal to the maximum side length of the object bounding box) at the beginning of an episode, and 2) we add an extra term to the reward function that penalizes the end-effector for being too far from the object,

$$J_{\text{dist}} = \begin{cases} -\lambda_{\text{dist}}(d_{\min} - 0.05), & d_{\min} > 0.05 \text{ m} \\ 0, & \text{otherwise} \end{cases} \quad (\text{D.1})$$

where d_{\min} is the minimum distance from the end-effector to the object point cloud vertices, and λ_{dist} is the weight for this reward term.

Point Cloud. Unlike HACMan, these point cloud baselines use PointNet++ classification-style backbones from PyG (<https://pytorch-geometric.readthedocs.io>). For each point cloud, it extracts a single global feature vector instead of per-point feature.

State. In the state-based baselines, the input consists of the pose of the current object, the goal, and optionally the end-effector if the baseline is using “No Contact Location”. Each pose is a vector (dim=7) that consists of a position (dim=3) and a quaternion (dim=4). The model concatenates all the pose vectors into a single vector as the input to an MLP.

We report the best results of the baselines by searching over different hyperparameters for each baseline, including learning rate, actor update frequency, initial timesteps, and EE distance weight λ_{dist} . The best hyperparameters for each baseline that are different from HACMan are summarized in Table D.2; any hyperparameter not listed in Table D.2 is the same as our method (Table D.1).

Table D.2: Baseline-specific Hyperparameters.

Baselines	Hyperparameters	Values
Regress Contact Location (Point Cloud)	Actor update freq per env step	0.25
No Contact Location (Point Cloud)	Actor update freq per env step	0.25
	EE Distance Weight λ_{dist}	1
No Contact Location (State)	EE Distance Weight λ_{dist}	5

D.3 Supplementary Experiment Results

D.3.1 Additional ablations

We perform additional ablation studies to analyze each component of the proposed method with all the variants of the object pose alignment task. The results of the ablations are summarized in Fig. D.5.

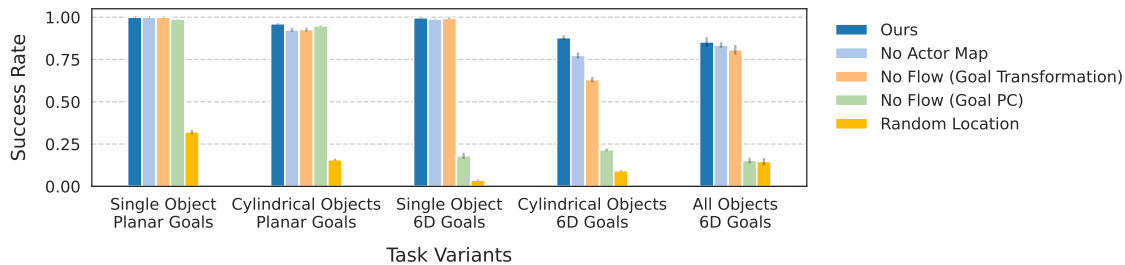


Figure D.5: **Additional ablations.** All of the components of our method are essential to achieve the best performance when the task becomes more difficult.

Effect of Contact Location: To test the hypothesis that contact location matters for non-prehensile manipulation, we design a “**Random Location**” ablation: the policy randomly selects a contact location on the object instead of learning to predict a contact location. From Fig. D.5, we observe a performance drop for not predicting the contact location even for the simplest task variant.

Effect of Goal Representations: As described in Section 5.5.3 and Appendix D.1.3, in our method, we represent the goal by first computing the correspondence between the observation and goal point clouds and concatenating a per-point “goal flow” to the observation. We include two alternative goal representations to justify the use of goal flow in our pipeline: “**No Flow (Goal PC)**” concatenates the goal point cloud with the observed point cloud [13, 14]. We use an additional segmentation label in the point features to distinguish the goal points from the observed points. From Fig. D.5, this ablation only works well on planar goals for this task. In “**No Flow (Goal Transformation)**”, we represent the goal as the transformation between the current observation pose and the goal pose. We represent this transformation as a 7D vector that includes a translation vector and a quaternion. We concatenate

the 7D goal pose to the observation at all of the object points. Note that, similar to our method, this baseline also requires computing correspondences between the observation and the goal. This approach performs well but slightly worse than our method in the last two task variants.

Effect of Actor Map: Instead of using an Actor Map which has per-point outputs, this ablation uses an actor that outputs a single vector of motion parameters while keeping the Critic Map. This is different from the baselines in the previous section that remove both the Actor and Critic Maps. In the “**No Actor Map**” experiments, we observe a relatively minor performance drop compared to the full method. Nonetheless, using the per-point action output from an Actor Map instead of a single output may allow the agent to reason more effectively about different actions for different contact locations, such as the multimodal solution shown in Fig. 5.6 (middle).

D.3.2 Training curves and tables

In this section, we include the full training results for all the methods with additional task variants. Fig. D.6 and Fig. D.7 include the training curves for the baselines and the ablations. Table D.3 and Table D.4 are recorded at 200k environment interaction steps from the training curves for all the methods. The numbers in the tables are used to generate the bar plots in Fig. 5.4 and Fig. D.5.

Note we also interpolate between the tasks ”Planar Goals” and ”6D Goals” and include an additional task configuration with a fixed initial object pose and a randomized 6D goal, “6D Goals (Fixed Init)”. This task configuration is combined with the Single Object dataset and the Cylindrical Object dataset. Thus, we include 7 variants in total (5 variants in the main text).

As discussed in Section 5.7, the baselines and ablations have poor performance when the task becomes more challenging. Our method achieves the best converged performance across all task variants while being more sample efficient.

