Learning to Make Decisions in Robotic Manipulation

by

Siyu Dai

B.S. and B.B.A., Shanghai Jiao Tong University (2016)
S.M., Massachusetts Institute of Technology (2018)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author . . . .
Department of Mechanical Engineering
April 25, 2022

Certified by . . . .
Brian C. Williams
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by . . . .
Nicolas G. Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

Learning to Make Decisions in Robotic Manipulation

by

Siyu Dai

Submitted to the Department of Mechanical Engineering
on April 25, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

In order for human-assisting robots to be deployed in the real world such as household environments, challenges in two major scenarios remain to be solved. First, for common tasks that the robot conducts day-to-day, the execution of motion plans need to ensure the safety of surrounding objects and humans. Second, to handle new tasks that some customers might occasionally demand, robots need to be able to learn novel tasks efficiently with a minimal amount of human supervision. In this thesis, we show that machine learning methods can be applied to solve challenges in both scenarios. In the first scenario, we propose learning-based p-Chekov, a chance-constrained motion planning approach that utilizes data-driven methods to obtain safe motion plans in real time. By pre-training a collision risk estimation model off-line instead of conducting online sampling-based risk estimation, learning-based p-Chekov is able to significantly improve the planning speed while maintaining the chance-constraint satisfaction performance. In the second scenario of learning new tasks, we first propose empowerment-based intrinsic motivation, a reinforcement learning (RL) approach that allows robots to learn novel tasks with only sparse or binary reward functions. Through maximizing the mutual dependence between robot actions and environment states, namely the empowerment, this intrinsic motivation helps the agent to focus more on the states where it can effectively "control" the environment during exploration instead of the parts where its actions cause random and unpredictable consequences. Empirical evaluations in different robotic manipulation environments with different shapes of the target object demonstrate that this empowerment-based intrinsic motivation approach can obtain higher extrinsic task rewards faster than other state-of-the-art solutions to sparse-reward RL tasks. Another approach we propose in the second scenario is automatic curricula via expert demonstrations (ACED), an imitation learning method that leverages the idea of curriculum learning and allows robots to learn long-horizon tasks when only provided with a handful of demonstration trajectories. Through moving the reset states from the end to the beginning of demonstrations as the learning agent improves its performance, ACED not only learns challenging manipulation tasks with unseen initializations and goals, but also discovers novel solutions that are distinct from the demonstrations. In addition, ACED can be naturally combined with other imitation learning methods to utilize expert demonstrations in a more efficient manner and allow robotic manipulators to learn novel tasks that other state-of-the-art automatic curriculum learning methods cannot learn. In the experiments presented in this thesis, we show that a combination of ACED with behavior cloning allows pick-and-place tasks to be learned with as few as one demonstration and block stacking tasks to be learned with twenty demonstrations.

Thesis Supervisor: Brian C. Williams
Title: Professor of Aeronautics and Astronautics

Acknowledgments

It is hard to believe that this is my sixth year at MIT. This journey has been full of surprises and challenges, and it also marks a tremendous milestone in my career. The completion of this thesis is owed to the continuous support and feedback from my thesis committee, Dr. Andreas Hofmann, Prof. Brian Williams, Prof. Leslie Kaelbling, Prof. John Leonard and Prof. Alberto Rodriguez, throughout my PhD. With their help on brainstorming research ideas and shaping thesis directions, I was able to accomplish the research milestones that laid the foundation for this thesis. I'm also extremely grateful to all my labmates in MERS who have provided valuable feedback on my research results and offered insightful research discussions. It was a great pleasure working with all of you over the past six years, and I deeply hope our friendship last beyond grad school.

Another person I can't forget to mention is my former internship supervisor and collaborator Dr. Wei Xu, who has contributed significantly to the formation of an essential piece of my thesis research and has guided me to a research direction where I found my true passion. The completion of this thesis wouldn't have been possible without his support. In addition to Dr. Xu, I'm also thankful to all my former internship supervisors, mentors and collaborators, especially Dr. Yebin Wang, Dr. David Isele, Dr. Sangjae Bae, Dr. Aaron Parness and Dr. Sisir Karumanchi. Internships have been a critical part of my PhD experience and they contributed significantly to the growth of my research skills. I'm very grateful for all the mentorship and collaboration I received during and after these internships.

The TA experience during the past four years has been an integral part of my PhD. Teaching is never an easy task, and I'm very grateful for the opportunities for joining 2.003, 8.02, 6.UAR and 2.14 to improve my teaching and supervising skills. My co-TAs and instructors, including Arkopal Dutt, Rohit Shamshery, Prof. Alberto Rodriguez, Prof. Thomas Peacock, Prof. Kim Vandiver, Dr. Michelle Tomasik, Schrasing Tong, Quanquan Liu, Yui Sujichantararat, Prof. Dina Katabi, Prof. Michael Carbin, Prof. Piotr Indyk, Prof. David Trumper and many others, have all been very nice to work with and I have been constantly feeling empowered by being part of an amazing team. My experience in 6.UAR especially, which I spent four semester on, is not only helpful for gaining the skills on supervising research projects, but also for helping me to think more systematically about my own research and communication skills. The growth I have obtained throughout the TA experience has been invaluable.

I have met a lot of fantastic friends and had a really great time working at the student organizations at MIT, including the Graduate Association of Mechanical Engineers (GAME), Ashdown House Executive Committee (AHEC) and the MIT Graduate Student Council (GSC). Even though organizing and budgeting for large events can sometimes be very time consuming, tt was a nice memory to have brought so many events to fellow graduate students. The people I met throughout these years are not only colleagues that sit through meetings together, but also friends that volunteer at event together, party together and hang out together during our free time. These precious times I spent with the student organization friends will always be a precious part of my MIT memory.

Last but not the least, I would like to thank my family for their continuous trust and support. Like many other international students, I have not been able to see my parents and my extended family throughout my entire PhD. However, I know deeply in my heart that everyone cares so much about me from across the earth and supports my career choice despite the separation it has caused us. I can't wait to see them again after the COVID-19 pandemic is over. I am also incredibly grateful for my husband Benjamin Ayton. As the only family member I have by my side over the past years, he has helped me through many challenges and supported me over tough times. He has not only taught me to maintain a better work-life-balance and been my major source of happiness, but also helped me gain confidence and strength. I would not have become who I am today without him, and I very much look forward to the future we are going to build together.

Contents

1 Introduction **19**

1.1 Motivation . . . 19

1.2 Technical Need . . . 21

1.2.1 Challenge 1: Fast-Reactive Chance-Constrained Motion Planning 21

1.2.2 Challenge 2: Self-Supervised Learning with Sparse Reward Functions . . . 22

1.2.3 Challenge 3: Demonstration-Efficient Imitation Learning . . . 24

1.3 Contributions . . . 25

1.3.1 Learning-based P-Chekov . . . 25

1.3.2 Empowerment-based Intrinsic Motivation . . . 26

1.3.3 Automatic Curricula via Expert Demonstrations . . . 28

1.4 Thesis Overview . . . 29

2 Related Work **31**

2.1 Chance-Constrained Motion Planning . . . 31

2.1.1 Collision Risk Estimation . . . 32

2.1.2 Machine Learning in Motion Planning . . . 34

2.2 Reinforcement Learning with Sparse Rewards . . . 34

2.3 Demonstration-Efficient Imitation Learning . . . 36

2.3.1 Curriculum Learning . . . 36

2.3.2 Learning from Demonstration . . . 37

2.3.3 State Resetting . . . 38

3 Problem Statement **39**

3.1 Chance-Constrained Motion Planning 39

3.1.1 Model Definition 40

3.1.2 Constraint Definitions 41

3.1.3 Problem Definition 42

3.1.4 Assumptions 43

3.2 Reinforcement Learning with Sparse Rewards 44

3.2.1 Markov Decision Process (MDP) 44

3.2.2 Reinforcement Learning (RL) 45

3.2.3 Intrinsic Motivation 46

3.2.4 Problem Definition 48

3.3 Curriculum Learning through Demonstrations 49

3.3.1 Curriculum Learning 49

3.3.2 Learning from Demonstration 50

3.3.3 Problem Definition 50

4 Learning-based P-Chekov **53**

4.1 Preliminaries: Sampling-based P-Chekov 53

4.1.1 The P-Chekov Framework 53

4.1.2 Quadrature-based Collision Probability Estimation 56

4.2 Learning-based P-Chekov 59

4.3 Experiment Setup 63

4.3.1 Environments 63

4.3.2 Modeling 64

4.4 Learning-based P-Chekov Experiment Results 70

4.4.1 Comparison between Different Regression Methods 70

4.4.2 Neural Network Learning-based P-Chekov Experiment Results 77

4.4.3 Discussion 84

5 Empowerment-based Intrinsic Motivation **85**

5.1 Background 85

5.1.1 Mutual Information 86
5.1.2 Conditional Mutual Information 88
5.1.3 Mutual Information Computation 91
5.1.4 Empowerment 95
5.1.5 Intrinsic Curiosity Module 97
5.2 Approach: Empowerment-based Intrinsic Motivation 98
5.2.1 Approximations to Simplify Empowerment Calculation 99
5.2.2 Maximizing Empowerment using Mutual Information Lower Bounds 101
5.2.3 Combination with ICM to Facilitate Empowerment Computation 104
5.3 Empirical Evaluation 106
5.3.1 Environment Setup 106
5.3.2 Implementation Details 107
5.3.3 Experiment Results 112
5.3.4 Off-Policy Implementation 117
5.4 Application: Learning a Diverse Set of Skills 117

6 Automatic Curricula via Expert Demonstrations 123
6.1 Preliminaries 123
6.1.1 Behavior Cloning 124
6.2 Approach 124
6.3 Experiment Setup 129
6.3.1 Implementation Details 130
6.4 Empirical Evaluation Results 133
6.4.1 Pick-and-Place Tasks 133
6.4.2 Block Stacking Tasks 135
6.4.3 Learning Curves 138
6.5 Comparison with BC + RL without ACED 140
6.6 Combination with Empowerment-based Intrinsic Motivation 142
6.7 Discussion 145

7 Conclusion 147

7.1 Summary 147

7.2 Main Contributions 148

7.2.1 Learning-based P-Chekov 148

7.2.2 Empowerment-based Intrinsic Motivation 149

7.2.3 Automatic Curricula via Expert Demonstrations 150

7.3 Discussion and Future Work 151

List of Figures

1-1 Example Scenarios 20

1-2 Thesis Overview 30

3-1 Agent-environment interaction in an MDP. 45

4-1 System diagram for p-Chekov [33] 54

4-2 Simulation Environments for Learning-based P-Chekov Evaluation [32] 61

4-3 P-Chekov information flow from nominal trajectories to state probability distributions. 70

4-4 Minimum loss as a function of hidden layer numbers [31] 75

4-5 Loss and training epoch relationship for networks with 9 hidden layers with 512 units each [31] 77

4-6 Learning-based p-Chekov statistics breakdown for feasible cases with end-effector observation, 0.0044 noise standard deviation and 10% chance constraint [31] 81

5-1 Mutual Information Illustration 88

5-2 Conditional Mutual Information Illustration 90

5-3 Empowerment Intuition 99

5-4 Overview of the empowerment-based intrinsic motivation approach [34] 105

5-5 The simulation environments used in this section include the Fetch robot in the Mujoco environment and the PR2 robot in the Gazebo environment. The robot needs to learn how to interact with objects of different shapes, including box, cylinder and sphere. 108

5-6 Experiment results in the Fetch environment. (a)-(c) compare the performance of the proposed empowerment-based approach (referred to as empowerment with ICM since ICM is used to help training the empowerment prediction networks) with ICM and Disagreement in object lifting tasks. The solid lines represent the mean, and the shadow areas represent the 95% confidence intervals. [34] 113

5-7 Experiment results in the PR2 environment comparing the performance of the proposed empowerment-based approach (referred to as empowerment with ICM since ICM is used to help training the empowerment prediction networks) with ICM and Disagreement in object lifting tasks. The solid lines represent the mean, and the shadow areas represent the 95% confidence intervals. [34] 114

5-8 Experiment results in the Fetch environment comparing the proposed empowerment-based approach with HER in pick-and-place tasks. The solid lines represent the mean, the shadow areas represent the 95% confidence intervals, and the dashed lines in represent the maximum and minimum values. [34] 116

5-9 Comparison of off-policy implementation and on-policy implementation of the empowerment-based intrinsic exploration approach in the sphere lifting environment. The solid lines represent the mean of 10 experiments with different random seeds, and the shadow areas represent the 95% confidence intervals. [34] 118

5-10 Qualitative performance of the proposed empowerment-based intrinsic motivation when combined with the diversity-driven DIAYN [41] approach in the box lifting task with a Fetch robot. (a)-(c) show the different skills learned when the number of skills in DIAYN is set to 3. [34] From the figures we can see that the robots has learned different ways of interacting with the object. 120

5-11 Qualitative performance of the proposed empowerment-based intrinsic motivation when combined with the diversity-driven DIAYN [41] approach in the box lifting task with a Fetch robot. (a)-(e) show the different skills learned when the number of skills in DIAYN is set to 5. [34] From the figures we can see that the robots has learned different ways of interacting with the object. 120

5-12 Different skills learned with DIAYN [41] without the empowerment-based intrinsic motivation in the box lifting task with a Fetch robot when the number of skills is set to 5. [34] From the figures we can see that the skills the robot has learned do not involve interactions with the object. 121

6-1 ACED Intuition 125

6-2 Example of demonstration trajectory segmentation: an expert demonstration trajectory can be divided into sections where larger section number indicates being closer to the initial state. The total number of sections is also called the total number of curricula C_{max}, and in this example $C_{max} = 3$. Normal rollout workers randomly sample an initial state from the initial state distribution S_0 and a goal state from the goal state distribution S_g. For curriculum rollout workers, the environments are reset based on the curriculum number C: curriculum-C tasks reset the environment to a section-C state on a randomly selected demonstration trajectory. ACED starts training with $C = 1$ and gradually moves reset states towards the beginning of demonstration trajectories by increasing C. When ACED switches to normal rollout workers, the reset states are drawn from the actual S_0 the target task specifies, and this is when it starts to generalize to unseen initializations. [30] . . . 126

6-3 Simulation environments for ACED evaluation 130

6-4 Number of environment steps ACED with BC takes to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval. [30] 132

6-5 Number of environment steps ACED without BC takes to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval. [30] 132

6-6 Visualization of two different stacking policies ACED with BC converged to with different $|\mathcal{T}|$ values. [30] 137

6-7 Learning curves of different algorithms in the pick-and-place environment with 5 demonstration trajectories. The horizontal axis represents the number of environment steps during training and the vertical axis represents the success rate. Expert and BC success rates are represented by dash lines because they didn't have training processes and their success rates remain constant. 139

6-8 Learning curves of ACED with BC, the reverse curriculum method [45], and the Montezuma's Revenge method [118] in the block stacking task. 139

6-9 Number of environment steps the combination of ACED with BC and empowerment-based intrinsic motivation takes to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval. 143

6-10 Number of environment steps ACED with empowerment and ACED without empowerment take to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval. 144

List of Tables

4.1 Best Parameters in Kernel Ridge Regression and Random Forest Regression [31] 72
4.2 Comparison of Different Regression Methods [31] 73
4.3 Performance of Neural Network with 9 Hidden Layers with 512 Units Each [31] 76
4.4 Learning-based P-Chekov Performance with Noise Level 0.0044 and Chance Constraint 10% [31] 82
4.5 Quadrature-based P-Chekov Performance with Noise Level 0.0044 and Chance Constraint 10% [29] 83

5.1 Comparison of Mutual Information Estimation Approaches [34] . . . 103

6.1 Pick-and-Place Success Rate [30] 133
6.2 Block Stacking Performance [30] 136
6.3 Pick-and-Place Comparison with BC + RL 141
6.4 ACED with Empowerment Success Rate in Pick-and-Place Tasks . . 143

Chapter 1

Introduction

1.1 Motivation

Imagine a robot factory is trying to manufacture home support robots and sell them all over the world. Two types of scenarios will likely need to be considered when designing the decision-making algorithms for the robot. First, there might be common skills that many customers will demand and the home support robot will need to execute day-to-day. When the user needs the robot to accomplish common tasks, the robot should be able to quickly come up with feasible solutions and execute them safely. It is important that the planning process for these common skills doesn't take too long because customers will lose patience if the robot requires a long time to think before it can complete common daily tasks. Additionally, as robots that interact with human users frequently, it is essential that the motion plans the robot comes up with can be executed safely without a high probability of collision to avoid causing injuries. The second scenario robot designers need to consider while designing the decision making algorithm is, when the user wants the robot to acquire a new skill, the robot needs to learn with a minimal amount of human supervision. The ability of acquiring new skills during deployment is important because it is impossible to pre-train the robot for all possible future tasks it might be asked to do that can satisfy a wide range of customers. A successful home support robot product should be able to gain new skills with a limited amount of supervision, because no customers like a robot that requires

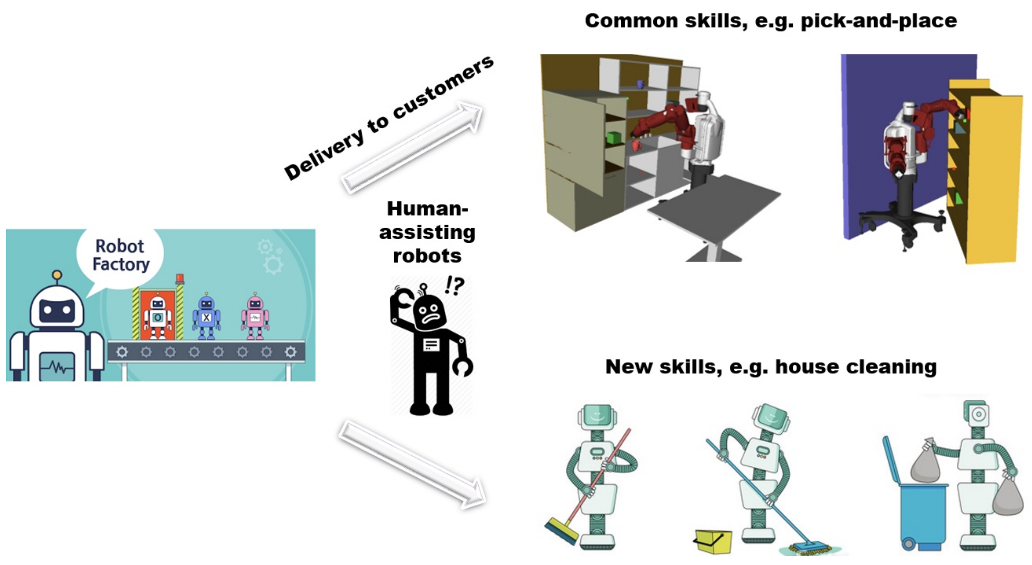

Common skills, e.g. pick-and-place
Delivery to customers
Human-assisting robots
Robot Factory
!?
New skills, e.g. house cleaning

Figure 1-1: Example Scenarios

intensive training every time it needs to learn a new skills. Figure 1-1 provides a visual illustration of the two types of scenarios.