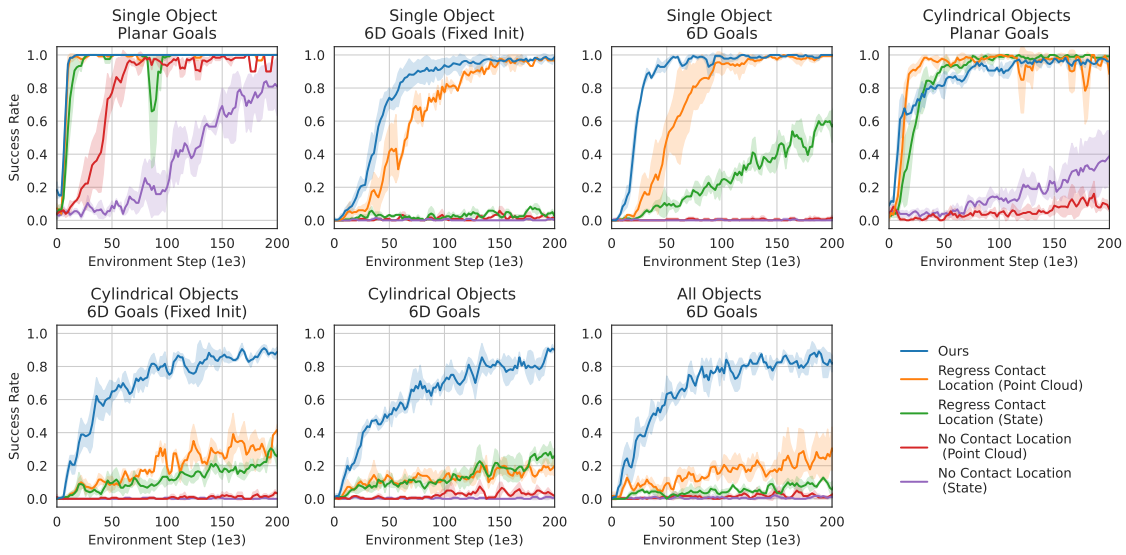


Figure D.6: **Baselines.** It shows success rates on the train dataset over environment steps. The shaded area represents the standard deviation across three training seeds.

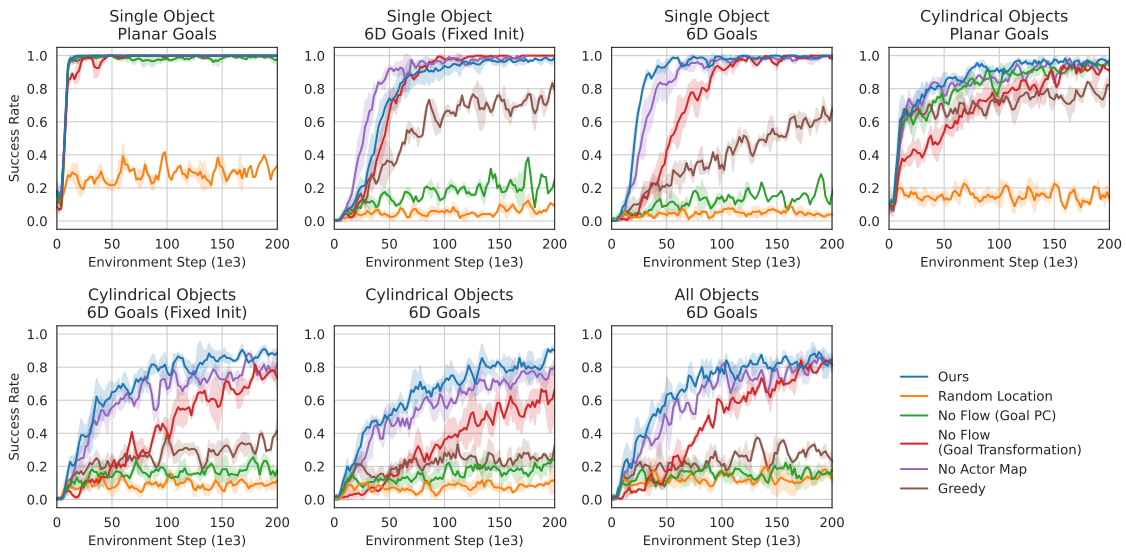


Figure D.7: **Ablations.** It shows success rates on the train dataset over environment steps. The shaded area represents the standard deviation across three training seeds.

Table D.3: **Baselines.** We compare our method with baselines with different action representations and observations. Our approach outperforms the baselines, with a larger margin for more challenging tasks. The success rate is reported with the mean and standard deviation across three seeds.

Object Dataset	Task Configuration	Methods				
		No Contact State	Location Point Cloud	Regress State	Contact Location Point Cloud	Ours
Single Object	Planar Goal	0.812 ± .012	0.973 ± .016	1.000 ± .000	0.996 ± .005	1.000 ± .000
	6D Goal (Fixed Init)	0.003 ± .000	0.020 ± .002	0.060 ± .014	0.971 ± .005	0.982 ± .004
	6D Goal	0.000 ± .000	0.009 ± .001	0.573 ± .015	0.991 ± .004	0.997 ± .003
Cylindrical Objects	Planar Goal	0.361 ± .019	0.107 ± .007	0.990 ± .002	0.924 ± .027	0.961 ± .003
	6D Goal (Fixed Init)	0.001 ± .001	0.021 ± .002	0.264 ± .017	0.324 ± .014	0.885 ± .004
	6D Goal	0.006 ± .002	0.035 ± .002	0.258 ± .010	0.187 ± .012	0.879 ± .014
All Objects	6D Goal	0.012 ± .004	0.016 ± .009	0.094 ± .018	0.243 ± .028	0.854 ± .028

Table D.4: **Ablations.** We show that all of the components are essential to achieve the best performance when the task becomes more difficult. Each success rate is reported with the mean and standard deviation across three seeds.

Object Dataset	Task Configuration	Methods					Ours
		Random Location	Greedy	No Flow (Goal PC)	No Flow (Goal Pose)	No Action Map	
Single Object	Planar Goal	0.323 ± .011	1.000 ± .000	0.989 ± .002	1.000 ± .000	1.000 ± .000	1.000 ± .000
	6D Goal (Fixed Init)	0.075 ± .005	0.754 ± .023	0.198 ± .025	1.000 ± .000	0.991 ± .002	0.982 ± .004
	6D Goal	0.037 ± .003	0.633 ± .014	0.181 ± .018	0.994 ± .004	0.989 ± .002	0.997 ± .003
Cylindrical Objects	Planar Goal	0.158 ± .006	0.767 ± .017	0.949 ± .003	0.927 ± .012	0.925 ± .012	0.961 ± .003
	6D Goal (Fixed Init)	0.097 ± .006	0.346 ± .012	0.189 ± .009	0.746 ± .015	0.805 ± .016	0.885 ± .004
	6D Goal	0.093 ± .004	0.262 ± .011	0.216 ± .008	0.631 ± .016	0.775 ± .018	0.879 ± .014
All Objects	6D Goal	0.147 ± .021	0.293 ± .026	0.153 ± .017	0.808 ± .028	0.835 ± .017	0.854 ± .028

D.3.3 Additional baseline: Global feature with query contact location

We consider an additional baseline in this section where both the actor and the critic use a global point cloud feature and a query contact location as input. The query contact location is represented as a 3D coordinate (x, y, z) . More specifically, the actor takes as input a global feature and a contact location and outputs a continuous vector of motion parameters. The critic takes as input a global feature and a contact location and outputs a Q-value. Since both the actor and the critic require a contact location as input, we still need a way of selecting the query contact location. We follow a similar way as our method to use the observed points on the object as candidate queries and select the contact location based on the highest Q-value. In this way, the action space remains a discrete-continuous action space as our method, but it uses a global point cloud feature rather than a segmentation-style per-point feature.