One way to handle the first scenario is to pre-define a set of basic skills that are hand-coded in the robot's software system. These skills should be able to incorporate continuous parameters in initial states and task goals, and come up with waypoints the robot needs to reach in order to solve the task. In this way, accomplishing these common tasks is turned into a motion planning problem that tracks a set of waypoints, and the challenge becomes how to make safe motion plans that satisfy constraints on collision risks in real time.

As for the second scenario, the robot designers need to consider what types of supervision can be easily provided and is suitable for a wide range of customers. In general, the customers will likely not be willing to program a detailed task specification or carefully design a set of rewards to guide the robot. What they are more likely to provide is probably similar to the reward signals they would give while teaching a human learning agent, including a high-level goal, a handful of demonstrations, or a combination of both. Therefore, it is important that the robot is able to conduct self-supervised learning when provided with only a sparse or binary reward function or a handful of human demonstration trajectories.

In order to provide solutions for the above two scenarios, this thesis considers three major challenges in robotic manipulation: for the first scenario, we consider 1) how to provide risk-aware motion plans while maintaining short planning times; for the second scenario, we consider 2) how to conduct effective self-supervised learning with only sparse or binary reward functions and 3) how to efficiently utilize human demonstrations and acquire new skills through imitation.

1.2 Technical Need

1.2.1 Challenge 1: Fast-Reactive Chance-Constrained Motion Planning

Robotic systems deployed in the real world have to contend with a variety of challenges: wheels slip for mobile robots, lidars do not reflect off glass doors, and humans in the environment move in unpredictable manners. However, many state-of-the-art robots, with inevitable uncertainties from various sources including approximate models of system dynamics, imperfect sensors, and stochastic motions caused by controller noise, are not yet ready to handle these challenges. Although nowadays feedback controllers can take care of a large portion of the uncertainties during the execution phase, the remaining deviations can still be problematic, especially for robots operating in hazardous environments or systems that collaborate closely with humans. One representative example is a manipulator mounted on an underwater vehicle, which faces not only the disturbances from currents and inner waves, but also the base movements caused by the interaction between manipulators and the vehicle on which they are mounted. A collision accident of such manipulators deployed in underwater scientific exploration tasks can often cost millions of dollars. Another typical example is a domestic assistive robot surrounded by elder people and children, which needs to be very careful about collision avoidance. Therefore, in those tasks, it is important that the motion planner can take uncertainties into account and can react quickly to plan interruptions.

Fast-reactive risk-aware motion planning for high-dimensional robots like humanoid robots, however, is a very challenging task. Unlike vehicle robots, a typical robotic manipulator can have seven degrees-of-freedom (DOFs), and this high-dimensionality makes it extremely difficult to quantify uncertainties into collision risks and to make safe motion plans in real time. Existing systems that tackle the risk-aware motion planning problem [136, 80, 91, 126, 22, 8, 81] lack the ability of efficiently handling high-dimensional robots and non-convex environments. In order to address these difficulties and provide motion plans that satisfy constraints over the probability of plan failure, i.e. *chance constraints* [90], Dai et al. [29, 33] proposed probabilistic Chekov (p-Chekov), a combined sampling-based and optimization-based approach that can effectively reason over uncertainties for high-dimensional robots. However, since the p-Chekov framework presented in [29, 33] uses a quadrature-based sampling approach to estimate collision risks at each time step in the motion plan, its application in real-time motion planning tasks for high-dimensional robots is severely obstructed by the speed of this risk estimation component. Therefore in this thesis, we propose *learning-based p-Chekov* in order to improve p-Chekov's efficiency and accuracy in terms of online collision estimation and achieve fast-reactive chance-constrained motion planning.

1.2.2 Challenge 2: Self-Supervised Learning with Sparse Reward Functions

In order to design robotic agents that can adapt to new environments and learn new tasks, it is important that they can explore and learn on their own instead of requiring intensive human supervision. The traditional task and motion planning approach to robotic manipulation [62] typically requires a significant amount of domain-specific prior knowledge, and acquiring this knowledge often involves intensive human engineering. On the other hand, reinforcement learning (RL) agents have demonstrated impressive performances in scenarios with well-structured environment and dense reward signals [73]. Unlike many games that have clear rules and can assign specific reward values to each of the agent's actions, it is often not intuitive to design this type of dense reward functions in robotics tasks. For example, if a robot is tasked with cooking a dish, the most intuitive way for the human user to provide a reward function is to rate the dish when cooking is finished instead of trying to come up with a specific reward value for each of the robot's actions. However, learning-based approaches to manipulation typically only work well when the reward function is densely distributed throughout the state space or when a significant amount of expert demonstrations are available. This is because when the state and action space is high-dimensional and the reward signal is sparse, the probability of accidentally bumping into a state the provides non-zero rewards is very low and the RL agents could potentially spend a long time exploring the state space without getting any reward signal. Therefore, RL has seen less success in tasks with unstructured environments like robotic manipulation where the dynamics and task rewards are less intuitive to model.

Designing task-specific dense reward functions to simplify the sparse-reward RL problem has been a common solution for manipulation problems, but in most practical applications, hand designing dense reward functions for every robot in every task and every environment is infeasible and might bias the agent's behavior in a suboptimal way [4]. Inverse reinforcement learning approaches seek to automate reward definition by learning a reward function from expert demonstrations, but inevitably demand a significant amount of task-specific knowledge and place considerable expert data collection burden on the user [116]. Recent advances in meta-learning allow agents to transfer learned skills to other similar tasks [42, 23], but a large amount of prior meta-training data across a diverse set of tasks is required, which also becomes a burden if a lot of human intervention is needed. Therefore, effectively solving sparse reward problems from scratch is an important capability that will allow RL agents to be applied in practical robotic manipulation tasks. To tackle these issues, in this thesis, we propose an empowerment-based intrinsic exploration approach that allows robots to learn manipulation skills with only sparse extrinsic rewards from the environment without the requirement on hand-designing dense reward shaping functions for each individual task.

1.2.3 Challenge 3: Demonstration-Efficient Imitation Learning

A notable cognitive capability humans have is the instinct to learn from expert demonstrations. Babies learn to walk before they are able to understand language instructions, and even adult humans often prefer demonstrations than verbal task goals when learning many new skills. In a lot of tasks in robotics, both a binary reward for tasks success and a small amount of demonstrations can be provided naturally, so can we use demonstrations to overcome the challenging exploration problem RL agents face in these long-horizon sparse-reward tasks?

Imitation learning [57, 107] methods resort to expert demonstrations instead of hand-designed reward signals and have shown impressive performance, especially in tasks where the reward functions are tricky to define but demonstrations are easier to obtain. However, how to efficiently and effectively utilize demonstration trajectories has been a long-standing challenge for researchers. Behavior cloning [109, 10] applies supervised learning to train agents that imitate expert behaviors, but inevitably demands a large amount of demonstrations [11, 138, 111] and its performance often suffers from data distribution mismatch [117]. Adversarial imitation learning approaches based on Generative Adversarial Networks (GAN) [57, 5, 107] have effectively scaled to applications with relatively high-dimensional environments, but still have limited ability to handle long-horizon tasks with complicated environments due to their unstable training performance. In this thesis, we introduce Automatic Curricula via Expert Demonstrations (ACED), an RL approach that combines the ideas of imitation learning and curriculum learning in order to solve challenging robotic manipulation tasks with sparse reward functions and allow robotic agents to acquire new skills with only a handful of state-only demonstration trajectories without access to the demonstration actions. One major advantage of ACED is that it is a intuitive approach for utilizing expert demonstrations and can be easily combined with most other imitation learning methods to further improve its sample efficiency in terms of demonstration trajectories.

1.3 Contributions

To provide viable solutions to the three challenges introduced in Section 1.2 respectively, we make three major contributions in this thesis:

1. Learning-based P-Chekov
2. Empowerment-based Intrinsic Motivation
3. Automatic Curricula via Expert Demonstrations

1.3.1 Learning-based P-Chekov

The quadrature-based p-Chekov approach introduced in [29, 33] applies a sampling-based collision risk estimation approach that mitigates the inaccuracy of random sampling and avoids the difficulty of mapping between C-space and workspace. Although it can significantly reduce the number of samples required for collision risk estimation at each time step in the trajectory, its computation time in high-dimensional planning space still obstructs its application in real-time motion planning tasks. Even though only two quadrature nodes per dimension are used to estimate the collision risk for each waypoint, the total number of collision tests conducted online is still very big when the manipulator have 7 DOFs ($2^7 \times n_{waypoints}$ collision tests for each nominal trajectory). Additionally, two-node quadratures have very limited ability of approximating non-smooth functions, whereas the collision functions here are highly non-smooth. Therefore, quadrature-based p-Chekov inevitably suffers from errors when approximating the collision risk, and the efficiency and accuracy of risk estimation becomes its bottleneck that restricts its application in uncertainty-sensitive real-time manipulation planning tasks.

In order to address the aforementioned issues, this thesis introduces machine learning approaches into the collision risk estimation component of p-Chekov in order to improve its efficiency and accuracy [31]. We hypothesize that if we take enough samples containing nominal configurations with their probability distributions and risks of collision from the environment that the robot will be interacting with in order to train a regression model offline, then this model can act as the collision risk estimation component in p-Chekov in the online planning phase which makes accurate predictions given a nominal trajectory and the state distributions. In this way, the time-consuming collision risk estimation component during online planning will be moved offline, hence the bottleneck of the original p-Chekov approach will break and the planning speed will likely improve significantly. In order to test this hypothesis, we analyze the risk estimation performance of three different classes of regressors in the Scikit Learn [105] package (kernel ridge regressor, random forest regressor, and Gaussian process regressor) as well as neural networks through the Keras [24] interface with TensorFlow [1] back engine. Preliminary experiments show that neural networks can outperform all the other supervised learning models and are able to efficiently train on a large amount of offline collected data and make accurate online risk predictions quickly. We then propose a *learning-based p-Chekov* approach that can overcome quadrature-based p-Chekov's limitations in planning speed and achieve fast-reaction as well as high chance constraint satisfaction rate for real-world high-dimensional robotic motion planning tasks. Empirical results show that this learning-based p-Chekov system can significantly reduce the planning time required to generate feasible solutions that satisfy the specified chance constraint in practical manipulation tasks.

1.3.2 Empowerment-based Intrinsic Motivation

Babies are naturally curious about the environment surrounding them and are intrinsically motivated to try out different actions and explore ways to interact with environment objects. Inspired by how human babies learn, intrinsic motivation has become a popular approach in RL in order to encourage robotic agents to explore properties of the environment without being directed by external reward functions. Curiosity-driven intrinsic motivations [100, 101] have seen success in various fields including video games and robotics. However, we argue that curiosity may not be the most suitable form of intrinsic motivation for many robotic manipulation tasks where principled and predictable interactions with environment objects are desired. Imagine a robot is interacting with a box on the table in order to learn how to lift it up. Intuitively, the undesirable behaviors of knocking the box onto the floor should generate higher novelty since it will always lead the object to states that haven't been previously explored, and a curiosity-driven agent seeking maximum novelty might get stuck pushing the box off to the floor again and again. Instead, an ideal agent for object manipulation should prefer actions that control the object instead of ones that lead it to unpredictable novel states.

Therefore, in this thesis, we propose an empowerment-based intrinsic motivation approach that addresses the above issues and allows robots to learn manipulation skills with only sparse extrinsic rewards from the environment [34]. Empowerment is an information-theoretic concept proposed in an attempt to find local and universal utility functions which help individuals survive in evolution by smoothening the fitness landscape [67]. Through measuring the mutual information between actions and states, empowerment indicates how confident the agent is about the effect of its actions in the environment. Therefore, using empowerment as an additional reward encourages the RL agents to learn policies that influences the environment objects in a predictable way. In contrast to novelty-driven intrinsic motivations which encourage the agent to explore actions with unknown effects, empowerment emphasizes the agent's "controllability" over the environment and favors actions with predictable consequences. Despite the well-suited intuition of using empowerment in robotic manipulation tasks, as a form of conditional mutual information, computing empowerment in continuous state space is intractable and it is nontrivial to extend this concept in high-dimensional robotic environments. We develop practical simplifications for conditional mutual information computation in order to overcome the challenges involved in empowerment estimation, and show that this empowerment-based approach outperforms other state-of-the-art intrinsic exploration methods in manipulation tasks with sparse reward functions, including object lifting and pick-and-place.

Although the concept of empowerment has previously been discussed in the context of RL [83], it was evaluated in a grid world application with low-dimensional discrete state spaces. To the author's best knowledge, our approach is the first successful demonstration of the effectiveness of empowerment in terms of assisting RL agents in learning complicated robotics tasks with sparse rewards.

1.3.3 Automatic Curricula via Expert Demonstrations

Curriculum learning [14] is a continual learning method that starts training with easier tasks and gradually increases task difficulty in order to accelerate the learning progress. Through preparing the learning agent for challenging tasks using easier step-stone tasks, it has seen success in many applications including language modeling [52], autonomous navigation [82, 61] and robotic manipulation [45]. However, many curriculum-based methods only involve a small and discrete set of manually generated task sequences as the curriculum, and the automated curriculum generating methods often assume known goal states or prior knowledge on how to manipulate the environment [45, 140] and bias the exploration to a small subset of the tasks [125]. With a well-designed curriculum, RL agents can first solve simpler problems where rewards are easy to obtain in order to master the skills that can increase their chance of getting rewards in the challenging tasks, but in many manipulation tasks, designing curricula can be tricky and tedious. Therefore, how to automatically design effective and generalizable curricula remains a challenging research problem. Inspired by the idea of imitation learning, we hypothesize that expert demonstration trajectories can be utilized to automatically generate a sequence of curricula and help solve challenging exploration problems in RL.

In order to test this hypothesis, we propose Automatic Curricula via Expert Demonstrations (ACED), a RL approach which uses states from different sections along demonstration trajectories as reset states and controls the curriculum by moving reset states from the end of the demonstrations to the beginning based on the agent's performance [30]. Suppose we have a demonstration trajectory $\tau = (s_0, s_1, \ldots, s_{T-1}, s_T)$, where s_0 is the initial state and s_T is the goal state. If we assume the demonstration trajectory solves the task in a reasonable way without deliberate detours (even though it could be suboptimal), then it is also reasonable to assume that s_{T-1} is closer to s_T in the task space than s_1, which means an RL agent starting from s_{T-1} will likely have a higher chance of reaching s_T than an agent starting from s_1 within a limited time of random exploration. Therefore, among all tasks with binary rewards that are only given when the goal s_T is fully reached, the ones with agents initialized at s_{T-1} should be easier than the ones with agents initialized at s_1. We hypothesize that agents who have already learned how to reach s_T from s_{T-1} can enjoy a warm start while trying to reach s_T from s_1, and that these tasks starting from different initial states can form a systematic curriculum for learning challenging long-horizon tasks with sparse rewards. With this intuition, we evaluate ACED in robotics pick-and-place tasks and block stacking tasks with only binary rewards, two challenging tasks in the continuous control domain that haven't been solved by vanilla RL algorithms, and analyze the influence of the number of demonstrations and the total number of sections the demonstrations are divided into on ACED's performance. An additional advantage of ACED is that it can be naturally combined with many other methods of utilizing human demonstrations in order to further improve its performance or reduce the number of demonstration trajectories needed, and a combination of ACED and behavior cloning (BC) is demonstrated as an example in this thesis. Empirical results show that pick-and-place can be learned with as few as 1 demonstration, and block stacking can be learned with as few as 20 demonstrations.

1.4 Thesis Overview

This thesis is structured as follows. Chapter 2 reviews literature related to the aforementioned three major challenges and compares them with our proposed solutions. Chapter 3 formally states the three challenges this thesis is proposing solutions to. Chapter 4 introduces learning-based p-Chekov, a fast-reactive chance-constrained motion planning approach, and provides extensive empirical evaluation results in simulation environments for practical robotic manipulation scenarios. Chapter 5 describes

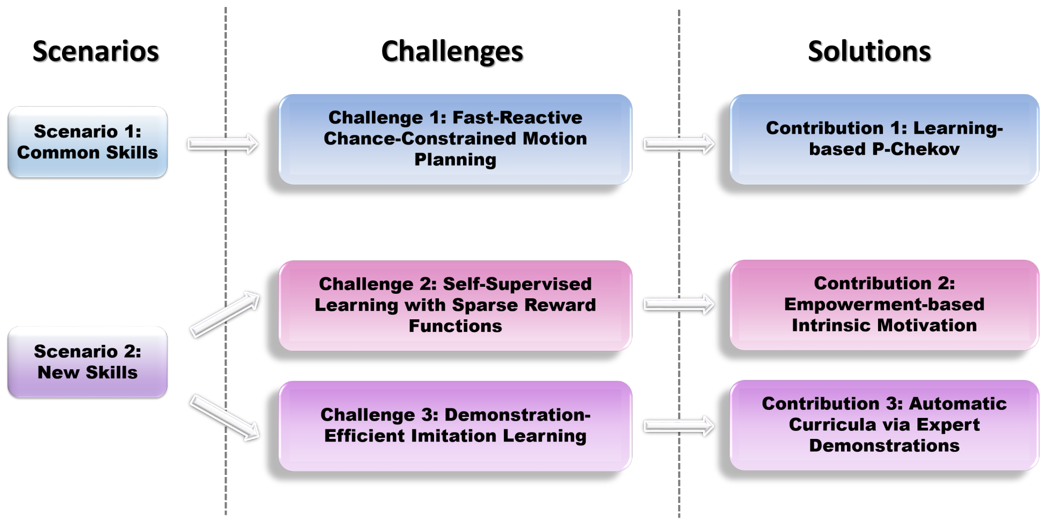

Scenarios
Challenges
Solutions
Scenario 1: Common Skills
Challenge 1: Fast-Reactive Chance-Constrained Motion Planning
Contribution 1: Learning-based P-Chekov
Scenario 2: New Skills
Challenge 2: Self-Supervised Learning with Sparse Reward Functions
Contribution 2: Empowerment-based Intrinsic Motivation
Challenge 3: Demonstration-Efficient Imitation Learning
Contribution 3: Automatic Curricula via Expert Demonstrations

Figure 1-2: Thesis Overview

empowerment-based intrinsic motivation, a form of intrinsic reward that can be included in any standard RL algorithm and guide robots to explore the states that are beneficial for accomplishing manipulation tasks, and demonstrates its performance in object lifting tasks and pick-and-place tasks. Chapter 6 presents automatic curricula via expert demonstrations, an RL approach that combines ideas from imitation learning and curriculum learning, and provides empirical evaluation results in pick-and-place tasks and object-lifting tasks. Chapter 7 summarizes the thesis and discusses potential directions of future work. Figure 1-2 illustrates the overall structure of this thesis.

Chapter 2

Related Work

This chapter reviews the literature related to the three major contributions made in this thesis respectively. Section 2.1 reviews related work of chance-constrained motion planning, the contribution made in Chapter 4. Section 2.2 reviews the related literature of reinforcement learning with sparse rewards, the contribution made in Chapter 5. Section 2.3 discusses the related work of demonstration-efficient imitation learning, the contribution made in Chapter 6.

2.1 Chance-Constrained Motion Planning

Existing motion planners that take uncertainties into consideration include two classes: some are safety-driven and provide motion plans that minimize the collision risks [135, 103, 102, 141], and others, also called *chance-constrained motion planners* [91], seek the optimal plans that can satisfy a user-specified constraint over the probability of collision. In this thesis, we focus on providing chance-constrained motion plans for high-dimensional robots in real time. Many uncertainty-aware motion planners are based on Markov Decision Processes (MDPs) [131, 21, 3], and an extension of MDP, Partially Observable MDP (POMDP), is often applied to address the sensing uncertainties in robotic motion planning tasks [69, 136, 81]. Despite their wide application, most of them require discretization of the state space. Even for extensions that can handle continuous planning domains, tractability is still a common issue due to the need of partitioning or approximation of the continuous state space [91].

Another class of probabilistic planners formulates motion planning into an optimization problem through approaches such as Disjunctive Linear Program (DLP). [15] introduced a DLP-based approach that can perform obstacle avoidance under uncertainties, [16] described a Mixed Integer Linear Programming (MILP) formulation of the robust path planning problem which approximates chance constraints with a probabilistic particle-control approach, [91] proposed the probabilistic Sulu planner (p-Sulu) which performs goal-directed planning in a continuous domain with temporal and chance constraints, and [71] adopted trajectory optimization in belief space and formulated collision avoidance constraints using sigma hulls. However, since p-Sulu encodes feasible regions with linear constraint approximations, it inevitably suffers from the exponential growth of computation complexity when applied in complicated 3D environments or tasks with multiple agents. Additionally, both linear approximations and sigma hulls place restrictions on robot and environment geometry and also introduce inaccuracies in collision probability estimation.

Uncertainty-aware extensions of search-based [72, 22] and sampling-based [79, 18, 80, 78, 126] planners are also popular in the motion and path planning field. However, their applications are often limited to car-like robots in simplified environments due to their disadvantages in planning speed and collision probability estimation ability for high-DOF robots in real-world complex environments. When the robot has high dimensionality, the collision checking happens in the 3D workspace, whereas the motion planning happens in the high-dimensional C-space. Mapping the collision-free workspace into the C-space is nontrivial, which hence becomes another barrier for high-dimensional risk-aware motion planning.

2.1.1 Collision Risk Estimation

The estimation of trajectory collision probability has been widely investigated in the motion planning field, yet no perfect solution has been proposed due to its inherent difficulties. In order to approach this problem, many approximations have been used, including the discretization of time and the convexification of obstacles [37]. For low-dimensional planning tasks in convex environments, the estimation of collision risks at discrete waypoints is relatively straightforward. In the p-Sulu planner presented by [91], each boundary of each obstacle is formulated into a linear constraint, and the half-spaces that represent those linear constraints form the collision-free regions. In this way, the waypoint collision probability becomes the probability of violating any of the linear constraints, and can be solved through linear program (LP) solvers. A very different idea is to take advantage of confidence intervals, which are ellipses and ellipsoids for Gaussian distributions [135]. If the configuration space is 2D, then the maximum factor by which the elliptical confidence interval can be scaled before it intersects obstacles gives an indication of the collision probability at that configuration, where the scale factor can be computed as the Euclidean distance to the nearest obstacle in the environment. [104] further investigate this idea and account for the fact that the collision probability at each step along a trajectory is conditioned on the previous steps being collision-free. They propose that the *a priori* state probability distributions for different waypoints along a trajectory can be truncated to better reflect the actual collision probabilities.

However, it is nontrivial to extend the aforementioned approaches to high-dimensional planning tasks. Obstacles defined in workspace can not be directly mapped into a 6-DOF or 7-DOF C-space in closed form [25], hence the feasible region idea [91] and the confidence interval scaling idea [135] can not be easily applied. [127] pointed out a key relation between workspace geometry and C-space geometry: configuration $\mathbf{q}$ lies on the boundary of a C-space obstacle if and only if the workspace distance between the obstacle and the robot configured at $\mathbf{q}$ is zero. Based on this relation, [127] proposed an approach that looks for the point on the boundary of C-space obstacles that is closest to the robot's mean configuration by calculating the gradient of the workspace signed-distance field. Although this approach builds an important bridge between workspace obstacles and C-space obstacles, it relies on the assumption that the geometries of the C-space obstacles are locally convex. Since p-Chekov aims at solving high-dimensional motion planning problems in 3D complex environments where obstacles maintain their original non-convex shapes, we apply regression methods with function approximators to learn risk distributions through offline sampling and to make predictions during online planning queries.

2.1.2 Machine Learning in Motion Planning

Machine learning approaches are still not widely applied in robotic motion planning. Existing applications include guiding the exploration of sampling-based motion planners using nearest neighbor and adaptive sampling [7, 6, 60], accelerating collision detection through supervised classification [98, 97], and pursuing end-to-end motion planning through learning from demonstration [139, 108, 53]. To the author's best knowledge, learning-based p-Chekov is the first application of learning-based methods on the collision risk estimation problem for probabilistic motion planning systems. We explore the real-time collision risk estimation performance of different machine learning algorithms with different structures in chance-constrained motion planning tasks for robotic manipulators. It is shown that neural networks with appropriate structures can efficiently generate accurate predictions on collision risks in the environments they are trained in. The experiment results in this thesis show that p-Chekov with neural networks as collision risk estimation component performs significantly better than the quadrature-based p-Chekov [33] in terms of planning speed while maintaining a similar success rate.

2.2 Reinforcement Learning with Sparse Rewards

Reinforcement learning for sparse reward tasks has been been extensively studied from many different perspectives. Curriculum learning [14] tackles challenging exploration problems by introducing auxiliary tasks with gradually increasing difficulty in order to accelerate the learning progress. However, manually designing curricula can be tricky and tedious, and the automated curriculum generating methods often bias the exploration to a small subset of the tasks [125]. Through implicitly designing a form of curriculum to first achieve easily attainable goals and then progress towards more difficult goals, Hindsight Experience Replay (HER) is the first work that allows complicated manipulation behaviors to be learned from scratch with only binary rewards [4]. However, when the actual task goal is very distinct from what random policies can achieve, HER's effect is limited. As mentioned in [4], HER is unable to allow manipulators to learn pick-and-place tasks without using demonstration states during training.

Hierarchal reinforcement learning (HRL) approaches divide the policy into different levels in order to better concentrate on different tasks during learning [35, 68, 54]. Temporal abstraction has been applied in the option-critic architecture [9] to scale up learning and planning for temporally extended tasks, information asymmetry between different policy levels and probabilistic modeling of the objective function [47, 132, 51] have been utilized to introduce inductive biases for learning complicated tasks and transferable skills, mutual information maximization [92, 41] has been applied to find a diverse set of action modes, and frameworks that combine the learning processes of multiple different tasks through a high level task selection policy [116, 26] have also shown effectiveness for sparse reward learning tasks. However, many of the existing HRL approaches only enable the robot to accomplish one single task without the ability to transfer the skills to diverse task sets, and among the methods that automatically learn skills, it is still not clear whether a useful and diverse set of skills can be learned efficiently in complicated robotic manipulation tasks with sparse rewards.

Intrinsic exploration approaches, instead, augments the reward signals by adding task-agnostic rewards which encourage the agent to explore novel or uncertain states [100, 19, 65, 20]. Many approaches in the theme of intrinsic exploration have been proposed to alleviate the burden of reward engineering when training RL agents: visit counts and pseudo-counts [129] encourage the agent to explore states that are less visited; novelty-based approaches [100, 101] motivate the agent to conduct actions that lead to more uncertain results; reachability-based approaches [119] gives rewards to the observations outside of the explored states that take more environment steps to reach; diversity-driven approaches [41, 124] learn skills using a maximum entropy policy to allow for the unsupervised emergence of diverse skills; and information gain [83, 59, 64] encourages the agent to explore states that will improve its belief about the dynamics. Ensemble methods with randomized value functions [94, 93, 95] have also been applied to achieve efficient exploration in reinforcement learning. However, count-based and uncertainty-based exploration methods often can't distinguish between task-irrelevant distractions and task-related novelties, and the high computational complexity largely restricts the application of existing information-theoretic methods in practical robotic manipulation tasks. The empowerment-based intrinsic motivation approach proposed in this thesis falls under the category of information-theoretic intrinsic exploration, and we provide insight into reasonable approximations that can make the computation of information-theoretic quantities feasible when the state and action spaces are continuous and high-dimensional with complex mutual dependencies. Extensive experiment results demonstrate the effectiveness of these approximations as well as the superiority of the proposed approach over existing intrinsic exploration approaches in robotic manipulation scenarios.

2.3 Demonstration-Efficient Imitation Learning

The Automatic Curricula via Expert Demonstrations (ACED) approach proposed in this thesis achieves demonstration-efficient imitation learning by introducing a sequence of curricula from demonstration states. This section reviews literature related to three major aspects of the ACED approach: curriculum learning, learning from demonstration and states resetting.

2.3.1 Curriculum Learning

Curriculum learning [14] is a continual learning method that accelerates the learning progress by gradually increasing the task difficulty. It has seen success in many applications including language modeling [52], autonomous navigation [82, 61] and robotic manipulation [45]. However, many curriculum-based methods only involve a small and discrete set of manually generated task sequences as the curriculum, and existing automated curriculum generating methods often assume prior knowledge on how to manipulate the environment [140], or inherit the instability of adversarial methods and bias the exploration to a small subset of the tasks [44, 125]. [45] introduced the idea of automatically generating initial states closer to the goal state in order to speed up training, but inevitably face the challenge of infeasible randomly-generated initial states and can't be trivially extended to problems where the Euclidean distance to the goal is not a good indicator of task difficulty. In order to address these issues, we propose to use states from expert demonstrations as initial states to guarantee feasibility and provide more accurate indication of task difficulty.

2.3.2 Learning from Demonstration

Learning from demonstration (LfD) is widely used in tasks where the reward function is hard to define but demonstrations are relatively easier to obtain. Behavior cloning (BC) [109, 10] is a classical LfD approach that utilizes supervised-learning to train agents that imitate demonstration behaviors. Although BC has seen success in various fields including autonomous driving [11, 138] and robotics manipulation [111], it inevitably demands a large amount of demonstrations and its performance often suffers from data distribution mismatch [117]. Inverse reinforcement learning [86, 43] infers the reward function through demonstrations in order to avoid manual reward engineering, but it is fundamentally challenging due to its ambiguity in solutions since one trajectory can often be explained by many different reward functions [46]. LfD approaches based on Generative Adversarial Networks (GAN) [57, 5, 107] have effectively scaled to applications with relatively high-dimensional environments, but challenges due to unstable GAN training have significantly restricted their success in long-horizon tasks with complicated environments. Another popular approach to effectively utilize expert demonstrations is to combine LfD with RL [137, 63, 85]. The advantage of ACED is that it can easily be combined with many existing LfD methods for more efficient utilization of expert demonstrations, including BC, GAN-based methods [57] and methods that add demonstrations in RL replay buffers [137, 85]. In the empirical evaluation section in Chapter 6 in this thesis, we demonstrate the performance of our approach when combined with BC and show that it provides better convergence performance compared to using ACED only.

2.3.3 State Resetting

State resetting is widely used in RL for introducing expert knowledge, applying curricula or providing safety guarantees. State resetting techniques include initializing RL training episodes to specific reset states as well as enforcing intermediate interventions that reset the states during learning. It allows RL training algorithms to provide additional information for accelerating learning and to incorporate additional constraints that simplify the tasks. [4] and [85] reset some training episodes to states from expert demonstrations to simplify the exploration challenges in long-horizon tasks. [40] and [142] show that learning both the forward policy and the reset policy can not only reduce human effort in real-world robotics training but also accelerate training by automatically forming a curriculum. [134] introduces "teacher's interventions" via state resetting to avoid costly mistakes during learning in safety-critical applications. Similar to our proposed approach, [118] and [114] reset the initial states during training to demonstration states in order to form a sequence of curricula. However, [118] pointed out its limitations in terms of generalizing to unseen states and didn't provide evaluation on continuous control tasks or in-depth analysis on different component's influence on the overall performance, whereas [114] used a set of fixed rules for switching curricula instead of adapting it based on the agent's performance. In contrast to [118] and [114] which can only utilize one demonstration trajectory, ACED allows for arbitrary numbers of demonstrations through trajectory sectioning. In this thesis, we evaluate ACED on continuous control tasks and analyze the influence of the number of demonstration trajectories and the number of sections they are divided into on ACED's overall performance.

Chapter 3

Problem Statement

This chapter formally defines the three challenges tackled in this thesis. Section 3.1 defines chance-constrained motion planning, the problem tackled in Chapter 4. Section 3.2 defines reinforcement learning with sparse rewards, the challenge solved in Chapter 5. Section 3.3 defines curriculum learning through demonstrations, the challenge tackled in Chapter 6.

3.1 Chance-Constrained Motion Planning

We define a *disturbance* as an unexpected change to task goals, environment, or robot state. It may be due to an actual physical change, or a change in the estimated state of the environment or robot. Here we distinguish between *severe disturbances* and *small disturbances*. Severe disturbances refer to the ones that will cause significant and qualitative plan changes, such as changes of the planning goal, the movement of some obstacles that obstructs the original plan, or a strong external force that results in large deviations from the desired trajectory and the feedback controllers can't get the robot back on track due to actuation limits. On the other hand, small disturbances are mainly caused by process noises and observation noises, and the control inputs within limit can get the robot back to the desired trajectory. In practice, motion planners should account for the risk of potential plan failure caused by small disturbances, and react fast and naturally to severe disturbances which would necessitate plan adjustment.

Learning-based p-Chekov solves robotic motion planning problems under uncertainty with constraints on the success probability, considering temporal, spatial and dynamical constraints. Under process and observation noises, the collision rate during plan executions should not exceed a user specified chance constraint. The resulting motions should be locally optimal or near-optimal according to a specified objective function, which may optimize a variety of characteristics such as path length or control effort. Learning-based p-Chekov's real-time planning feature is key to providing robots the capability of operating safely and effectively in unstructured and uncertain environments.

3.1.1 Model Definition

Let $\mathcal{X} = \mathbb{R}^{n_x}$ denote the robot *state space* and $\mathcal{U} = \mathbb{R}^{n_u}$ the system *control input space*, where n_x and n_u are the dimensions of the state space and the control input space respectively. Consider a discretized series of time steps $t = 0, 1, 2, \ldots, T$ with a fixed time interval ΔT, where the number of time steps T is a finite integer. Let $\mathbf{x}_t \in \mathcal{X}$ denote the robot state at time step t. We assume applying a control input $\mathbf{u}_t \in \mathcal{U}$ at time step t will bring the robot from state $\mathbf{x}_t \in \mathcal{X}$ to $\mathbf{x}_{t+1} \in \mathcal{X}$, according to a given stochastic dynamics model:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \mathbf{m}_t), \quad \mathbf{m}_t \sim \mathcal{N}(0, M_t), \tag{3.1}$$

where $\mathbf{m}_t$ is the zero-mean Gaussian distributed process noise at time step t with a given covariance matrix M_t. $\mathbf{m}_t$ can be modeled based on the prior knowledge about robot controllers. Function f governs the robot dynamics and is assumed to be either linear or can be well approximated locally by its linearization.

The robot states are observed by taking a measurement at each time step t, denoted as $\mathbf{z}_t$. We assume that measurements are provided by noisy sensors according to a stochastic observation model:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{n}_t), \quad \mathbf{n}_t \sim \mathcal{N}(0, N_t), \tag{3.2}$$

where $\mathbf{n}_t$ is the zero-mean Gaussian distributed observation noise at time step t with a given covariance matrix N_t.

For each specific planning task, a start state $\mathbf{x}^{\text{start}}$ and a goal state $\mathbf{x}^{\text{goal}}$ or a convex goal region $\mathcal{X}^{\text{goal}}$ will be given. Let $\mathbf{x}_0 \in \mathcal{X}$ denote the initial state of the robot that follows a Gaussian distribution with mean $\mathbf{x}^{\text{start}}$ and covariance matrix $\mathbf{\Sigma}_{\mathbf{x}_0}$:

$$\mathbf{x}_0 \sim \mathcal{N}(\mathbf{x}^{\text{start}}, \mathbf{\Sigma}_{\mathbf{x}_0}). \tag{3.3}$$

An initial condition is defined as a combination of $\mathbf{x}^{\text{start}}$ and $\mathbf{\Sigma}_{\mathbf{x}_0}$. A trajectory Π is defined as a sequence of nominal robot states and control inputs $(\mathbf{x}^*_0, \mathbf{u}^*_0, \ldots, \mathbf{x}^*_T)$ that satisfies the deterministic dynamics model $\mathbf{x}^*_t = f(\mathbf{x}^*_{t-1}, \mathbf{u}^*_{t-1}, 0)$ for $0 < t \leq T$. We assume that an objective function $J(\Pi)$ will be specified for each planning task, which can implement planning goals such as minimizing trajectory length.