As shown in Fig. D.8, this alternative baseline performs worse than our method. We hypothesize that a segmentation-style point cloud network can reason about the local point features more effectively than a global feature extractor due to skip connections.

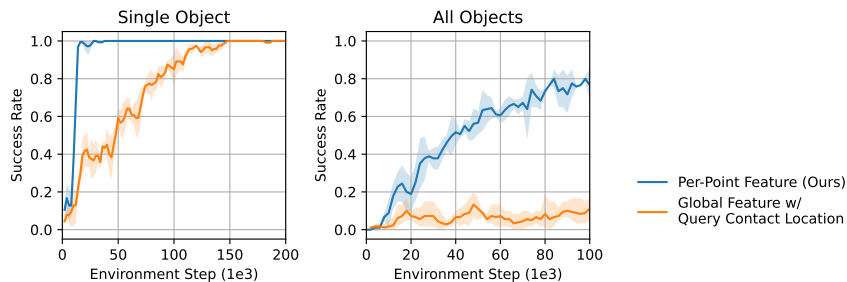


Figure D.8: **Comparison between our method and the additional baseline with query contact locations.** The left figure shows the success rate of the simplest task variant - a single object with planar goals. The right figure shows the most challenging task variant - all objects with 6D goals. The shaded area represents the standard deviation across three training seeds. Our method performs better than the baseline in both cases.

D.3.4 Extending Motion Parameters

The motion parameters in the main results are defined as a 3D vector that describes the translation motion of the gripper. In this section, we extend the motion parameters in different ways:

6D Contact. The motion parameters also predict the orientation of the gripper when the gripper approaches the contact location. The orientation is in the form of ZYX Euler angles. To account for the physical constraints of our task setup, we restrict the y and x angles to the range of $[-0.5\pi, 0.5\pi]$, and the z angle to the range of $[-\pi, \pi]$.

6D Motion. We introduce the ability for the gripper to change orientation while executing the motions after making contact. Similar to the translation motion parameters, the rotation motion parameters (ZYX Euler angles) represent the delta rotation at each action repeat step.

Per-point Contact Location Offset. We conduct an experiment where the agent learns a per-point contact location offset combined with 3D motion. The agent’s continuous action space is defined as (contact_offset, 3D motion parameters). For each action on a given point at location x , the agent uses $(x + x_{\text{offset}})$ as the contact location. Notably, the x_{offset} value is mapped to scale with the bounding box since it ranges between $[-1, 1]$.

Table D.5 compares the performance of HACMan with the modified action spaces. The success rates are reported along with their corresponding standard deviations. We find that including 6D motion in the motion parameters results in a slightly higher success rate. However, 6D contact or contact offset does not seem to provide significant benefits. We also try to include 6D motion in the **Regress Contact Location** baseline. As shown in Table D.5, 6D motion improves the performance of the baseline, but the success rate is still much worse than our method.

Table D.5: **Success rates with different motion parameters.** All methods are evaluated on all train objects with 6D goals.

Method	Success Rate
HACMan Default	$0.833 \pm .018$
+ with 6D Motion	$0.866 \pm .090$
+ with 6D Contact	$0.819 \pm .077$
+ with Contact Offset	$0.800 \pm .011$
Regress Contact Location Default	$0.243 \pm .028$
+ with 6D Motion	$0.356 \pm .133$

D.3.5 Experiments in cluttered environments

We can directly apply HACMan to a setting of manipulating objects in cluttered scenes. We conduct preliminary experiments in which we introduce varying numbers of scene objects into the bin. The scene objects serve as obstacles that add challenges to the task. We train HACMan under two conditions: **with one scene object** and **with five scene objects**, and we compare the results with the performance achieved in the absence of any scene objects (default setting). From Table D.6, as expected, the task becomes more challenging when there are more obstacles in the bin. As illustrated in Fig D.9, the policy tends to push the object directly toward the goal by pushing the scene object aside.

Table D.6: **Success rates under different cluttered scenes.** All methods are evaluated with 6D goals.

# of Scene Objects	Success Rate
0 (Default)	0.833
1	0.773
5	0.580



Figure D.9: **Qualitative results for object pose alignment tasks in cluttered environments.** HACMan shows complex non-prehensile behaviors that move objects to goal poses (shown as the transparent objects). The scene objects are colored in brown to distinguish from the target object to be manipulated to the goal pose.

D.3.6 Effect of longer training time

Although we report the success rate at 200k training steps for all the results due to computational limitations, our method continues to improve performance with longer training (Figure D.10). The graph illustrates the success rate achieved by our method as the number of training steps increases. Notably, after 500k training steps,

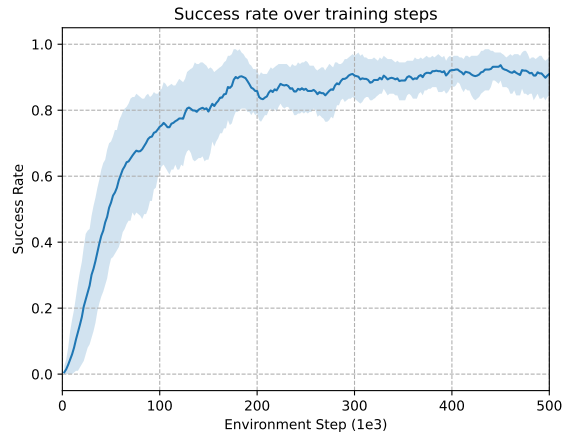


Figure D.10: **Success rate with extended training.** The success rate of our method reaches $91.1 \pm 7.3\%$ after 500k training steps, compared to 83.3% after 200k training steps.

our method achieves a success rate of $91.1 \pm 7.3\%$, significantly improving from the 83.3% success rate reported in the main text (at 200k training steps).

D.3.7 Effect of longer episode lengths

We conducted an additional experiment to explore the relationship between success rates and maximum episode length. In the main text, our episodes were limited to a maximum of 10 steps, and any episode exceeding this limit was deemed a failure. During this additional evaluation, we relaxed the episode length restrictions and allowed the agent to operate with a maximum episode length of 30. As shown in Fig D.11, HACMan achieves more than 95% success rates across all datasets (Train 96.6%, Train (Common) 99.4%, Unseen Instance (Common) 99.7%, Unseen Category 95.1%) when the maximum episode length is extended to 30. This suggests that providing the agent with a longer time horizon enables it to achieve higher success rates without the need for retraining.