3.1.2 Constraint Definitions

A valid solution provided by learning-based p-Chekov should satisfy temporal constraints, chance constraints over collision risks, goal state constraints, control input constraints, and system dynamics constraints specified by the robot model. A temporal constraint defines an upper bound τ on the execution duration of a trajectory:

$$T \times \Delta T \leq \tau. \tag{3.4}$$

We assume a joint collision chance constraint with bound $\Delta_c \in [0, 1]$ will be given for each planning task, which specifies the allowed probability of collision failure. Let C_i denote the no-collision constraint for each obstacle $i = 1, \ldots, N$, then the probability of colliding with obstacle i is $\mathbb{P}(\overline{C_i})$. The collision chance constraint over an entire trajectory can then be expressed as:

$$\mathbb{P}\left(\bigvee_{i=1}^{N} \overline{C_i}\right) \leq \Delta_c. \tag{3.5}$$

The control input constraint requires that $\mathbf{u}_t^* \in \mathcal{U}, \forall t = 1, \ldots, T$. The system dynamics constraints require that the robot states at each time step along the trajectory are within the robot state space $\mathcal{X}$, and the state transitions between adjacent time steps satisfy the deterministic system dynamics model:

$$\mathbf{x}_t^* = f(\mathbf{x}_{t-1}^*, \mathbf{u}_{t-1}^*, 0) \in \mathcal{X}, \quad \forall t = 1, \ldots, T. \tag{3.6}$$

3.1.3 Problem Definition

Problem 1 defines the chance-constrained optimization problem solved by learning-based p-Chekov. It aims at finding a feasible trajectory Π that minimizes the given objective $J(\Pi)$ while satisfying the chance constraint and temporal constraint. The solution trajectory Π should satisfy the initial condition and the robot dynamics model, and the control inputs along the trajectory should fall into the control input space. If a C-space goal pose $\mathbf{x}^{\text{goal}}$ is given, the robot configuration at the final time step should be at $\mathbf{x}^{\text{goal}}$; on the other hand, if a convex goal region of the workspace end-effector pose $\mathcal{X}^{\text{goal}}$ is specified, then the end-effector should be in $\mathcal{X}^{\text{goal}}$ at the end of Π.

Problem 1.

$$
\begin{aligned}
\underset{\Pi}{\text{minimize}} \quad & J(\Pi) \\
\text{subject to} \quad & \mathbf{x}_0 \sim \mathcal{N}(\hat{\mathbf{x}}_0, \mathbf{\Sigma}_{\mathbf{x}_0}) \\
& \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \mathbf{m}_t), && 0 < t \leq T \\
& \mathbf{z}_t = h(\mathbf{x}_t, \mathbf{n}_t), && 0 < t \leq T \\
& \mathbf{m}_t \sim \mathcal{N}(0, M_t), && 0 < t \leq T \\
& \mathbf{n}_t \sim \mathcal{N}(0, N_t), && 0 < t \leq T \\
& \mathbf{x}_t \in \mathcal{X}, && 0 < t \leq T \\
& \mathbf{u}_t \in \mathcal{U}, && 0 < t \leq T \\
& \mathbf{x}_T^* = \mathbf{x}^{\text{goal}} \quad \text{or} \quad \mathbf{x}_T^* \in \mathcal{X}^{\text{goal}} \\
& \mathbb{P}\left(\bigvee_{i=1}^{N} \overline{C_i} \right) \leq \Delta_c \\
& T \times \Delta T \leq \tau
\end{aligned}
\tag{3.7}
$$

3.1.4 Assumptions

In order to achieve the promised performance of p-Chekov, several key assumptions are made. First, we assume that the collision environments are practical real-world environments that are not overly complex. Second, we assume that both the process noise and observation noise have Gaussian distribution with diagonal covariance matrices. This assumption is reasonable because in real-world scenarios, noises often accumulate from inconsistent, random sources, and based on the Central Limit Theorem [58], Gaussian models are appropriate in these cases. Correlated noise components can also be transformed through robot states space coordinate transformation so that the covariance matrices will become diagonal. With this assumption, the optimal performance of Kalman filter will be guaranteed [48] and the requirement of Linear-quadratic Gaussian (LQG) control will be satisfied. Third, we assume that both the system dynamics model and the observation model are either linear or can be well approximated locally by their linearizations. In real-world executions, robot motions will be controlled to closely follow the planned trajectory, thus local linearizations of the non-linear dynamics are good approximations for robot motions.

3.2 Reinforcement Learning with Sparse Rewards

Reinforcement Learning (RL) is a computational approach that solves sequential decision making problems by interacting with the environment. In this thesis, we model sequential decision making problems as fully observable Markov decision processes.

3.2.1 Markov Decision Process (MDP)

An MDP is discrete-time stochastic control process that models decision making problems in which the outcomes are influenced by the actions of the decision maker. In RL, the goal of the learner is to learn how to best select actions in an MDP in order to optimize an objective function regarding the outcomes. In MDPs, the learner or decision maker is called the *agent*, and the system it interacts with, comprising everything outside the agent, is called the *environment*. The interaction between the agent and the environment happens in a sequence of discrete time steps: $t = 0, 1, 2, \cdots$. A typical MDP consists of four elements [128]:

1. State: $\mathbf{s} \in \mathcal{S}$, where $\mathcal{S}$ is the state space.

2. Action: $\mathbf{a} \in \mathcal{A}$, where $\mathcal{A}$ is the action space.

3. Transition function: $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. $P_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = \mathbb{P}(\mathbf{s}_{t+1} = \mathbf{s}'|\mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a})$ denotes the probabilistic mapping from the state and action at time step t to the state at time step $t+1$. P specifies a probability distribution that satisfies $\int_{\mathbf{s}' \in S} P_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = 1$

4. Reward: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is typically a nonnegative scalar. $r_t(\mathbf{s}, \mathbf{a})$ denotes the reward the agent receives from the environment when conducting action $\mathbf{a}$ at state $\mathbf{s}$ at time step t. In this thesis, we only consider stationary reward

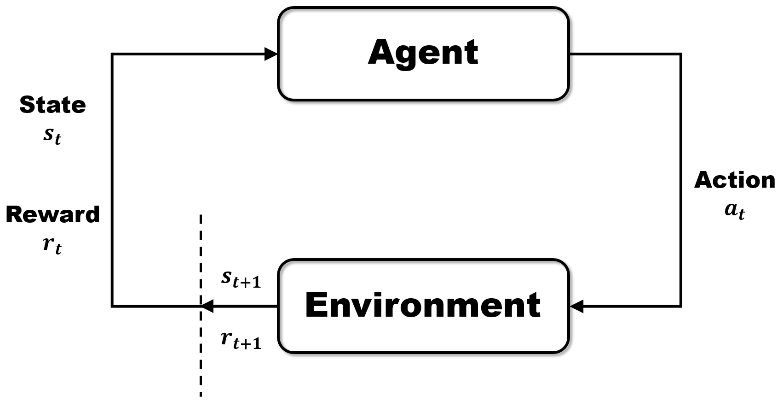


Figure 3-1: Agent-environment interaction in an MDP.

functions, hence $r_t(\mathbf{s}, \mathbf{a}) \rightarrow r(\mathbf{s}, \mathbf{a})$. We instead use r_t to denoted the specific reward scalar received at time step t.

The state transitions of an MDP satisfy the Markov property, meaning that given $\mathbf{s}_t$ and $\mathbf{a}_t$, $\mathbf{s}_{t+1}$ is conditionally independent of the states and actions at all previous time steps. Figure 3-1 demonstrates the agent-environment interaction in a typical MDP.

3.2.2 Reinforcement Learning (RL)

A *policy* is defined as the mapping from states to the probabilities for selecting actions. If the agent is following policy π at time t, then $\pi_t(\mathbf{a}|\mathbf{s})$ is the probability that $\mathbf{a}_t = \mathbf{a}$ when $\mathbf{s}_t = \mathbf{s}$. We only consider stationary policies in this thesis, hence $\pi_t \rightarrow \pi$. The goal of RL is to find a policy π that maximizes an objective function:

$$J = \mathbb{E}_\pi[\sum_\tau r(\mathbf{s}_t, \mathbf{a}_t)|\mathbf{a}_t \sim \pi(\mathbf{s}_t), \mathbf{s}_0 \sim p_0(\mathbf{s})], \tag{3.8}$$

where τ denotes the trajectory, $p_0(\mathbf{s})$ denotes the initial state distribution.

In RL, it is common to divide the agent-environment interactions into subsequences. For example, a common practice in RL for robotics is to let the robot interact with the environment for a number of steps, reset the robot and the environment both to the initial state, and start the interaction sequence again. In this thesis, we refer to each of these subsequences as an *episode*. Usually there are two ways to determine the end of an episode: 1) specify a maximum number of time steps and terminate the episode once the maximum step is reached; 2) specify a set of terminal states and terminate the episode once a terminal state is reached. In this thesis, we use a combination of both termination strategies, i.e. the episode is terminated if the maximum time step is reach or a terminal state is reached. The *value function* of a state $\mathbf{s}$ under a policy π, denoted as $v_\pi(\mathbf{s})$, is the expected return when starting from state $\mathbf{s}$ and following policy π until the end of the episode.

This thesis focuses on RL tasks that learn in an episodic manner, i.e. *episodic tasks*. In episodic tasks, the environment and the agent are both reset to a state drawn from an initial state distribution specified by the task at the beginning of each episode, and the performance of the RL agent is evaluated in terms of the *return* of each episode. In this thesis, we consider the undiscounted return defined as follows:

$$R = \sum_{t=1}^{t=T} r_t, \tag{3.9}$$

where T denotes the terminal time step of the episode. In Chapter 5, the performance of RL agents is evaluated using *average return*, the mean of the returns of all parallel training episodes.

3.2.3 Intrinsic Motivation

The central idea of RL is to learn by trial-and-error. In contrast to supervised learning, RL agents are not provided with labeled data and they have to obtain the training data by interacting with the environment and observing the reward signals. This has caused a major challenge in RL: the trade-off between exploration and exploitation. *Exploration* refers to taking actions with the intention of probing a large portion of the state space in order to find promising solutions to the task, whereas *exploitation* refers to probing a limited (but promising) region of the search space with the hope of improving a promising solution that the agent already have at hand. In RL problems with high-dimensional continuous state and action spaces, the exploration challenge has been a major issue researchers face because exhausting the entire state space is infeasible and intelligent exploration strategy is necessary for RL agents to discover any solution to the task.

In this thesis, we focus on RL problems with high-dimensional continuous state and action spaces and *sparse reward functions*. For a reward function to be sparse, it needs to only provide a non-zero reward scalar for a very small portion of the state and action space, whereas for the vast majority of state-action pairs, the RL agent only receives zero reward. This is to be distinguished from *dense reward functions*, where most of the states involve non-zero rewards. A *binary reward function* is an example of sparse reward functions. A binary reward function only takes two values: 0 or 1. It only provides $r = 1$ when a goal state is reached, and gives $r = 0$ for all other states:

$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} 0, & \text{if } \mathbf{s}_t \notin S_g, \\ 1, & \text{if } \mathbf{s}_t \in S_g, \end{cases} \tag{3.10}$$

where S_g is a set of desired goal states. Specifying sparse reward functions is usually the intuitive way to define a task, and dense reward functions are often designed through reward shaping. In the robotic manipulation environment we discuss in this thesis, an example of a sparse reward function is one that only provides the robot rewards when the objects reach their corresponding goal states, whereas a dense reward function can include the distance from the robot gripper to the objects and the distance between the objects and their corresponding goal states. Such dense reward functions are very task-specific and are considered as a form of reward shaping.

A popular approach for handling challenging exploration problems is using intrinsic motivation. *Intrinsic motivation* refers to exploration strategies that provide the RL agent with auxiliary intrinsic rewards that are unrelated to the task to be accomplished in order to explore the environment more effectively and to ultimately lead to task success. The inspirations for intrinsic motivation methods often come from the human instinct in psychology to learn and to explore regardless of whether external rewards are given.

The distinction between intrinsic rewards and extrinsic rewards is key to the problem studied in this thesis. *Extrinsic rewards* refer to the MDP reward defined in Section 3.2.1, i.e. the actual rewards received from the environment. Extrinsic rewards typically depend on the desired task to be solved, hence they are also often referred to as the task rewards. The RL objective only maximizes the expected extrinsic rewards along the trajectory, whereas the intrinsic rewards are purely auxiliary rewards to help the agent achieve the ultimate objective. *Intrinsic rewards* are task-agnostic, meaning that the same intrinsic reward can be used for many different target tasks in the environment. In order to implement desired exploration strategies, the objective function is often modified during optimization in order to include intrinsic rewards, but only the original objective function without intrinsic rewards should be considered when evaluating the RL agent's performance in target tasks.

In this thesis, we refer to the reward from the environment as extrinsic reward r^e and the artificial reward from the algorithm as intrinsic reward r^i, hence the reward used when computing value functions during the optimization process is $r = r^e + r^i$. The sum r is used during the learning process, whereas only r^e is considered when evaluating the performance of a learning algorithm.

3.2.4 Problem Definition

The problem solved by Chapter 5 Empowerment-based Intrinsic Motivation is a robotic manipulation task formulated as an episodic RL problem where the environment is modeled as a fully-observable MDP and the extrinsic reward function is sparse. It searches for a policy π that optimizes the expectation of the undiscounted episode return:

$$J = \mathbb{E}_{\pi}[\sum_{\tau} r^e(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{a}_t \sim \pi(\mathbf{s}_t), \mathbf{s}_0 \sim p_0(\mathbf{s})]. \tag{3.11}$$

The goal of Chapter 5 is to find a form of intrinsic reward r^i so that by maximizing the auxiliary objective function:

$$J_a = \mathbb{E}_\pi[\sum_\tau (r^e(\mathbf{s}_t, \mathbf{a}_t) + \beta r^i(\mathbf{s}_t, \mathbf{a}_t))|\mathbf{a}_t \sim \pi(\mathbf{s}_t), \mathbf{s}_0 \sim p_0(\mathbf{s})], \tag{3.12}$$

the optimization of the original objective function J can be made much easier. Here β denotes the intrinsic reward coefficient.

3.3 Curriculum Learning through Demonstrations

The problem tackled in Chapter 6 is also an episodic RL problem where the environment is modeled as a fully-observable MDP, hence we won't be repeating concepts that are already defined in Section 3.2. We instead focus on the definition of curriculum learning and learning from demonstration in this section, and present the problem statement for the challenge tackled in Chapter 6.

3.3.1 Curriculum Learning

In RL, curriculum learning often refers to the training strategy that divides the entire training process into multiple stages and presents tasks of increasing difficulties as the learning agent moves through different stages. We denote the target RL task as T_0. In this thesis, we define a *curriculum* as a class of tasks with similar difficulties when presented to the same learning agent, and we use *curriculum numbers* C to index these curricula. We use increasing curriculum numbers to represent curricula with increasingly difficult tasks, and we refer to a task drawn from curriculum-C as T_C. Although each individual curriculum can have different tasks goals and initialization distributions, task across all curricula should be related to the same target task T_0. Ideally, training on these curriculum tasks should provide guidance for the learning agent in terms of how to solve the target task T_0, and agents who have mastered the curriculum tasks should find T_0 easier than those who are learning from scratch.

3.3.2 Learning from Demonstration

Learning from demonstration (LfD) is a broad class of approaches that utilizes demonstration trajectories to facilitate learning. In this thesis, we focus only on the LfD methods that uses demonstrations to assist RL. We define a state-only demonstration trajectory as a sequence of states at consecutive time steps generated by a rollout of an MDP, i.e. $\tau = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{T-1}, \mathbf{s}_T)$, where T denotes the terminal time step. Similarly, we define a state-action demonstration trajectory as a sequence of state-action pairs at consecutive time steps generated by a rollout of an MDP, i.e. $\tau = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{s}_T)$. In this thesis, the rewards associated with the demonstration trajectories are not used. The demonstration trajectories can be generated either by a human or a robot, and they need to solve the target task T_0 in a reasonable way without deliberate detours. The demonstration trajectories do not need to be optimal solutions to the target task T_0.

3.3.3 Problem Definition

The problem solved by Chapter 6 Automatic Curricula via Expert Demonstrations is a robotic manipulation task T_0 formulated as an episodic RL problem where the environment is modeled as a fully-observable MDP with a binary reward function. It searches for a policy π that optimizes the expectation of the undiscounted episode return:

$$J = \mathbb{E}_\pi[\sum_\tau r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{a}_t \sim \pi(\mathbf{s}_t), \mathbf{s}_0 \sim p_0(\mathbf{s})], \tag{3.13}$$

where τ denotes the rollout trajectory. The target task T_0 is challenging so that vanilla RL algorithms are not able to solve it. The goal of Chapter 6 is to extract a set of curriculum tasks $T_1, \cdots, T_C, \cdots$ from a small amount of demonstration trajectories so that pre-training the RL agents on these curriculum tasks can make the original target task T_0 solvable in an efficient manner. We consider two different types of demonstration trajectories in Chapter 6: 1) demonstration trajectories with only states and no actions, i.e. $\tau_e = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_t, \ldots, \mathbf{s}_{T-1}, \mathbf{s}_T)$; 2) demonstration trajectories with state-action pairs, i.e. $\tau_e = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T)$.

Chapter 4

Learning-based P-Chekov

This chapter presents the first contribution of this thesis: learning-based p-Chekov. Section 4.1 briefly reviews sampling-based p-Chekov, the prior work learning-based p-Chekov improves upon and compares with. Section 4.2 formally introduces the learning-based p-Chekov approach, Section 4.3 describes the setup for empirical experiments and Section 4.4 presents the empirical evaluation results.

4.1 Preliminaries: Sampling-based P-Chekov

4.1.1 The P-Chekov Framework

P-Chekov, originally introduced in [29], is a risk-aware motion planning and execution system that accounts for the potential uncertainties during execution while making plans and returns solutions that can satisfy user-specified chance constraints over plan failure. As mentioned in Section 1.2.1, it targets high-dimensional robots such as robotic manipulators and humanoid robots in environments where collisions can cause disastrous outcomes. It is composed of a planning phase and an execution phase, as shown in Figure 4-1. In this thesis, we mainly focus on the planning phase and will briefly review the execution phase Iterative Risk Allocation (IRA) procedure. We refer to this original version of p-Chekov as the sampling-based p-Chekov. The p-Chekov planning phase includes three major components: nominal

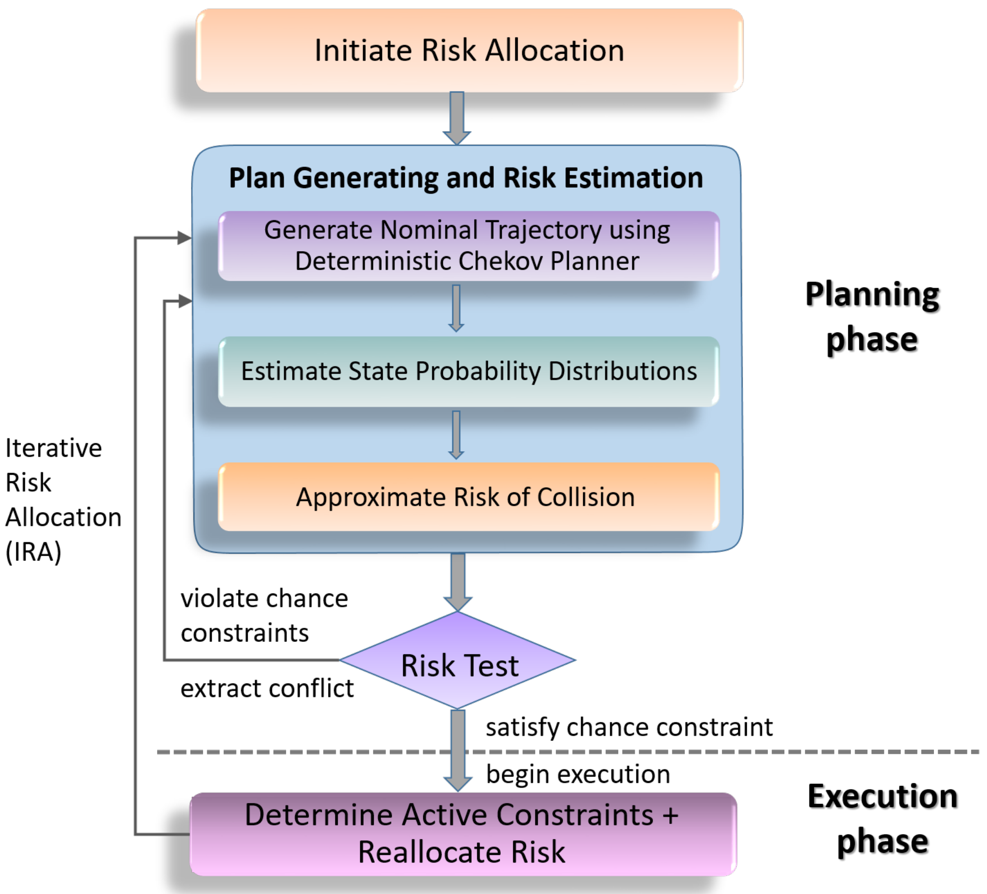

Initiate Risk Allocation
Plan Generating and Risk Estimation
Generate Nominal Trajectory using Deterministic Chekov Planner
Estimate State Probability Distributions
Approximate Risk of Collision
Planning phase
Iterative Risk Allocation (IRA)
violate chance constraints
extract conflict
Risk Test
satisfy chance constraint
begin execution
Determine Active Constraints + Reallocate Risk
Execution phase

Figure 4-1: System diagram for p-Chekov [33]

trajectory generation, state distribution estimation, and collision risk estimation. The main difference between learning-based p-Chekov and sampling-based p-Chekov is the collision risk estimation component.

The goal in the planning phase is to find a feasible solution trajectory along which the estimated risk of collision is smaller than or equal to the given joint chance constraint Δ_c. In p-Chekov, time is discretized into fixed-interval time steps, and the collision risk at each waypoint is considered separately through risk allocation. Risk allocation decomposes a joint chance constraint Δ by allocating risk bounds δ_i to individual constraints, where $\sum_1^N \delta_i = \Delta$. When the planning phase starts, p-Chekov first uniformly distributes the joint chance constraint into the allowed collision risk bounds for each waypoint along the trajectory. Provided with a risk allocation, p-Chekov then uses the deterministic Chekov approach introduced in [32] to generate a nominal trajectory that is feasible and collision-free under deterministic dynamics. Given the estimated model of controller and sensor noises during execution, p-Chekov then computes the *a priori* probability distribution of robot states along this nominal trajectory. With this state distribution information, p-Chekov estimates the probability of collision at each waypoint along this trajectory through a quadrature-based sampling approach and compares the allocated risk bound with the estimated probability of collision, shown as the "risk test" step in Figure 4-1. If the nominal trajectory fails to pass the risk test, the robot configurations at the waypoints where the estimated risk of collision exceeds the allocated risk bound will be viewed as conflicts. Before p-Chekov goes back to the "plan generating and risk estimation" stage in Figure 4-1, constraints associated with the conflict configurations and conflict waypoints [121] will be added so that collision costs will be triggered when the planner tries to get to these configurations and waypoints, and deterministic Chekov can be guided to avoid these configurations and waypoints while finding safer nominal trajectories.

When the solution trajectory passes the risk test, p-Chekov will transition to the execution phase, where it optimizes the solution it found in the planning phase while the robot is executing the trajectory based on the iterative risk allocation (IRA) algorithm [89, 29]. Through gradually reallocating the risk from inactive constraints to active constraints, IRA provides less conservative risk allocations and allows for higher quality motion plans. After that, p-Chekov will go back to the "plan generating and risk estimation" stage with zero penalty hit-in distance [121], i.e. the collision cost will hit in only when the configuration reaches the obstacles without using any buffer, and find a new feasible solution which satisfies the new risk allocation. When it finds a valid plan, the robot will keep executing based on the updated plan. This risk reallocation and plan refinement process is conducted iteratively, which will help the planner to converge to a locally optimal solution if given enough number of iterations.

4.1.2 Quadrature-based Collision Probability Estimation

This section briefly reviews how collision risks are estimated in sampling-based p-Chekov. The main contribution in learning-based p-Chekov is to replace this quadrature sampling-based collision risk estimation component with a learning-based version which significantly improved p-Chekov's real-time planning performance, as shown in 4.2.

Given the state probability distribution around a nominal configuration, the collision probability can be approximated by sampling from this distribution and checking the percentage of configurations that are in collision. However, as with all Monte Carlo methods, this approach would suffer from inaccuracy when the sample size is small and high computational cost when the sample size is large. [29] proposed a quadrature-based sampling method that can closely approximate the collision probability with only a small number of samples.

A Monte Carlo collision probability estimation approach is essentially estimating the expectation of a collision function:

$$c(\mathbf{x}_t) = \begin{cases} 0, & \text{if } \mathbf{x}_t \text{ is collision free} \\ 1, & \text{if } \mathbf{x}_t \text{ is in collision} \end{cases}$$

along the distribution $\mathbf{x}_t \sim \mathcal{N}(\hat{\mathbf{x}}_t, \mathbf{\Sigma}_{\mathbf{x}_t})$, where $\mathbf{x}_t \in \mathbb{R}^{n_x}$ is the nominal configuration at time step t. Since expectations can be written as integrals, non-random numerical integration methods (also called quadratures [55]) can be applied to solve this problem. Assume $\mathbf{x}_t$ is d-dimensional and let x_t^i denote its ith component whose distributions are independent from each other. This assumption is reasonable because correlated noise components can be transformed through robot state space coordinate transformation so that the covariance matrices will become diagonal. Since $\mathbf{x}_t$ is Gaussian distributed, we can write $x_t^i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. Then, based on the conditional distribution rule of multivariate normal distribution [39], the probability density function of $\mathbf{x}_t$ can be expressed as:

$$
\begin{aligned}
&p(\mathbf{x}_t) = p(\mathbf{x}_t^{1:d}) = p(x_t^1)p(\mathbf{x}_t^{2:d}|x_t^1) = p(x_t^1)p(\mathbf{x}_t^{2:d}),\\
&x_t^1 \sim \mathcal{N}(\mu_1, \sigma_1^2),\\
&\mathbf{x}_t^{2:d} \sim \mathcal{N}(\mu_{2:d}, \mathbf{\Sigma}_{2:d}),
\end{aligned}
\tag{4.1}
$$

where $\mu_{2:d}$ and $\mathbf{\Sigma}_{2:d}$ denote the mean and variance of $\mathbf{x}_t^{2:d}$ respectively. Then we can write the expectation of the collision function as:

$$
\mathbb{E}(c(\mathbf{x}_t)) = \int_{-\infty}^{\infty} p(x_t^1) \int_{\mathbb{R}^{n_x-1}} p(\mathbf{x}_t^{2:d})c(\mathbf{x}_t)d\mathbf{x}_t^{2:d}dx_t^1. \tag{4.2}
$$

Let $g(x_t^1) = \int_{\mathbb{R}^{n_x-1}} p(\mathbf{x}_t^{2:d})c(\mathbf{x}_t)d\mathbf{x}_t^{2:d}$ and apply the probability density function of Gaussian distributions, we have:

$$
\begin{aligned}
\mathbb{E}(c(\mathbf{x}_t)) &= \int_{-\infty}^{\infty} p(x_t^1)g(x_t^1)dx_t^1\\
&= \int_{-\infty}^{\infty} \frac{1}{\sigma_1\sqrt{2\pi}} \exp\Big(-\frac{(x_t^1-\mu_1)^2}{2\sigma_1^2}\Big)g(x_t^1)dx_t^1.
\end{aligned}
\tag{4.3}
$$

Gauss-Hermite quadrature approximates the value of an integral by calculating the weighted sum of the integrand function at a finite number of reference points, i.e.

$$
\int_{-\infty}^{\infty} e^{-y^2}h(y)dy \approx \sum_{j=1}^{n} w_j h(y_j), \tag{4.4}
$$

where n is the number of sampled points, x_j are the roots of the Hermite polynomial $H_n(x)$ and the associated weights w_j are given by [2]:

$$
w_j = \frac{2^{n-1}n!\sqrt{\pi}}{n^2[H_{n-1}(y_j)]^2}. \tag{4.5}
$$

A quadrature rule with n sampled points is called a n-point rule.

$\mathbb{E}(c(\mathbf{x}_t))$ in its form in Equation 4.3 still doesn't correspond to the Hermite polynomial, therefore we conduct the following variable change:

$$
y_1 = \frac{x_t^1 - \mu_1}{\sqrt{2}\sigma_1} \Leftrightarrow x_t^1 = \sqrt{2}\sigma_1 y_1 + \mu_1. \tag{4.6}
$$

Applying Equation 4.6 to Equation 4.3 yields:

$$\mathbb{E}(c(\mathbf{x}_t)) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} e^{-(y_1)^2} g(\sqrt{2}\sigma_1 y_1 + \mu_1) dy_1. \tag{4.7}$$

If we iteratively conduct this Gauss-Hermite quadrature approximation procedure from x_t^1 through x_t^d, we will be able to approximate the value of $\mathbb{E}(c(\mathbf{x}_t))$ through:

$$\begin{aligned}\mathbb{E}(c(\mathbf{x}_t)) \approx \pi^{-\frac{d}{2}} \sum_{j_1=1}^{n_1}\sum_{j_2=1}^{n_2}\cdots\sum_{j_d=1}^{n_d}\left(\prod_{i=1}^{d} w_{i,j_i}\right) g(\sqrt{2}\sigma_1 y_{1,j_1} \\ + \mu_1, \sqrt{2}\sigma_2 y_{2,j_2} + \mu_2, \ldots, \sqrt{2}\sigma_d y_{d,j_d} + \mu_d).\end{aligned} \tag{4.8}$$

This sampling-based approach of estimating the collision probability based on Gauss-Hermite quadrature theory is summarized in Algorithm 1.

Algorithm 1: GHCollisionProbabilityEstimation

```
Input:
$\Pi$: desired trajectory
$\mathcal{D}$: robot state distribution along desired trajectory
$\mathcal{R}, \mathcal{E}$: robot and environment collision models respectively
$dof$: robot degrees of freedom
$n$: number of samples used in quadrature rule
$l_u, l_l$: upper and lower limits of active joints respectively
Output:
$\mathbf{r}$: collision risk at each waypoint along desired trajectory
1  Initialize $\mathbf{r}$ to a list of zeros
2  for $i = 1, 2, \ldots$, len($\Pi$) do
3      Initialize $NodeList$ to an empty set
4      for $d = 1, 2, \ldots, dof$ do
5          $(\mu, \sigma) \leftarrow \mathcal{D}[i, d]$      /* Draw from $\mathcal{D}$ at the $i$th waypoint $d$th joint */
6          $(nodes, weights) \leftarrow$ QuadratureSampling($\mu, \sigma, n$)
7          for $node$ in $nodes$ do
8              if $node > l_u[d]$ then $node \leftarrow l_u[d]$
9              if $node < l_l[d]$ then $node \leftarrow l_l[d]$
10         end
11         Append $(nodes, weights)$ to $NodeList$
12     end
13     Estimate $\mathbf{r}(i)$ by taking nodes from $NodeList$, checking collision with $\mathcal{E}, \mathcal{R}$, and
         averaging the collision number
14 end
```

In one-dimensional space, a n-point rule yields $2n$ parameters and it is possible to integrate polynomials of degree up to $2n-1$ without error. For $a < x < b$ and $h(x)$

with $2n$ continuous derivatives, the error in a Gauss rule is:

$$\frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3}h^{(2n)}(x). \tag{4.9}$$

Note that although quadrature methods are well tuned to one-dimensional problems, extending them to multi-dimensional problems through iterated one-dimensional integrals still can't escape the "curse of dimensionality" [13]. The result of a d-dimensional quadrature rule can not be better than the worst of the rules we use in each dimension. If we use the same n-point one-dimensional quadrature rule for each of the d-dimensions, then we need $N = n^d$ function evaluations. If the one-dimensional rule has error $O(n^{-r})$, then the combined rule has error

$$|\hat{I} - I| = O(n^{-r}) = O(N^{-r/d}). \tag{4.10}$$

Even a modestly large d can give a very inaccurate result [96]. Additionally, the collision function $c(\mathbf{x}_t)$ p-Chekov needs to evaluate is not smooth, which adds to the inaccuracy of the approximations through this quadrature-based sampling method. Consequently, this quadrature-based collision probability estimation approach is a relatively rough one.

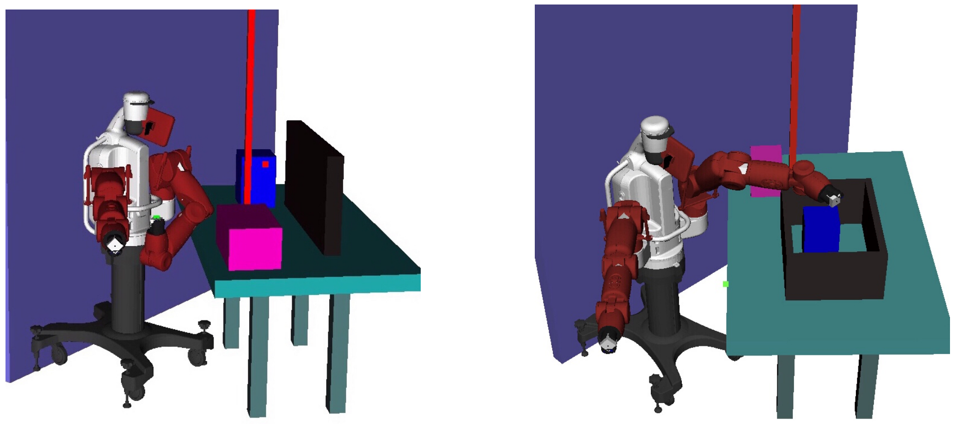
4.2 Learning-based P-Chekov

The quadrature-based sampling approach introduced in Section 4.1.2 mitigates the inaccuracy of random sampling and avoids the difficulty of mapping between the configuration space (C-space) and the workspace, but its computation time in high-dimensional planning space still obstructs its application in real-time motion planning tasks. Even though only two quadrature nodes per dimension are used in [29] to estimate the collision risk for each waypoint, the total number of collision tests conducted online is still very big since the manipulator have 7 DOFs ($2^7 \times n_{waypoints}$ collision tests for each nominal trajectory). Additionally, two-node quadratures have very limited ability of approximating non-smooth functions, whereas the collision functions here are highly non-smooth. Therefore, quadrature-based p-Chekov inevitably suffers from errors when approximating the collision risk, and the efficiency and accuracy of risk estimation becomes its bottleneck that restricts its application in uncertainty-sensitive real-time manipulation planning tasks. In this thesis, we introduce supervised learning approaches into the collision risk estimation component of p-Chekov in order to improve its efficiency and accuracy [31].

We hypothesize that if we take enough samples containing nominal configurations with their probability distributions and risks of collision from the environment that the robot will be interacting with in order to train a regression model offline, then this model can act as the "Approximate Risk of Collision" component in Figure 4-1 in the online planning phase which makes accurate predictions given a nominal trajectory and the state distributions outputted by the "Estimate State Probability Distributions" component. In this thesis, we evaluate this hypothesis in two tabletop environments originally introduced in [32] (shown in Figure 4-2). 60000 data points are collected in each of the tabletop environments, each of which contains a nominal joint configuration that is randomly sampled from the uniform distribution defined by the manipulator's joint limit, a randomly sampled standard deviation whose range is decided according to the real experiment data from quadrature-based p-Chekov tests, and a collision risk scalar that is viewed as the "ground-truth" risk associated with this configuration distribution. This collision risk is estimated using a simple Monte Carlo method: randomly sample 100000 nodes from the Gaussian distribution defined by the nominal joint configuration and the standard deviation, and compute the average collision rate. The nominal configuration together with its standard deviation forms the input vector to the regression algorithm, and the collision risk is its label.