D.3.8 Per-category result breakdown

Fig. D.12 shows the breakdown of the results for each object category. Although our method demonstrates consistent performance across the majority of objects, there

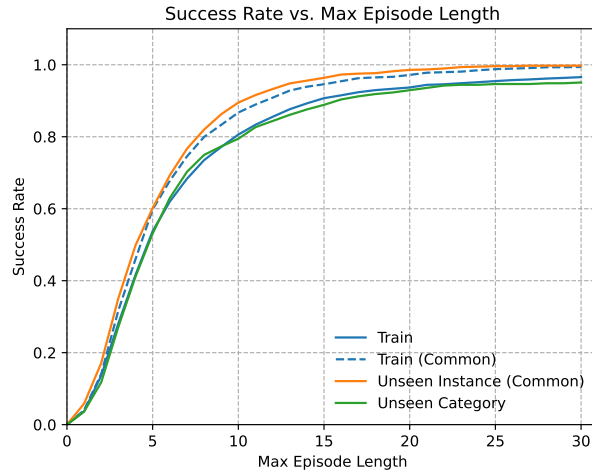


Figure D.11: **Success rates at various maximum episode lengths.** This line plot shows the success rates of HACMan evaluated on the four datasets. It is worth noting that the success rates for Unseen Instance (Common) and Train (Common) are marginally higher compared to Train and Unseen Category, similar to the pattern in Table 5.2.

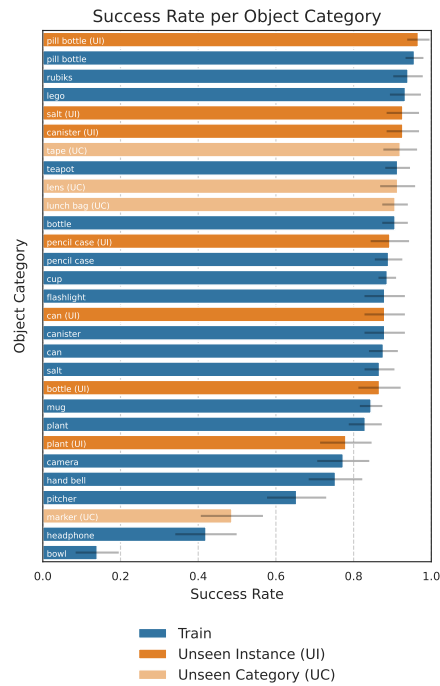


Figure D.12: **Results breakdown.** Object categories in the unseen instance set (orange) can be compared to the same object categories in the train set (blue) to see the level of instance generalization.

are certain objects with geometries that pose intrinsic challenges for our approach. For example, our method is limited to poking a bowl from the top due to occlusions, making it difficult to flip an upward-facing bowl downwards.

D.3.9 Final Distance to Goal

To further analyze the performance of our method, Fig. D.13 visualizes the distribution of distances to goal of our method at the end of the episodes for the “All Objects 6D Goals” task variant. These distances are computed as the norms of the flow distances between the objects and their respective goals.

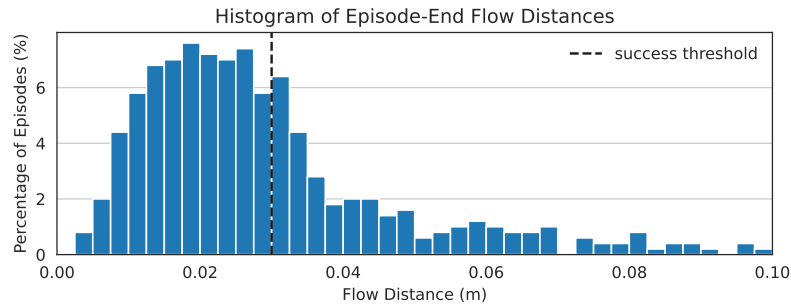


Figure D.13: **Distribution of distances to the goal at the end of the episode for our method in the “All Objects 6D Goals” experiment.** The vertical dashed line represents the success threshold at a distance of 0.03m. The distribution has a median of 2.57cm, a mean of 3.66cm, and a standard deviation of 4.27cm.

D.4 Real Robot Experiments

D.4.1 Real robot setup

The robot setup is shown in Fig. D.14. We use three cameras on the real robot to get a combined point cloud. We follow a similar procedure as Appendix D.1 to process the point cloud and to execute the action except the following details: We segment the object points from the full point cloud based on the location and the dimension of the bin instead of using the ground truth segmentation labels from Robosuite. To move the gripper to the pre-contact location, we first move the robot

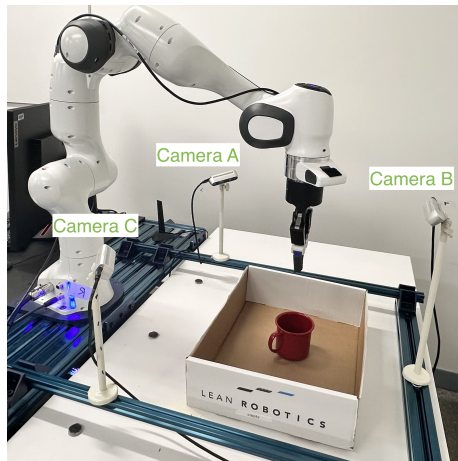


Figure D.14: Real robot setup.

to a location above the pre-contact location and then move down to the pre-contact location, instead of “teleporting” the gripper in simulation.

To obtain goals for the real world evaluation, we record 10 goal point clouds for each object by manually setting the objects into different stable poses. During each timestep, we use point cloud registration algorithm to estimate the goal transformation to calculate the goal flow. Specifically, we use the global registration implementation from Open3D (http://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html) and then use Iterative Closest Point (ICP) for local refinement. Note that we only match the shapes of the object instead of matching both the colors and the shapes due to the limitation of the registration algorithms.

Note that the evaluation process can be done automatically without any manual resets. The reward and the episode termination condition (Appendix D.1.4) are both calculated automatically.

For the real robot experiments, we use the policy trained with “6D goals” and the “All Objects” dataset. We perform zero-shot sim2real transfer without finetuning or additional domain randomization. We have tried to add noise to the contact location execution and add noise to the point cloud observation. However, these modifications did not result in better real robot performance.

D.4.2 Analysis

We include additional analysis on the real robot results in this section. The proposed method assumes an estimated goal transformation as input. To estimate the transformation from the object to the goal, we use point cloud registration, as described above. However, the estimation of the transformation might not be perfect in the real world. To better understand the performance of our system, we define two types of evaluation criteria: The “flow success” is automatically calculated based on the *estimated* point cloud registration according to the evaluation metric in Appendix D.1.4. Hence, “flow success” will sometimes mark an episode as a success or failure incorrectly due to errors in the point cloud registration. For the “actual success” evaluation metric, we manually mark as failures the cases among the flow success episodes where the goal estimation is significantly wrong. Thus, “flow success” indicates the performance of the trained policy (assuming perfect point cloud registration at termination) while the “actual success” indicates the performance of the full system (accounting for errors in the point cloud registration). Fig. 5.7 in the main text reports the actual success. We include both success metrics in Table D.7 below. The policy achieves a 61% success rate based on the flow success, indicating that some of our errors are due to failures in point cloud registration.