We compare the performance of three different classes of regressors in the Scikit Learn [105] package (kernel ridge regressor, random forest regressor, and Gaussian process regressor) as well as neural networks through the Keras [24] interface with TensorFlow [1] back engine. Kernel ridge regression combines ridge regression (linear least squares with $l2$-norm regularization) with the kernel trick [84]. It can operate in a high-dimensional, implicit feature space without ever computing the coordinates of

Figure 4-2: Simulation Environments for Learning-based P-Chekov Evaluation [32]



(a) The “tabletop with a pole” environment

(b) The “tabletop with a container” environment

the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. It thus learns a linear function in the space induced by the respective kernel and the data. For non-linear kernels, this corresponds to a non-linear function in the original space. In kernel ridge regression, the objective to minimize is:

$$J = ||y - w^T X||^2 + \alpha ||w||^2, \tag{4.11}$$

where X is the input vector, y is the true label, w is the weight vector given by the regressor and α is the parameter that determines the regularization strength.

In the kernel ridge regression tests in this chapter, the performance of three different classes of kernels are compared: radial basis function (RBF) kernel, polynomial kernel and Matern kernel. In RBF kernels, each element in the kernel matrix between datasets X and Y is computed by:

$$K(x, y) = exp(-\gamma ||x - y||^2) \tag{4.12}$$

for each pair of rows x in X and y in Y. Therefore, the parameter γ represents how far the influence of a single training example reaches, with low values meaning “far” and high values meaning “close”. In polynomial kernels, *degree* is a parameter that represents the order of polynomials used in the kernel. A $degree-d$ polynomial kernel is defined as:

$$K(x, y) = (x^T y + c)^d, \tag{4.13}$$

where $c \geq 0$ is a free parameter trading off the influence of higher-order versus lower-order terms in the kernel. Matern kernel is defined by:

$$K(x, y) = \frac{1}{2^{\nu-1}\Gamma(\nu)}(\frac{2\sqrt{\nu}||x-y||}{\theta})^{\nu} H_{\nu}(\frac{2\sqrt{\nu}||x-y||}{\theta}), \tag{4.14}$$

where the length scale parameter θ is similar to the γ in RBF kernels, Γ is the Gamma function, the ν parameter controls the smoothness of the learned function, and H_ν is the modified Bessel function of the second kind of order ν. When ν approaches infinity, the Matern kernel becomes equivalent to the RBF kernel, and when $\nu = 0.5$ it's equivalent to the absolute exponential kernel.

Random forest regression [75] constructs an ensemble of decision trees using a different bootstrap sample of the data for each tree (also called bagging), and selects a random subsets of the features at each candidate split in the decision tree learning process. These two main strategies help random forest correct for decision trees' habit of overfitting to their training set and make accurate predictions very quickly during test time. Gaussian process regression [112] defines a collection of random variables, any finite number of which have a joint Gaussian distribution, and then conducts probabilistic inference directly in the function space. It can capture the model uncertainty directly and allows users to add prior knowledge by selecting different kernel functions. Here we choose to use Matern kernels in the Gaussian process regression tests.

Artificial neural network is another powerful tool for conducting supervised regression on large datasets. Section 4.4 compares the performance of different regression methods and shows that neural networks with appropriate configurations have the best performance in this collision risk regression task, thus we apply them to p-Chekov and compare learning-based p-Chekov's performance with the quadrature-based p-Chekov described in Section 4.1.

4.3 Experiment Setup

4.3.1 Environments

We evaluate learning-based p-Chekov in the two tabletop environments shown in Figure 4-2. The "tabletop with a pole" environment, shown in Figure 4-2-(a), is a simple tabletop pick-and-place task environment, with a slender pole in the middle of the table and a box on each side of the pole. The "tabletop with a container" environment is similar, but with a large container on the table with boxes both inside and outside of it, as shown in Figure 4-2-(b). For each environment, we generate 500 feasible planning queries by randomly sampling start and target end-effector pose pairs that are collision-free and kinematically feasible. Note that the second tabletop environment not only has the narrow spaces inside the container which are difficult for chance-constrained motion planners, but also include difficult test cases where the robot joints are close to their limits. For each experiment trial, planners are provided with the starting C-space position and the goal end-effector pose. We specify the goal in workspace to give planners the opportunity to find different C-space solutions to the planning problem. We have ensured that all test queries have a feasible solution by executing five different motion planners on each test case, and re-sampling start and goal poses when no planner could find a solution. The Baxter robot [115] with its 7-DOF left manipulator is used as the experiment testbed. Baxter's specification indicates that its worst case accuracy of joints is ± 0.25 degree, which is about ± 0.0044 rad. Controller accuracy for industrial robots can vary, and we use Baxter in this chapter to demonstrate that our approach can be effective even on robots with high uncertainties. Hence in the experiments in this chapter, the standard deviation of noises during execution is set to 0.0044 rad. All the experiments shown in this paper are conducted on a 10-core Intel i7 3.0 GHz desktop with 64 GB RAM.

4.3.2 Modeling

We simplify system dynamics into a discrete-time linear dynamics model and use accelerations as control inputs at each time step. Note that in highly dynamics robots like quadcopters, torque inputs would be more appropriate than acceleration inputs. However, since this work mainly targets at robotic manipulators, a kinematics model with acceleration inputs is sufficient. We express the system model in terms of the deviations from the desired trajectory $\Pi = (\mathbf{x}_0^*, \mathbf{u}_0^*, \ldots, \mathbf{x}_T^*, \mathbf{u}_T^*)$:

$$\begin{aligned} \bar{\mathbf{x}}_t &= \mathbf{x}_t - \mathbf{x}_t^*, \\ \bar{\mathbf{u}}_t &= \mathbf{u}_t - \mathbf{u}_t^*, \\ \bar{\mathbf{z}}_t &= \mathbf{z}_t - h(\mathbf{x}_t^*, 0), \end{aligned} \tag{4.15}$$

where the state $\mathbf{x}_t$ includes the joint position and velocity, and the input $\mathbf{u}_t$ includes the joint acceleration. Since robot motions will be controlled to closely follow the planned trajectory during execution, it is reasonable to linearize the system dynamics model and observation model as:

$$\begin{aligned} \mathbf{x}_t =& f(\mathbf{x}_{t-1}^*, \mathbf{u}_{t-1}^*, 0) + A_t(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^*) \\ &+ B_t(\mathbf{u}_{t-1} - \mathbf{u}_{t-1}^*) + V_t\mathbf{m}_t, \\ \mathbf{z}_t =& h(\mathbf{x}_t^*, 0) + H_t(\mathbf{x}_t - \mathbf{x}_t^*) + W_t\mathbf{n}_t, \end{aligned} \tag{4.16}$$

where

$$\begin{aligned} A_t &= \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}_{t-1}^*, \mathbf{u}_{t-1}^*, 0) \\ B_t &= \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_{t-1}^*, \mathbf{u}_{t-1}^*, 0) \\ V_t &= \frac{\partial f}{\partial \mathbf{m}}(\mathbf{x}_{t-1}^*, \mathbf{u}_{t-1}^*, 0) \\ H_t &= \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}_t^*, 0) \\ W_t &= \frac{\partial h}{\partial \mathbf{n}}(\mathbf{x}_t^*, 0) \end{aligned} \tag{4.17}$$

are the Jacobian matrices of f and h along the desired trajectory Π and $\bar{\mathbf{z}}_t$ is the observation. Using Equation 4.15, we can then express the system models as follows:

$$
\begin{aligned}
\bar{\mathbf{x}}_t &= A_t \bar{\mathbf{x}}_{t-1} + B_t \bar{\mathbf{u}}_{t-1} + V_t \mathbf{m}_t, && \mathbf{m}_t \sim \mathcal{N}(0, M_t), \\
\bar{\mathbf{z}}_t &= H_t \bar{\mathbf{x}}_t + W_t \mathbf{n}_t, && \mathbf{n}_t \sim \mathcal{N}(0, N_t).
\end{aligned}
\tag{4.18}
$$

All the joints are assumed to be fully actuated and independent from each other, corrupted by process noise $\mathbf{m}_{t,j} \sim \mathcal{N}(0, M_{t,j})$, where $j = 1, 2, \ldots, 7$ denotes the degree of freedom (DOF) index, and

$$
M_{t,j} = \begin{bmatrix} \sigma_{x,j}^2 & 0 \\ 0 & \sigma_{v,j}^2 \end{bmatrix}. \tag{4.19}
$$

Using the linearization from Equation 4.15, we have:

$$
\bar{\mathbf{x}}_{t,j} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \bar{\mathbf{x}}_{t-1,j} + \begin{bmatrix} \Delta T^2/2 \\ \Delta T \end{bmatrix} \bar{\mathbf{u}}_{t-1,j} + \mathbf{m}_{t,j}, \tag{4.20}
$$

where $\bar{\mathbf{x}}_{t,j}$ includes the position and velocity of the jth joint at time step t.

Instead of observing the joint configurations, we model our system as partially-observable by adopting an end-effector observation model. For manipulators mounted on mobile robots, for example, their head camera is often an important source of observations. However, unexpected movements of the mobile base caused by arm movements or external disturbances can often lead to inaccurate estimations of the relative position between the manipulator and the object to be grasped in a pick-and-place task. In these cases, the observations of the spatial relationship between obstacles and the manipulator from cameras mounted on the vehicle will inevitably be corrupted. As a result, it is of more practical significance to incorporate camera observations compared to using the fully observable joint configuration observation model.

Ideally, observations of the whole manipulator should be evaluated. However, this is nontrivial since it requires modeling the forward kinematics mapping of all the points on each link. In addition, directly modeling the observation noises for the relative spatial relationship between the entire manipulator and workspace objects is also difficult. Thus as a start, an end-effector observation model is used in this thesis to approximate the real-world camera observations. The transformation matrix between workspace objects and the end-effector can be expressed as:

$$T_{obj}^{ee} = T_{obj}^{cam} \cdot T_{cam}^{ee}, \tag{4.21}$$

where T_{obj}^{cam} is the transformation from the workspace object to the camera frame, and T_{cam}^{ee} is the transformation from the camera frame to the end-effector. Therefore, the noises for observing T_{obj}^{ee} can be transformed into observation noises for T_{cam}^{ee} through the transformation matrix T_{obj}^{cam}. Then T_{cam}^{ee} can be transformed into T_{ee}^{cam} through matrix inversion. Therefore, we can approximate the observation noises through corrupted observations of the end-effector pose from the camera.

The observations of the end-effector can be expressed in C-space through the nonlinear relationship:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{n}_t), \quad \mathbf{n}_t \sim \mathcal{N}(0, N_t), \tag{4.22}$$

where $h(\mathbf{x}_t, 0)$ is the forward kinematics, $\mathbf{n}_t$ is the observation noise, and N_t is the covariance matrix of the observation noise. The linearization of this observation model around a nominal configuration $\mathbf{x}_t^*$ can be expressed as:

$$\mathbf{z}_t - h(\mathbf{x}_t^*, 0) = J_t(\mathbf{x}_t - \mathbf{x}_t^*) + W_t\mathbf{n}_t, \tag{4.23}$$

where

$$J_t = \frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}_t^*, 0). \tag{4.24}$$

Since $h(\mathbf{x}_t, 0)$ is the forward kinematics, J_t is the end-effector Jacobian matrix at the nominal configuration $\mathbf{x}_t^*$. In this way, the linearized system observation matrix becomes the Jacobian matrix, which is usually easy to obtain during computation. Again using the linearization from Equation 4.15, the end-effector pose observation model given in Equation 4.23 can be rewritten as:

$$\bar{\mathbf{z}}_t = J_t\bar{\mathbf{x}}_t + W_t\mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}(0, N_t) \tag{4.25}$$

In p-Chekov, a discrete-time Kalman filter is used, which includes the following prediction and update phases:

$$\begin{aligned}
&Prediction: \\
&\qquad \tilde{\mathbf{x}}_t^- = A_t\tilde{\mathbf{x}}_{t-1} + B_t\bar{\mathbf{u}}_{t-1} \\
&\qquad P_t^- = A_t P_{t-1} A_t^T + V_t M_t V_t^T \\
&Measurement\ update: \\
&\qquad L_t = P_t^- H_t^T (H_t P_t^- H_t^T + W_t N_t W_t^T)^{-1} \\
&\qquad \tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_t^- + L_t(\bar{\mathbf{z}}_t - H_t\tilde{\mathbf{x}}_t^-) \\
&\qquad P_t = (I - L_t H_t) P_t^-
\end{aligned} \tag{4.26}$$

LQR controllers optimize control inputs by minimizing a quadratic cost function defined over the execution. In order to keep the robot close to the desired trajectory, we minimize deviations of robot states and control inputs in the cost function:

$$J = \mathbb{E}\left(\sum_{t=1}^{t=T}(\bar{\mathbf{x}}_t^T Q\bar{\mathbf{x}}_t + \bar{\mathbf{u}}_t^T R\bar{\mathbf{u}}_t)\right), \tag{4.27}$$

where Q and R are positive-definite weight matrices.

P-Chekov assumes the system is fully actuated and uses a finite-horizon discrete-time LQR, where the feedback matrix K_t can be computed through backward recursion:

$$\begin{aligned}
&S_T = Q, \\
&K_t = -(B_t^T S_t B_t + R)^{-1} B_t^T S_t A_t, \\
&S_{t-1} = Q + A_t^T S_t A_t + A_t^T S_t B_t K_t.
\end{aligned} \tag{4.28}$$

In LQG, since the true state $\bar{\mathbf{x}}_t$ is unknown, the state estimation $\tilde{\mathbf{x}}_t$ from the Kalman filter is used to determine the control input at each time step during the trajectory execution. This is reasonable because the separation theorem tells us that observer design and controller design can be separated into two independent processes with the guarantee of LQG optimality. Therefore, the optimal control input can be given by:

$$\bar{\mathbf{u}}_t = K_{t+1}\tilde{\mathbf{x}}_t. \tag{4.29}$$

During the execution of the whole desired trajectory, optimal state estimations based on the Kalman filter and optimal control policy computations based on LQR take turns and cycles until the execution is complete, so as to optimize the execution and track the desired trajectory.

Based on the Kalman filter and LQR, we can predict the evolution of the true state $\bar{\mathbf{x}}_t$ and the estimated state $\tilde{\mathbf{x}}_t$ at each time step t as follows [135]:

$$\begin{aligned}
\bar{\mathbf{x}}_t &= A_t\bar{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1} + V_t\mathbf{m}_t, \\
\tilde{\mathbf{x}}_t &= A_t\tilde{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1} + L_t(\bar{\mathbf{z}}_t - H_t(A_t\tilde{\mathbf{x}}_{t-1} \\
&\quad + B_tK_t\tilde{\mathbf{x}}_{t-1})) \\
&= A_t\tilde{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1} + L_t(H_t\bar{\mathbf{x}}_t + W_t\mathbf{n}_t \\
&\quad - H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1})) \\
&= A_t\tilde{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1} + L_t(H_t(A_t\bar{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1} \\
&\quad + V_t\mathbf{m}_t) + W_t\mathbf{n}_t - H_t(A_t\tilde{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1})) \\
&= A_t\tilde{\mathbf{x}}_{t-1} + B_tK_t\tilde{\mathbf{x}}_{t-1} + L_tH_tA_t\bar{\mathbf{x}}_{t-1} + L_tH_tV_t\mathbf{m}_t. \\
&\quad + L_tW_t\mathbf{n}_t - L_tH_tA_t\tilde{\mathbf{x}}_{t-1}
\end{aligned} \tag{4.30}$$

Equation 4.30 can be rewritten into matrix form:

$$\begin{aligned}
\begin{bmatrix} \bar{\mathbf{x}}_t \\ \tilde{\mathbf{x}}_t \end{bmatrix} &= \begin{bmatrix} A_t & B_tK_t \\ L_tH_tA_t & A_t + B_tK_t - L_tH_tA_t \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_{t-1} \\ \tilde{\mathbf{x}}_{t-1} \end{bmatrix} \\
&+ \begin{bmatrix} V_t & 0 \\ L_tH_tV_t & L_tW_t \end{bmatrix} \begin{bmatrix} \mathbf{m}_t \\ \mathbf{n}_t \end{bmatrix},
\end{aligned} \tag{4.31}$$

where

$$\begin{bmatrix} \mathbf{m}_t \\ \mathbf{n}_t \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} M_t & 0 \\ 0 & N_t \end{bmatrix}). \tag{4.32}$$

If we define

$$\begin{aligned} \mathbb{X}_t &\triangleq \begin{bmatrix} \bar{\mathbf{x}}_t \\ \tilde{\mathbf{x}}_t \end{bmatrix}, \\ E_t &= \begin{bmatrix} A_t & B_t K_t \\ L_t H_t A_t & A_t + B_t K_t - L_t H_t A_t \end{bmatrix}, \\ F_t &= \begin{bmatrix} V_t & 0 \\ L_t H_t V_t & L_t W_t \end{bmatrix}, \\ G_t &= \begin{bmatrix} M_t & 0 \\ 0 & N_t \end{bmatrix}, \end{aligned} \tag{4.33}$$

and initialize the variances for estimate states with 0 and the variances for true states with Σ_0, then the variance matrix C_t for $\mathbb{X}_t$ can be expressed as:

$$C_t = E_t C_{t-1} E_t^T + F_t G_t F_t^T, \quad C_0 = \begin{bmatrix} \Sigma_0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.34}$$

Therefore, the matrix of true states and estimated states $\mathbb{X}_t$ has the distribution:

$$\mathbb{X}_t \sim \mathcal{N}(\mathbf{0}, C_t). \tag{4.35}$$

Substitute into Equation 4.15 and Equation 4.16, we can get the *a priori* distributions of the true states and control inputs during the execution of the desired trajectory:

$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \mathbf{x}_t^* \\ \mathbf{u}_t^* \end{bmatrix}, \Lambda_t C_t \Lambda_t^T), \tag{4.36}$$

where

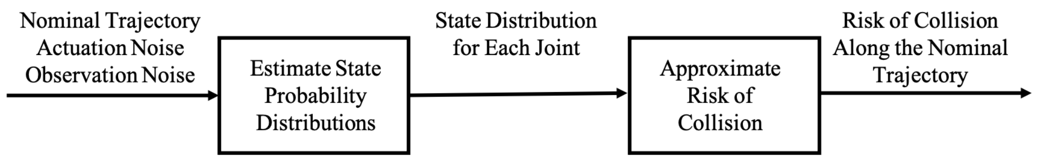

Nominal Trajectory
Actuation Noise
Observation Noise
Estimate State Probability Distributions
State Distribution for Each Joint
Approximate Risk of Collision
Risk of Collision Along the Nominal Trajectory

Figure 4-3: P-Chekov information flow from nominal trajectories to state probability distributions.

$$\Lambda_t = \begin{bmatrix} I & 0 \\ 0 & K_{t+1} \end{bmatrix}. \tag{4.37}$$

With these *a priori* distributions of robot states, we can then evaluate the probability of collision along the desired trajectory to find feasible solutions that can satisfy the given chance constraint. To summarize, the information flow from nominal trajectories to state probability distributions is illustrated in Figure 4-3.

4.4 Learning-based P-Chekov Experiment Results

The results from [29] indicate that the planning time of quadrature-based p-Chekov will severely constrain its application in real-time planning tasks that require fast-reaction. In this section, we first compare the training performance of four different classes of machine learning methods in the same two tabletop environments as shown in Figure 4-2, and then demonstrate the performance of neural network-based p-Chekov, the best performer among the four, with 500 feasible test cases in each environment.

4.4.1 Comparison between Different Regression Methods

In p-Chekov, since the nominal trajectories generated by the deterministic planner are guaranteed to be collision-free without the presence of noise, the nominal configurations inputted into the collision estimator component is more likely to lie in the collision-free configuration space. Therefore, the 60000 samples in each environment include two parts: 20000 have their mean configurations sampled from the entire configuration space (referred to as *Sample Set 1*), and 40000 have their mean configurations sampled purely from the deterministic collision-free configuration space (referred to as *Sample Set 2*). 2000 samples are held out for testing in every experiment no matter how large the training size is. Note that even though the mean configuration is not in collision, the associated collision risk is not necessarily zero if the standard deviation is nonzero. Having more samples taken from the deterministic collision-free space can better represent the practical data p-Chekov faces during online planning. All the 60000 samples from both environments are used when training neural networks, while we only use Sample Set 1 and half of Sample Set 2 to train Scikit Learn regressors because they get very slow when the data size exceeds 40000.

In order to find the best parameters for the regressors, we conduct grid search on kernel ridge regressors and random forest regressors, and use gradient descent on Gaussian process regressors. Table 4.1 shows the best parameters found for different kernels in kernel ridge regression as well as random forest regression when the training data have different sizes. Since gradient descent for Gaussian process is applied during training to maximize the log marginal likelihood, the best parameters are not shown in Table 4.1. All the experiments in this section use the best parameters we found for the corresponding data size.

The comparison between different regression methods on different datasets is shown in Table 4.2. In each dataset, 2000 randomly selected data points are used for testing and the rest are used for training. Mean squared error (MSE) and R^2 score are used to measure the test accuracy for different regressors. Given the predicted value $\hat{y}_i$ and the true value y_i for each test data point, MSE is calculated by:

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} (y_i - \hat{y}_i)^2, \tag{4.38}$$

and the R^2 score is calculated by:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n_{samples}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_{samples}} (y_i - \bar{y}_i)^2}, \tag{4.39}$$

where $\bar{y} = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} y_i$. Here the R^2 scores are computed using the test data,

Table 4.1: Best Parameters in Kernel Ridge Regression and Random Forest Regression [31]

Regressor	8000 training data	18000 training data	38000 training data
RBF Kernel Ridge Regression	$\alpha = 0.2$, $\gamma = 0.3$	$\alpha = 0.2$, $\gamma = 0.3$	$\alpha = 0.2$, $\gamma = 0.5$
Polynomial Kernel Ridge Regression	$\alpha = 0.1$, $degree = 5$	$\alpha = 0.1$, $degree = 6$	$\alpha = 0.1$, $degree = 6$
Matern Kernel Ridge Regression	$\alpha = 0.1$, $\nu = 2.29$, $length\ scale = 1.5$	$\alpha = 0.01$, $\nu = 1.14$, $length\ scale = 2.5$	$\alpha = 0.01$, $\nu = 1.44$, $length\ scale = 1.66$
Random Forest Regression	$estimators = 300$, $min\ split = 5$, $min\ leaf = 3$	$estimators = 600$, $min\ split = 5$, $min\ leaf = 3$	$estimators = 600$, $min\ split = 4$, $min\ leaf = 3$

and the MSE scores and standard deviations are computed using cross validation on the training data.

In terms of the training time performance, Gaussian process regression and Matern kernel ridge regression are the slowest. Gaussian process regression conducts gradient descent to search for best parameters during training, and also outputs distributions instead of single predicted values, which would explain its low training speed. As for Matern kernel ridge regression, the best ν parameter found by grid search is not one of the default values provided by Scikit Learn, and this would incur a considerably higher computational cost (approximately 10 times higher) since they require to evaluate the modified Bessel function [123]. In contrast, random forest regressor tends to take a very short time to train, and its training time also grows relatively slowly as the size of training data increases. In terms of prediction accuracy, Matern kernel ridge regression and random forest regression have the best performance when the training data include Sample Set 1 data, while polynomial kernel shows the worst performance. RBF kernel ridge regression shows slightly better performance compared to random forest regression when the training data are purely from Sample Set 2. When provided with 38000 training data, the MSE error of random forest regressor is relatively

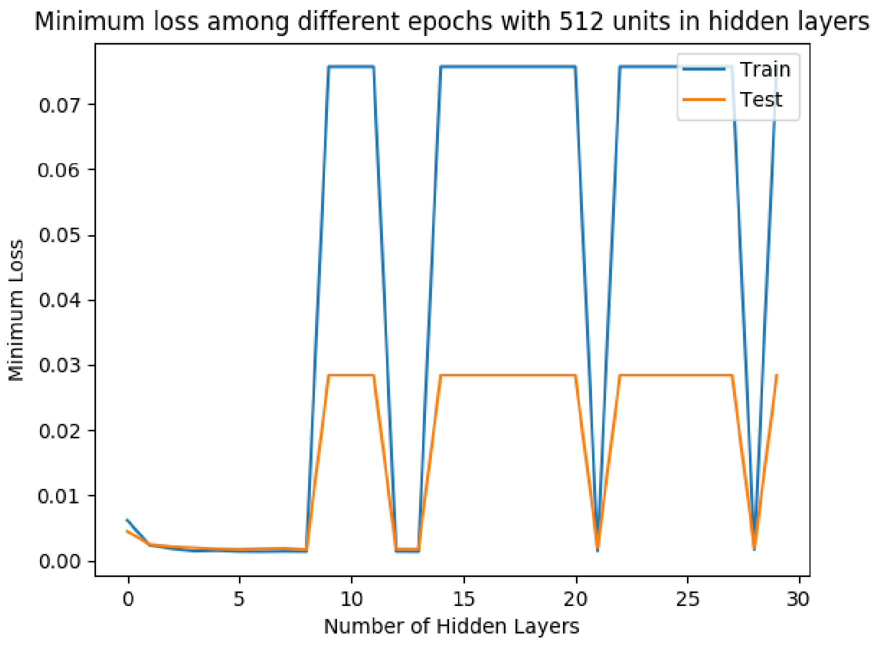
Table 4.2: Comparison of Different Regression Methods [31]

Data	Size	Regression Method		R^2 Score	MSE Error	Std of MSE	Training Time (s)
Set 1	10000	Kernel Ridge	RBF	0.792	0.0217	0.0006	5.30
			Polynomial	0.731	0.0314	0.0010	7.37
			Matern	0.801	0.0210	0.0006	40.83
		Random Forest		0.799	0.0217	0.0007	1.76
		Gaussian Process		0.791	0.0221	0.0006	140.15
	20000	Kernel Ridge	RBF	0.842	0.0180	0.0005	31.78
			Polynomial	0.783	0.0261	0.0007	58.21
			Matern	0.851	0.0172	0.0005	311.11
		Random Forest		0.854	0.0166	0.0006	4.80
		Gaussian Process		0.844	0.0181	0.0005	1108.75
Set 2	10000	Kernel Ridge	RBF	0.616	0.0077	0.0003	5.29
			Polynomial	0.551	0.0097	0.0004	8.29
			Matern	0.629	0.0075	0.0003	79.25
		Random Forest		0.593	0.0083	0.0003	5.48
		Gaussian Process		0.626	0.0077	0.0003	166.39
	20000	Kernel Ridge	RBF	0.682	0.0066	0.0002	33.27
			Polynomial	0.603	0.0085	0.0002	42.73
			Matern	0.697	0.0063	0.0002	213.28
		Random Forest		0.661	0.0071	0.0002	4.23
		Gaussian Process		0.689	0.0066	0.0002	1337.29
Both Sets	40000	Kernel Ridge	RBF	0.847	0.0117	0.0002	435.60
			Polynomial	0.773	0.0165	0.0002	261.03
			Matern	0.855	0.0110	0.0002	1619.93
		Random Forest		0.868	0.0107	0.0002	18.08
		Gaussian Process		0.849	0.0113	0.0002	8833.86

satisfactory, and a number of manually selected test points showed that the prediction is very close to the “ground truth” risk value.

If we compare the results between training on Sample Set 1 and training on Sample Set 2, we can see that Sample Set 2 tests show a smaller MSE error but a lower R^2 score. This is because the Sample Set 2 data points are all sampled from the collision-free configuration space, which would tend to have lower collision risk than the in-collision configurations. Therefore, it is reasonable that a lower absolute value leads to a lower MSE error. However, since R^2 scores measures the relative error compared to the variance of the original data, it won’t decrease as the absolute values of data points decrease. One hypothesis about why the R^2 score is lower compared to Sample Set 1 is that Sample Set 2 tends to have "ground-truth" collision risk close to 0, and there's not enough variety on the data distribution to ensure that the regressor can capture the data structure. This conclusion shows that although in practical motion planning tasks the configurations that p-Chekov needs to predict collision risk for are more likely to be in the collision-free configuration space, having data from the entire configuration space helps the regressor to learn the data distribution better and achieve higher prediction accuracy.

The neural networks used in this paper are fully connected multi-layer perceptron networks with ReLU activation for the input layer and hidden layers. Adam [66] optimizer is used, and the batch size is set to 64. Sigmoid is used as the output layer activation function since the output is collision probability. MSE is used as the loss function because this regression problem aims at minimizing the prediction error. All the 60000 data points are used in the neural network experiments: 58000 are for training and 2000 are for testing. In all the neural networks tested in this paper, the number of units in the input layer is kept as 1024, and all the output layer have 1 unit to match the sigmoid output. We compare the networks' performance with different numbers of hidden layers in Figure 4-4, where the top figure represents results from networks with 512 units in each hidden layer and the bottom figure represents the ones with 216. The vertical axis in these two figures shows the minimum training and validation losses within 50 training epochs. From Figure 4-4 we can see that when the number of hidden layers with 512 units lies between 0 and 9, the neural networks have relatively stable performance, and the minimum loss has a decreasing trend as the number of layers increases. However, when the number of hidden layer reaches 10, the neural networks start to have trouble minimizing the MSE loss. One of the potential reasons for this phenomenon is that when the neural networks get very deep, the input to the last activation layer, the sigmoid layer, might get very large. Since sigmoid function has very small gradient when the input is large, this could potentially cause the optimizer not being able to properly conduct gradient descent, which then causes high training losses and validation losses. When there



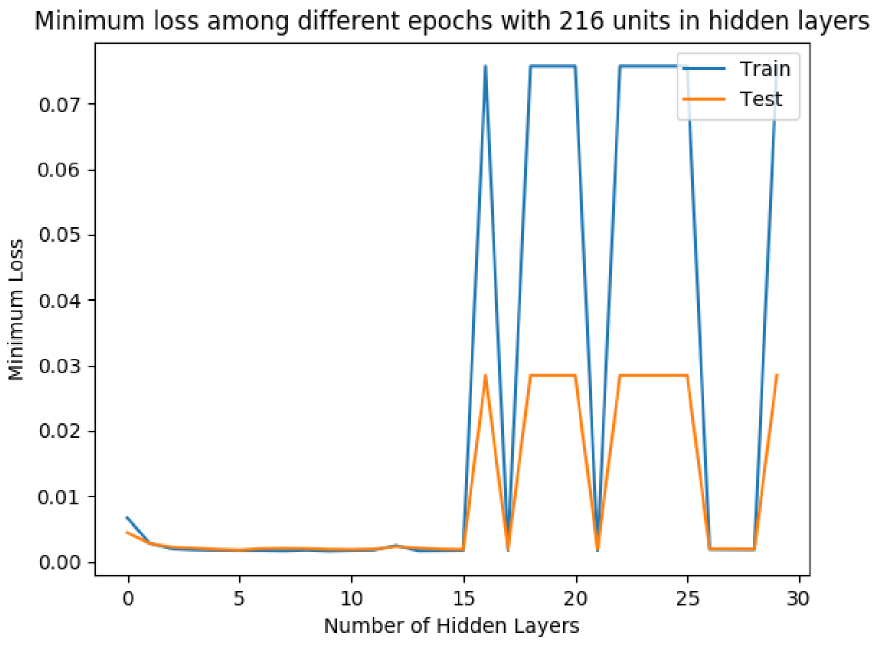


Figure 4-4: Minimum loss as a function of hidden layer numbers [31]

Table 4.3: Performance of Neural Network with 9 Hidden Layers with 512 Units Each [31]

Number of Training Epochs	Final Training Loss	Final Validation Loss	Minimum Validation Loss	Number of Epoch for Minimum Validation Loss
50	0.001371	0.001698	0.001645	49
70	0.001031	0.001519	0.001437	63
100	0.000426	0.001400	0.001302	99

are 216 units in hidden layers, the minimum loss curves show similar trends, but the networks have a wider range of hidden layer numbers where the optimization is stable since the layers are narrower. This is potentially related to the fact that when the width of each layer is smaller, it takes the networks more layers to reach saturation where the gradient of activation function approaches zero. The minimum validation loss in the bottom figure of Figure 4-4 is 0.0017, when the number of hidden layers is 6, and in the top figure the minimum reaches 0.0016, when the number of hidden layers is 9. Therefore, the optimal network structure among all tested ones has 9 hidden layers with 512 units each, an input layer with 1024 units and an output layer with one sigmoid activation unit.

Table 4.3 and Figure 4-5 show the performance of the optimal structure network, 9 hidden layers with 512 units each, when more training epochs are provided. Table 4.3 compares their performance in terms of the training and validation loss after the last epoch's training (the "Final Training Loss" and "Final Validation Loss" columns), the minimum validation loss among all epochs (the "Minimum Validation Loss" column), and the number of epoch where they reach this minimum (the "Number of Epoch for Minimum Validation Loss" column). As we can see from Table 4.3, both the training loss and the validation loss are decreasing given more training epochs, but the improvement for validation loss is much smaller compared to that of training loss. This means that as we exploit the training data more, although the performance of neural networks will gradually improve, this improvement is more about better fitting the training data structure than generalizing to the entire C-space. Therefore, we would expect very limited improvement or even decreasing validation performance

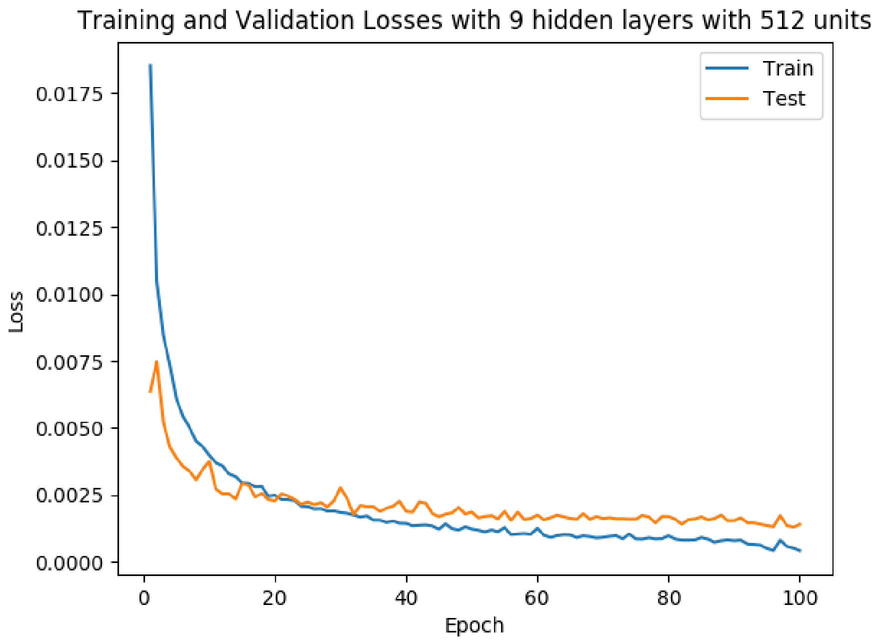


Figure 4-5: Loss and training epoch relationship for networks with 9 hidden layers with 512 units each [31]

when training more than 100 epochs. Figure 4-5 also shows that the validation loss decreases drastically in the first 30 epochs and then starts to drop slowly, whereas the training loss is still decreasing relatively fast and diverges from the validation loss in the final 30 epochs.