D.4.3 Failure cases

We discuss the failure cases of the real robot experiments in this section and include the videos on our website: <https://hacman-2023.github.io/>. The most noticeable failure cases are due to the errors of point cloud registration. The challenges of the registration methods come from noisy depth readings and partial point clouds. The error of the point cloud registration methods will lead to unexpected actions during the episode. In addition, it may end the episode early because the episode termination depends on the goal estimation. This motivates us to separate out the success criteria in Table D.7 based on the failures of the registration method.

On the action side, both the contact location and the motion parameters might have execution errors. Since the contact location is selected from the observed point cloud, when the camera calibration is not accurate enough, the robot might not be able to reach the desired contact location of the object. In addition, since we use a

Table D.7: **Additional analysis on the real robot experiments.** An episode is considered a “flow success” if the average norm of the estimated flow is less than 3 cm. An episode is considered as an “actual success” if the object is aligned with the goal pose without point cloud registration failure.

Object Name	Planar Goals		Non-planar Goals		Total	
	Flow	Actual	Flow	Actual	Flow	Actual
(a) Blue cup	4/7	4/7	7/13	4/13	5/20	4/20
(b) Milk carton	6/7	6/7	10/13	10/13	16/20	16/20
(c) Box	2/5	2/5	10/15	10/15	12/20	12/20
(d) Red bottle	7/7	4/7	6/13	0/13	13/20	4/20
(e) Hook	5/8	5/8	5/12	5/12	10/20	10/20
(f) Black mug	4/7	4/7	2/13	0/13	6/20	4/10
(g) Red mug	5/7	5/7	7/13	3/13	12/20	8/20
(h) Wood block	6/7	6/7	8/13	6/13	14/20	12/20
(i) Toy bridge	9/10	9/10	7/10	5/10	16/20	14/20
(j) Toy block	2/2	2/2	10/18	10/18	12/20	12/20
Total	50/67	47/67	72/133	53/133	122/200	100/200
Percentage	75%	70%	54%	40%	61%	50%

compliant low-level controller to execute the motion parameters, the robot might not be able to execute the desired motion the same way as in the simulation.

In addition, the object dynamics might be different from the simulation due to the surface friction and the density of the object. The performance of our method could be further improved with domain randomization over the physical parameters.

D.5 Discussion on non-prehensile manipulation

Non-prehensile manipulation is an important aspect of the robot’s capabilities, particularly in scenarios where grasping encounters limitations, as demonstrated in Fig. D.15. This section discusses the importance of non-prehensile actions in two key contexts:

Environment Occlusion. The first row of Figure D.15 shows an example where the potential grasp poses of the object are occluded by the wall. Non-prehensile moves, like nudging objects to a better position, offer a practical solution in such a scenario.

Oversized Objects. The last two rows of Figure D.15 include scenarios where certain dimensions of the object are larger than the width of the gripper.

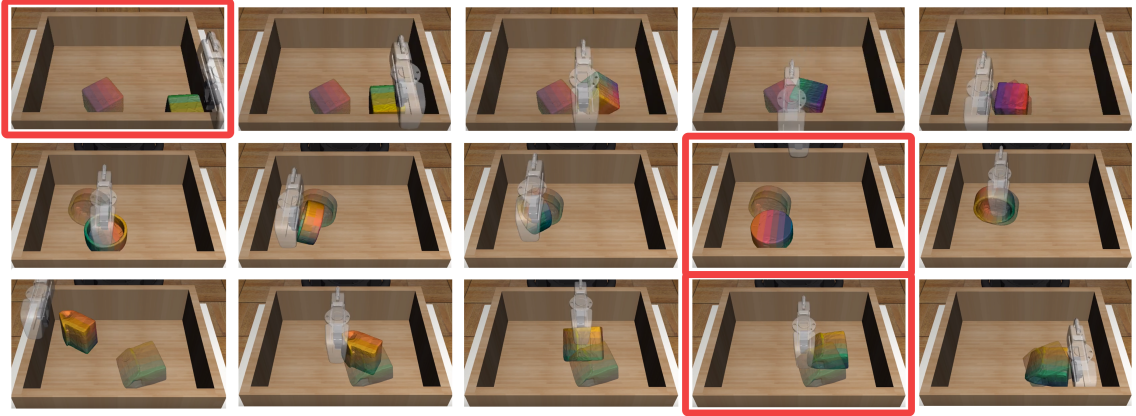


Figure D.15: **Examples showcasing limitations of prehensile manipulation.** The frames where prehensile manipulation is challenging are highlighted. The first row shows a cube placed at the corner of the bin, where any grasp is obstructed by the bin wall. Both the second and third rows depict instances where objects are too large to be grasped at specific poses.

D.6 More discussion on the related work

Compared to Chen et al. [13, 14]. Our work is substantially different from Chen et al. [13, 14] from the follow aspects: **Approach:** The approach in Chen et al. [13, 14] follows a student-teacher training pipeline. The teacher training is equivalent to the “No Contact Location” baseline with “states” observations in this work. The policy takes all the relevant robot state and object state information and outputs delta robot actions. Note that they train a single teacher policy across all shapes without using the point cloud which results in a state-observation policy that is “robust” to shapes instead of “adaptive” to shapes (see Discussion section in Chen et al. [14]). As shown in Table II, this baseline performs significantly worse than our method in our task because it lacks shape information from the point cloud and the robot-centric action space is not as efficient as our object-centric action space. On the other hand, although the student policy in Chen et al. [13, 14] takes point cloud observation, it is trained using imitation learning from the teacher, so its performance is upper bounded by the teacher policy which has been shown to be worse than our proposed method. **Task:** We investigate a completely different task and thus the numbers are not really comparable with the numbers from previous work [13, 14].

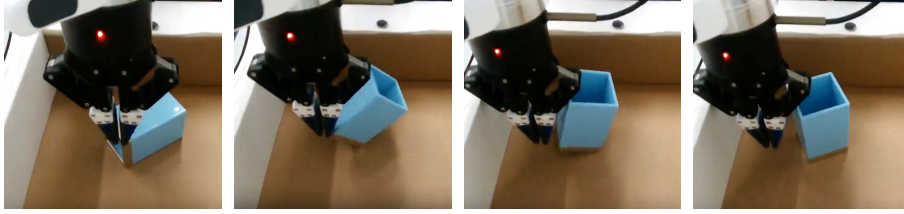


Figure D.16: **An example of non-quasi-static motion.** The figure shows an example of executing the motion parameters to flip a mug upright. After the gripper pushes against the edge of the mug (first two images), it relies on the inertia of the mug to finish the motion which is not quasi-static (last two images).

First, we use a simple gripper instead of a dexterous hand. Second, we consider matching the orientation and position of the goal pose while Chen et al. [13, 14] only considers orientation.

Compared to Cheng et al. [16], Hou and Mason [43]. Unlike Cheng et al. [16], Hou and Mason [43], our method is not limited to a simplified gripper model and does not require the knowledge of object environment contact modes which are challenging to estimate during real robot execution. In addition, Cheng et al. [16], Hou and Mason [43] are restricted by quasi-static assumptions. Although our method requires static point cloud observations in between robot actions, the robot interaction with the object is not restricted to quasi-static motion. As shown in Figure D.16, the flipping motion could be non-quasi-static. However, we also want to point out that static point cloud observations may limit the method from more complex dynamic motions.

Bibliography

- [1] Abbas Abdolmaleki, Jost Tobias Springenberg, Jonas Degraeve, Steven Bohez, Yuval Tassa, Dan Belov, Nicolas Heess, and Martin Riedmiller. Relative entropy regularized policy iteration. *arXiv preprint arXiv:1812.02256*, 2018. [3.3.2](#)
- [2] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018. [3.3.2](#), [3.4](#)
- [3] Abbas Abdolmaleki, Sandy H Huang, Giulia Vezzani, Bobak Shahriari, Jost Tobias Springenberg, Shruti Mishra, Dhruva TB, Arunkumar Byravan, Konstantinos Bousmalis, Andras Gyorgy, et al. On multi-objective policy optimization as a tool for reinforcement learning. *arXiv preprint arXiv:2106.08199*, 2021. [3.3.2](#)
- [4] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. *International Conference on Machine Learning*, 2020. [2.2](#)
- [5] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019. [3.2](#)
- [6] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019. [3.2](#)
- [7] 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. [C.2.7](#)
- [8] Michael Bloesch, Jan Humplik, Viorica Patraucean, Roland Hafner, Tuomas Haarnoja, Arunkumar Byravan, Noah Yamamoto Siegel, Saran Tunyasuvunakool, Federico Casarini, Nathan Batchelor, et al. Towards real robot learning in the wild: A case study in bipedal locomotion. In *Conference on Robot Learning*, pages 1502–1511. PMLR, 2022. [3.7.1](#)
- [9] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309,

2013. [4.2.2](#)
- [10] Lillian Y Chang, Siddhartha S Srinivasa, and Nancy S Pollard. Planning pre-grasp manipulation for transport tasks. In *2010 IEEE International Conference on Robotics and Automation*, pages 2697–2704. IEEE, 2010. [4.1](#), [4.2.2](#), [4.4.2](#)
 - [11] Nikhil Chavan-Dafle and Alberto Rodriguez. Sampling-based planning of in-hand manipulation with external pushes, 2017. [4.1](#), [4.2.1](#)
 - [12] Nikhil Chavan-Dafle, Rachel Holladay, and Alberto Rodriguez. In-hand manipulation via motion cones, 2019. [4.1](#), [4.2.1](#)
 - [13] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. *Conference on Robot Learning*, 2021. [1](#), [4.2.3](#), [5.2](#), [5.5.3](#), [5.1](#), [5.7](#), [5.7](#), [D.3.1](#), [D.6](#)
 - [14] 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. [5.2](#), [5.5.3](#), [5.1](#), [5.7](#), [5.7](#), [D.3.1](#), [D.6](#)
 - [15] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6520–6526. IEEE, 2021. [4.1](#), [4.2.1](#)
 - [16] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2730–2736. IEEE, 2022. [1](#), [4.1](#), [4.2.1](#), [5.1](#), [5.2](#), [D.1.6](#), [D.6](#)
 - [17] Xuxin Cheng, Ashish Kumar, and Deepak Pathak. Legs as manipulator: Pushing quadrupedal agility beyond locomotion, 2023. [5.11.1](#)
 - [18] Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S Srinivasa, Michael Erdmann, Matthew T Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1578–1585. IEEE, 2014. [1](#), [4.1](#), [4.2.1](#)
 - [19] Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34:21847–21859, 2021. [5.11.2](#)
 - [20] Carl Doersch. Tutorial on variational autoencoders, 2016. [2.3.3](#)
 - [21] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023. [6.2](#)
 - [22] Ben Eisner, Harry Zhang, and David Held. Flowbot3d: Learning 3d articulation

- flow to manipulate articulated objects. *arXiv preprint arXiv:2205.04382*, 2022. [1](#)
- [23] Clemens Eppner and Oliver Brock. Visual detection of opportunities to exploit contact in grasping using contextual multi-armed bandits. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 273–278, 2017. doi: 10.1109/IROS.2017.8202168. [4.2.1](#)
- [24] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005. [3.2](#)
- [25] Zohar Feldman, Hanna Ziesche, Ngo Anh Vien, and Dotan Di Castro. A hybrid approach for learning to shift and grasp with elaborate motion primitives. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6365–6371. IEEE, 2022. [5.2](#)
- [26] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022. [1](#)
- [27] Joan Fontanals, Bao-Anh Dang-Vu, Oliver Porges, Jan Rosell, and Máximo A. Roa. Integrated grasp and motion planning using independent contact regions. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 887–893, 2014. doi: 10.1109/HUMANOIDS.2014.7041469. [4.2.2](#), [4.4.5](#)
- [28] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. [3.1](#), [3.2](#)
- [29] 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. [2.1](#), [2.5](#), [A.2](#)
- [30] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018. [2.3.1](#), [2.4.3](#), [2.5.4](#), [5.1](#), [5.2](#), [5.3](#), [5.5.2](#), [D.2.1](#)
- [31] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019. [1](#), [2.1](#), [2.2](#), [2.3.2](#), [2.4.1](#), [2.4.2](#), [2.5](#), [3.1](#), [3.2](#), [3.4](#)
- [32] Dibya Ghosh, Avi Singh, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. Divide-and-conquer reinforcement learning. *arXiv preprint arXiv:1711.09874*, 2017. [4.4.5](#)
- [33] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou

- Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023. [5.11.2](#)
- [34] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018. [4.4.5](#), [4.5](#), [5.2](#), [5.3](#), [5.5.2](#), [C.2.7](#)
- [35] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020. [3.1](#), [3.2](#), [3.4](#)
- [36] Kaiyu Hang, Andrew S. Morgan, and Aaron M. Dollar. Pre-grasp sliding manipulation of thin objects using soft, compliant, or underactuated hands. *IEEE Robotics and Automation Letters*, 4(2):662–669, 2019. doi: 10.1109/LRA.2019.2892591. [4.1](#), [4.2.2](#), [4.4.2](#)
- [37] Demis Hassabis, Dhharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017. [3.1](#)
- [38] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015. [5.2](#), [5.3](#)
- [39] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016. [3.7.1](#)
- [40] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision—ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*, pages 548–562. Springer, 2013. [D.1.3](#)
- [41] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. [3.2](#)
- [42] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016. [2.2](#)
- [43] Yifan Hou and Matthew T Mason. Robust execution of contact-rich motion plans by hybrid force-velocity control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1933–1939. IEEE, 2019. [1](#), [5.1](#), [5.2](#), [D.1.6](#), [D.6](#)
- [44] Yifan Hou, Zhenzhong Jia, and Matthew T. Mason. Fast planning for 3d

- any-pose-reorienting using pivoting. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1631–1638, 2018. doi: 10.1109/ICRA.2018.8462834. [4.1](#), [4.2.1](#)
- [45] Yifan Hou, Zhenzhong Jia, and Matthew Mason. Manipulation with shared grasping. In *Robotics: Science and Systems*, 2020. [1](#), [4.1](#), [4.2.1](#)
- [46] Wenlong Huang, Igor Mordatch, Pieter Abbeel, and Deepak Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021. [5.2](#)
- [47] Andrew Hundt, Benjamin Killeen, Nicholas Greene, Hongtao Wu, Heeyeon Kwon, Chris Paxton, and Gregory D Hager. “good robot!”: Efficient reinforcement learning for multi-step visual tasks with sim to real transfer. *IEEE Robotics and Automation Letters*, 5(4):6724–6731, 2020. [5.2](#)
- [48] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020. [3.2](#)
- [49] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019. [2.2](#)
- [50] Rae Jeong, Jost Tobias Springenberg, Jackie Kay, Daniel Zheng, Yuxiang Zhou, Alexandre Galashov, Nicolas Heess, and Francesco Nori. Learning dexterous manipulation from suboptimal experts. *arXiv preprint arXiv:2010.08587*, 2020. [3.1](#), [3.2](#), [3.3.2](#)
- [51] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil Sreenath. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot, 2022. [5.11.1](#)
- [52] Yandong Ji, Gabriel B. Margolis, and Pulkit Agrawal. Dribblebot: Dynamic legged manipulation in the wild, 2023. [5.11.1](#)
- [53] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019. [3.6](#)
- [54] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018. [1](#), [4.2.2](#)
- [55] 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. [4.4.4](#)

- [56] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020. [3.2](#)
- [57] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020. [2.2](#), [2.3.2](#), [3.2](#)
- [58] Jennifer King, Matthew Klingensmith, Christopher Dellin, Mehmet Dogar, Prasanna Velagapudi, Nancy Pollard, and Siddhartha Srinivasa. Pregrasp manipulation as trajectory optimization. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013. doi: 10.15607/RSS.2013.IX.015. [4.2.2](#), [4.4.2](#)
- [59] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014. [2.3.3](#)
- [60] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. [3.1](#), [3.2](#)
- [61] Orr Krupnik, Igor Mordatch, and Aviv Tamar. Multi agent reinforcement learning with multi-step generative models. *CoRR*, abs/1901.10251, 2019. URL <http://arxiv.org/abs/1901.10251>. [2.2](#)
- [62] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11784–11794, 2019. [2.2](#), [2.3.2](#), [2.4.1](#), [2.5](#), [3.2](#)
- [63] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020. [3.2](#)
- [64] Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. *arXiv preprint arXiv:2211.15144*, 2022. [6.2](#)
- [65] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012. [2.1](#), [2.2](#), [3.2](#)
- [66] Alex X Lee, Coline Manon Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *5th Annual Conference on Robot Learning*, 2021. [4.2.3](#)

- [67] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 1, 2.1, 2.2, 3.2
- [68] Jacky Liang, Xianyi Cheng, and Oliver Kroemer. Learning preconditions of hybrid force-velocity controllers for contact-rich manipulation. *arXiv preprint arXiv:2206.12728*, 2022. 1, 5.2
- [69] 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. 1, 2.3.1, 2.4.3
- [70] 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. 5.1, 5.2
- [71] Minghua Liu, Xuanlin Li, Zhan Ling, Yangyan Li, and Hao Su. Frame mining: a free lunch for learning robotic manipulation from 3d point clouds. In *6th Annual Conference on Robot Learning*, 2022. 5.2
- [72] Weiyu Liu, Tucker Hermans, Sonia Chernova, and Chris Paxton. Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects. *arXiv preprint arXiv:2211.04604*, 2022. 5.6, D.1.1
- [73] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017. 3.1, 3.2
- [74] Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 515–524. PMLR, 13–15 Nov 2017. 1
- [75] Roberto Martín-Martín, 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. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1010–1017. IEEE, 2019. 4.4.4
- [76] Matthew T Mason. Progress in nonprehensile manipulation. *The International Journal of Robotics Research*, 18(11):1129–1141, 1999. 5.2
- [77] Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. Prediction and control with temporal segment models. *CoRR*, abs/1703.04070, 2017. URL <http://arxiv.org/abs/1703.04070>. 2.2

- [78] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 1
- [79] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 5.2
- [80] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 2.1
- [81] Kaichun Mo, Leonidas J. Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6813–6823, October 2021. 1, 5.1, 5.2, 5.7
- [82] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012. 5.2
- [83] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *International Conference on Computer Vision (ICCV)*, 2019. 4.1, 4.2.2, 4.3
- [84] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Cathera Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 5.5.1, 5.1, 5.7, 5.7
- [85] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping, 2020. 4.2.2, 4.3
- [86] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-dof grasping for target-driven object manipulation in clutter, 2020. 4.1, 4.2.2, 4.3
- [87] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020. 1, 4.2.3
- [88] Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7477–7484. IEEE, 2022. 5.11.2

- [89] Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on Robot Learning*, pages 735–751. PMLR, 2020. [5.2](#)
- [90] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 7(3):3–16, 1988. [4.2.2](#)
- [91] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik’s cube with a robot hand, 2019. [4.1](#), [4.2.3](#), [4.4](#), [4.4.5](#), [4.4.6](#), [C.3](#)
- [92] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017. [2.1](#)
- [93] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017. [2.1](#)
- [94] 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. [2.2](#)
- [95] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007. [2.2](#)
- [96] 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. [1](#), [4.2.2](#)
- [97] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. [5.5.1](#)
- [98] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. [5.5.2](#)
- [99] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. *Conference on Robot Learning (CoRL)*, 2022. [1](#), [5.2](#), [5.1](#), [5.7](#), [5.7](#)

- [100] 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. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018. [2.1](#)
- [101] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018. [3.6](#)
- [102] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. [3.1](#), [3.2](#), [3.4](#)
- [103] 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, 2011. [2.2](#)
- [104] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. [4.4.2](#), [C.4.1](#)
- [105] 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. [3.2](#)
- [106] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. [3.1](#), [3.2](#)
- [107] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015. [4.4.1](#)
- [108] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018. [3.2](#)
- [109] Daniel Seita, Yufei Wang, Sarthak Shetty, Edward Li, Zackory Erickson, and David Held. ToolFlowNet: Robotic Manipulation with Tools via Predicting Tool Flow from Point Clouds. In *Conference on Robot Learning (CoRL)*, 2022. [5.2](#)
- [110] Karun B Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996. [4.2.2](#)

- [111] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022. [5.2](#)
- [112] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020. [3.2](#)
- [113] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014. [2.3.1](#)
- [114] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550 (7676):354–359, 2017. [1](#)
- [115] Mohsen Sombolstan and Quan Nguyen. Hierarchical adaptive locomanipulation control for quadruped robots, 2023. [5.11.1](#)
- [116] Mohsen Sombolstan and Quan Nguyen. Hierarchical adaptive control for collaborative manipulation of a rigid object by quadrupedal robots, 2023. [5.11.1](#)
- [117] 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. [4.2.2](#)
- [118] Zhaole Sun, Kai Yuan, Wenbin Hu, Chuanyu Yang, and Zhibin Li. Learning pregrasp manipulation of objects from ungraspable poses, 2020. [4.1](#), [4.2.2](#), [4.4.2](#)
- [119] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444, 2021. doi: 10.1109/ICRA48506.2021.9561877. [5.2](#)
- [120] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444. IEEE, 2021. [5.11.2](#)
- [121] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. [1](#)
- [122] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017. [3.2](#), [6.2](#)

- [123] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998. [3.1](#), [3.2](#)
- [124] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world, 2017. [4.2.3](#)
- [125] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. [4.5](#), [5.6](#), [C.2.1](#)
- [126] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz-Rodríguez, and David Filliat. Discorl: Continual reinforcement learning via policy distillation. *arXiv preprint arXiv:1907.05855*, 2019. [3.2](#)
- [127] Nikolaus Vahrenkamp, Martin Do, Tamim Asfour, and Rüdiger Dillmann. Integrated grasp and motion planning. In *2010 IEEE International Conference on Robotics and Automation*, pages 2883–2888, 2010. doi: 10.1109/ROBOT.2010.5509377. [4.2.2](#), [4.4.5](#)
- [128] Lirui Wang, Yu Xiang, and Dieter Fox. Manipulation trajectory optimization with online grasp synthesis and selection. In *Robotics: Science and Systems (RSS)*, 2020. [4.1](#), [4.2.2](#), [4.4.5](#)
- [129] Lirui Wang, Yu Xiang, Wei Yang, Arsalan Mousavian, and Dieter Fox. Goal-auxiliary actor-critic for 6d robotic grasping with point clouds. In *The Conference on Robot Learning (CoRL)*, 2021. [4.2.2](#), [5.2](#)
- [130] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020. [3.2](#), [3.3.2](#), [3.7.3](#)
- [131] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Johnny Lee, Szymon Rusinkiewicz, and Thomas Funkhouser. Spatial action maps for mobile manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020. [5.2](#)
- [132] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. [2.2](#), [2.3.2](#), [2.4.1](#), [2.5](#), [3.2](#)
- [133] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018. [D.1.3](#)
- [134] Annie Xie and Chelsea Finn. Lifelong robotic reinforcement learning by retaining experiences. *arXiv preprint arXiv:2109.09180*, 2021. [3.2](#)

- [135] Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018. 5.2
- [136] Zhenjia Xu, He Zhanpeng, and Shuran Song. Umpnet: Universal manipulation policy network for articulated objects. *IEEE Robotics and Automation Letters*, 2022. 1, 5.1, 5.2, 5.7
- [137] Fan Yang, Chao Yang, Huaping Liu, and Fuchun Sun. Evaluations of the gap between supervised and reinforcement lifelong learning on robotic manipulation tasks. In *Conference on Robot Learning*, pages 547–556. PMLR, 2022. 3.2
- [138] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 30–37, 2016. doi: 10.1109/IROS.2016.7758091. 5.1, 5.2
- [139] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL <https://arxiv.org/abs/1910.10897>. 5.5.1, 5.1, 5.7, 5.7, 6.2
- [140] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020. 4.4.5, 6.2
- [141] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020. 2.2, 3.2
- [142] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021. 3.2, 6.2
- [143] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017. 3.7.1
- [144] 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. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE,

2018. [5.2](#)
- [145] 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. *Conference on Robot Learning (CoRL)*, 2020. [1](#), [5.2](#), [5.11.2](#)
- [146] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018. [3.2](#)
- [147] Hongchang Zhang, Jianzhun Shao, Yuhang Jiang, Shuncheng He, and Xiangyang Ji. Reducing conservativeness oriented offline reinforcement learning. *arXiv preprint arXiv:2103.00098*, 2021. [3.2](#)
- [148] 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. [C.4.1](#)
- [149] Wenxuan Zhou and David Held. Learning to grasp the ungraspable with emergent extrinsic dexterity. In *Conference on Robot Learning (CoRL)*, 2022. [1](#), [5.2](#), [5.5.1](#), [5.1](#), [5.7](#), [5.7](#)
- [150] Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. *arXiv preprint arXiv:1907.11740*, 2019. [3.7.1](#)
- [151] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, 2020. [1](#), [3.2](#)
- [152] Wenxuan Zhou, Steven Bohez, Jan Humplik, Nicolas Heess, Abbas Abdolmaleki, Dushyant Rao, Markus Wulfmeier, and Tuomas Haarnoja. Forgetting and imbalance in robot lifelong learning with off-policy data. In Sarath Chandar, Razvan Pascanu, and Doina Precup, editors, *Proceedings of The 1st Conference on Lifelong Learning Agents*, volume 199 of *Proceedings of Machine Learning Research*, pages 294–309. PMLR, 22–24 Aug 2022. URL <https://proceedings.mlr.press/v199/zhou22a.html>. [1](#)
- [153] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020. [4.5](#), [C.2.1](#)
- [154] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020. [5.6](#), [D.1.5](#)