4.4.2 Neural Network Learning-based P-Chekov Experiment Results

Section 4.4.1 shows that the best performer among all the tested machine learning methods on this collision risk regression problem is the neural network with 9 hidden layers with 512 units in each layer, thus it is used in this section to evaluate the performance of learning-based p-Chekov. The collision risk of a solution trajectory returned by learning-based p-Chekov is evaluated with 100 noisy executions. To assess the chance constraint satisfaction performance of p-Chekov, we provide the definition of *chance constraint satisfied test cases.* If p-Chekov works perfectly, the 100 independent executions for a particular solution trajectory should all have their probability of collision equal to the chance constraint. For example, if the chance constraint allows for a 10% collision probability, the probability of collision happening during an execution should be 10%. Then the number of failures out of the 100 executions follows a binomial distribution with the number of independent experiments $n = 100$ and the probability of occurrence in each experiment $p = 0.1$. The cumulative probability distribution function of binomial distributions can be expressed as:

$$F(k; n, p) = \Pr(X \leq k) = \sum_{i}^{k} \binom{n}{i} p^i (1-p)^{n-i} \tag{4.40}$$

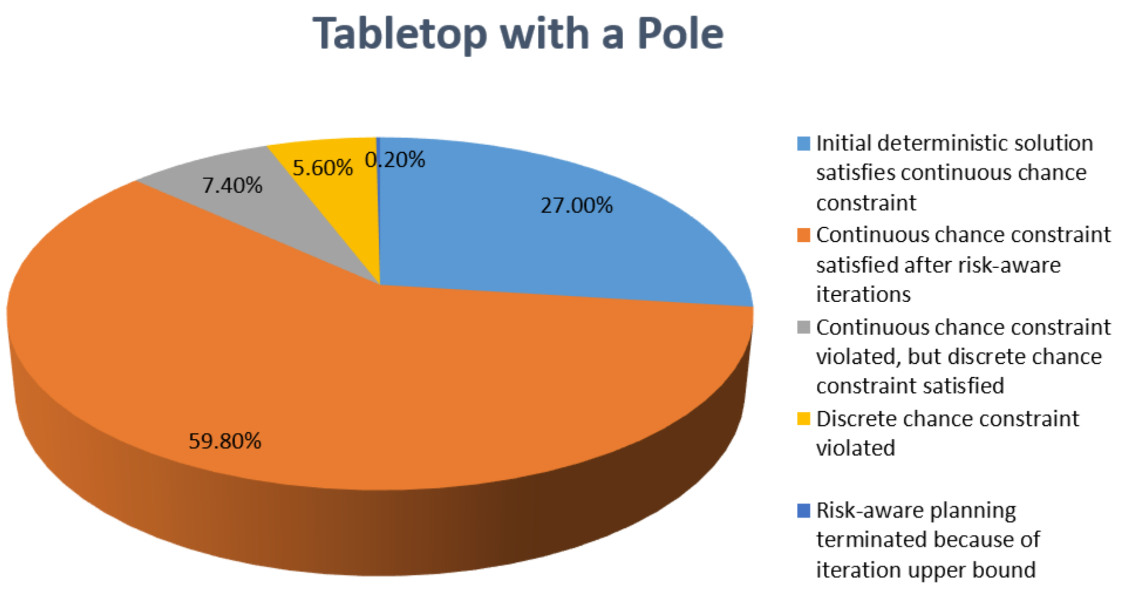
For $n = 100$ and $p = 0.1$, we can calculate from Equation 4.40 that the probability of having less than or equal to 10 failures out of 100 executions is only about 56%. Similarly, if the chance constraint is 5%, then the probability of having less than or equal to 5 failures in 100 executions is about 59%. However, to better represent the actual collision risk of solutions returned by p-Chekov, we want the classification error for *chance constraint satisfied test cases* to be small, so that we are confident to say the test case has violated the chance constraint when there are more than the corresponding number of executions end up in collision. If we define chance constraint satisfied test cases as the ones where the collision rate out of 100 executions is lower than or equal to 1.5 times of the chance constraint, Equation 4.40 shows that for $p = 0.1$ the classification accuracy is about 94%, and for $p = 0.05$ the accuracy is around 86%. Consequently, we decide to use 1.5 times of the chance constraint as the boundary between chance constraint satisfied cases and chance constraint violated cases.

We noticed that a lot of test queries where p-Chekov fails have their start or goal very close to obstacles. In these cases, feasible solutions might not exist if the collision probability of the start or goal has already exceeded the chance constraint. Therefore, we introduce a pre-processing procedure before running p-Chekov in order to filter out these potentially infeasible test queries. We estimate the collision probability of the start and goal based on the nominal trajectory computed by deterministic Chekov, and discard the test cases where the collision probability of either the start or goal exceeds 1.5 times of the chance constraint. Although it is possible that some of these cases might be feasible since our collision probability estimation approach doesn't know the ground truth, most of them are highly likely to be infeasible compared to other cases where the start and goal has low estimated collision probabilities. We pick 500 test cases that have passed this pre-processing and refer to them as "feasible cases".

Since theoretically p-Chekov only has probabilistic guarantees for waypoints instead of the entire trajectory, we distinguish between continuous-time and discrete-time chance constraint satisfaction performances. If the 100 noisy executions of a test case shows that the average continuous-time (or waypoint) collision rate is within 1.5 times of the collision chance constraint, then we say this test case satisfies the continuous-time (or discrete-time) chance constraint. Only the continuous-time satisfaction is the true criterion for success, but we use discrete-time performance to show the impact of edge collisions, i.e. the collisions in between waypoints. The test cases are divided into five groups: (1) chance constraint is satisfied by the initial deterministic Chekov solution, (2) continuous-time collision rate satisfies the chance constraint, (3) continuous-time collision rate violates the chance constraint but discrete-time collision rate satisfies it, (4) discrete-time collision rate violates the chance constraint but the p-Chekov algorithm terminated before it hits its iteration number upper bound, and (5) p-Chekov terminates because it hit the iteration limit.

Figure 4-6 demonstrates the statistics breakdown of the neural network learning-based p-Chekov experiments with noise standard deviation 0.0044 and chance constraint 10%. From Figure 4-6 we can see that the continuous-time chance constraint is satisfied in 86.8% of the feasible cases in the "tabletop with a pole" environment and 79.4% of feasible cases in the "tabletop with a container" environment. The percentage of cases that satisfy the discrete-time chance constraint but doesn't satisfy the continuous-time chance constraint is small in both environments (7.4% in the "tabletop with a pole" environment and 4.2% in the "tabletop with a container" environment), meaning that the influence of edge collisions is not significant. We randomly selected a number of test cases where learning-based p-Chekov failed due to timeout or discrete-time chance constraint violation to conduct a closer inspection and found that most of them have either start or goal pose very close to obstacles. This means a lot of these cases might be infeasible because the start or goal collision probability has already violated the chance constraint, which makes the chance-constrained query infeasible. Compared with the performance of quadrature-based p-Chekov shown in [29], we can see that the statistics breakdown for learning-based p-Chekov is similar to that of quadrature-based p-Chekov. Note that the pre-processing here is conducted with the learning-based collision estimator, which might select slightly different test cases compared to using the quadrature-based collision estimator because their estimated risk for the same start and goal pose pair might not be exactly the same. Since the learning-based collision estimation component is less conservative than the quadrature-based one, it filters out fewer difficult test cases, which could explain the relatively lower chance constraint satisfaction rate in Figure 4-6.

We present detailed performance of learning-based p-Chekov in the two tabletop environments in Table 4.4, and show quadrature-based p-Chekov's performance in Table 4.5 for comparison. Table 4.4 shows that learning-based p-Chekov significantly reduced the overall collision rate (averaged over 500 test cases with 100 noisy executions each) compared to the nominal trajectories provided by deterministic Chekov. From both tables we can see that the discrete and continuous chance constraint satisfaction performances are very close, which means edge collisions in these experiments don't have significant influence. Comparing learning-based p-Chekov's performance in the continuous chance constraint satisfied cases and violated cases, we can see that the satisfied cases take much fewer iterations than the violated cases and also have much lower average collision rate. Comparing Table 4.4 with Table 4.5, we can see that although the collision rate performance and the path length performance of the two algorithms are similar, learning-based p-Chekov's planning time is significantly shorter, especially in the more difficult "tabletop with a container" environment where it reduced the average planning time by 67%. In the cases where the continuous chance constraint is satisfied, learning-based p-Chekov is able to achieve an average planning time of about 6 s. From the "improvement from IRA" section in table 4.4 we can see



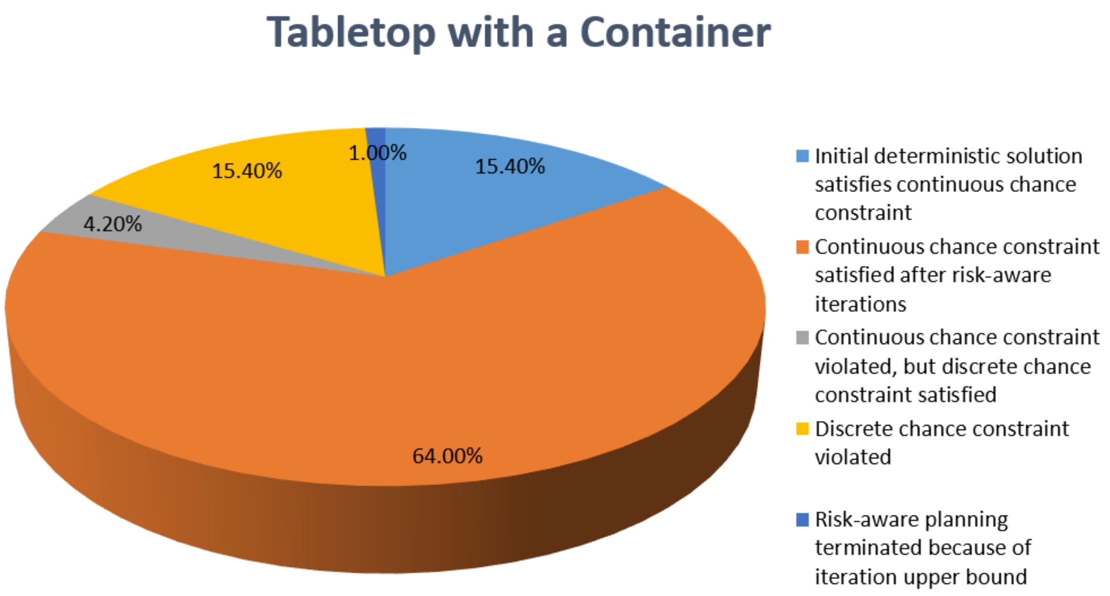


Figure 4-6: Learning-based p-Chekov statistics breakdown for feasible cases with end-effector observation, 0.0044 noise standard deviation and 10% chance constraint [31]

Table 4.4: Learning-based P-Chekov Performance with Noise Level 0.0044 and Chance Constraint 10% [31]

Environment			Tabletop with a Pole	Tabletop with a Container
Planning Time (s)	deterministic Chekov		1.12	1.22
	p-Chekov		8.65	10.15
Overall Collision Rate	deterministic Chekov		31.05%	42.54%
	p-Chekov		11.82%	18.53%
Average Path Length (rad)	deterministic Chekov		0.51	0.61
	p-Chekov		0.71	0.85
P-Chekov Performance	continuous chance constraint satisfaction rate		86.80%	79.40%
	continuous satisfied cases	average iteration number	3.49	4.41
		average planning time (s)	6.39	7.96
		average collision rate	0.11%	0.12%
		average risk reduction	0.27	0.33
	continuous violated cases	average iteration number	10.66	8.82
		average planning time (s)	22.96	18.10
		average collision rate	86.19%	85.35%
		average risk reduction	-0.28	-0.09
	discrete chance constraint satisfaction rate		94.20%	83.60%
	discrete satisfied cases	average iteration number	4.13	4.59
		average planning time (s)	8.29	8.58
		average collision rate	0.10%	0.16%
		average risk reduction	0.22	0.29
	discrete violated cases	average iteration number	9.03	9.06
		average planning time (s)	13.54	17.69
		average collision rate	65.82%	79.65%
		average risk reduction	-0.19	-0.11
Improvement from IRA (for non-zero number of IRA iteration cases)	Without IRA	continuous satisfaction rate	79.94%	72.31%
		discrete satisfaction rate	91.22%	78.23%
		average path length	0.88	0.96
	With IRA	continuous satisfaction rate	81.82%	71.24%
		discrete satisfaction rate	93.42%	77.42%
		average path length	0.86	0.97

Table 4.5: Quadrature-based P-Chekov Performance with Noise Level 0.0044 and Chance Constraint 10% [29]

Environment			Tabletop with a Pole	Tabletop with a Container
Planning Time (s)	deterministic Chekov		1.10	1.27
	p-Chekov		19.34	31.17
Overall Collision Rate	deterministic Chekov		27.51%	41.04%
	p-Chekov		11.39%	16.46%
Average Path Length (rad)	deterministic Chekov		0.51	0.60
	p-Chekov		0.68	0.84
P-Chekov Performance	continuous chance constraint satisfaction rate		87.60%	82.20%
	continuous satisfied cases	average iteration number	4.14	5.19
		average collision rate	0.08%	0.11%
		average risk reduction	0.25	0.33
	continuous violated cases	average iteration number	10.52	10.35
		average collision rate	88.50%	88.02%
		average risk reduction	-0.44	-0.13
	discrete chance constraint satisfaction rate		94.40%	86.80%
	discrete satisfied cases	average iteration number	4.82	5.49
		average collision rate	0.13%	0.10%
		average risk reduction	0.19	0.28
	discrete violated cases	average iteration number	6.94	10.32
		average collision rate	73.39%	86.59%
		average risk reduction	-0.39	-0.23
Improvement from IRA (for non-zero number of IRA iteration cases)	Without IRA	continuous satisfaction rate	82.16%	76.80%
		discrete satisfaction rate	90.35%	82.67%
		average path length	0.81	0.94
	With IRA	continuous satisfaction rate	84.21%	77.87%
		discrete satisfaction rate	91.23%	83.47%
		average path length	0.77	0.88

that IRA can slightly improve the planning phase solutions in the "tabletop with a pole" environment, but it's not able to make improvement in the more complicated "tabletop with a container" environment.

4.4.3 Discussion

To summarize, empirical experiments in robotic manipulation simulation environments in this section shows that learning-based p-Chekov inherits Chekov's advantage in reacting fast to plan requests and its ability of generating smooth motion plans. By applying supervised-learning techniques to the collision risk estimation module, learning-based p-Chekov moves significant amount of the computation to off-line to avoid the requirement on-line Monte Carlo sampling and collision risk estimation. The comparison of a variety of supervised-learning models in the p-Chekov framework led to the conclusion that neural networks are the best performers in terms of both training performance and testing performance. We demonstrate that learning-based p-Chekov with neural networks is able to provide smooth motion plans that satisfy pre-specified chance-constraints while significantly accelerating the planning speed. The comparison with sampling-based p-Chekov shows that learning-based p-Chekov can reduce planning time by 50% - 70% while maintaining similar chance constraint satisfaction rate.

To further reduce the planning time needed for finding the optimal trajectory that satisfy the chance constraint, future work should focus on reducing the number of iterations p-Chekov takes during the planning phase. This can be achieved by using more conservative constraints during the initial deterministic planning phase so that the initial nominal trajectory is suboptimal but conservative in terms of the collision risk. The trajectory can then be further optimized to fully utilize the chance constraint through IRA during the execution phase.

Chapter 5

Empowerment-based Intrinsic Motivation

This chapter presents the second contribution of this thesis: empowerment-based intrinsic motivation. Section 5.1 introduces the key concepts necessary for understanding our approach, Section 5.2 describes the empowerment-based intrinsic motivation approach, and Section 5.3 presents the empirical evaluation results. Section 5.4 demonstrates a potential application of our approach: learning a diverse set of skills through intrinsic motivation.

5.1 Background

The key concept in the approach proposed in this chapter is empowerment, which is originally defined as the maximum mutual information between a sequence of actions and the final states conditioned on the initial states. In this thesis, we found that various simplifications and approximations are necessary for adapting this concept to robotic manipulation tasks. In this section, we present the background knowledge necessary for understanding the definition of empowerment and our approaches to simplifying it in the robotic manipulation framework. This section first introduces the concept of mutual information and methods for maximizing the mutual information (Section 5.1.1 - Section 5.1.3) since the key concept in our proposed approach, empowerment, is a form of conditional mutual information. We then describe empowerment (Section 5.1.4) and intrinsic curiosity module (Section 5.1.5), two key methods that are utilized in the empowerment-based intrinsic motivation approach. Our approach makes simplifications to the empowerment definition to make it practical for robotic manipulation tasks with continuous state space, and also combine it with intrinsic curiosity module to get a warm start during training.

5.1.1 Mutual Information

Mutual information (MI) is a fundamental quantity for measuring the mutual dependence between random variables. It quantifies the amount of information obtained about one random variable through observing the other. For a pair of discrete random variables X and Y, the MI is defined as:

$$\mathcal{I}(X;Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}, \tag{5.1}$$

where $P_X(x)$ and $P_Y(y)$ are the marginal probability mass functions for X and Y respectively, and $P_{XY}(x,y)$ is the joint probability mass function of X and Y. If X and Y are continuous random variables, then the MI between them can be written as:

$$\begin{aligned} \mathcal{I}(X;Y) &= \iint p_{XY}(x,y) \log \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}\, dx\, dy \\ &= \mathbb{E}_{XY}\Big[\log \frac{p_{XY}}{p_X p_Y}\Big], \end{aligned} \tag{5.2}$$

where $p_X(x)$ and $p_Y(y)$ are the marginal probability density functions for X and Y respectively, and $p_{XY}(x,y)$ is the joint probability density function of X and Y.

MI is also often expressed in terms of Shannon entropy [99]:

$$\begin{aligned}\mathcal{I}(X;Y) &= \mathcal{H}(X) - \mathcal{H}(X|Y)\\ &= \mathcal{H}(Y) - \mathcal{H}(Y|X)\\ &= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y)\\ &= \mathcal{H}(X,Y) - \mathcal{H}(X|Y) - \mathcal{H}(Y|X),\end{aligned} \tag{5.3}$$

where $\mathcal{H}(X)$ and $\mathcal{H}(Y)$ are the marginal entropies for random variables X and Y respectively, $\mathcal{H}(X|Y)$ and $\mathcal{H}(Y|X)$ are conditional entropies, and $\mathcal{H}(X,Y)$ is the joint entropy of X and Y. Qualitatively, entropy represents the uncertainty inherent in the random variable's possible outcomes. The conditional entropy $\mathcal{H}(X|Y)$ measures the average uncertainty about X after observing a second random variable Y. Quantitatively, it is defined as:

$$\mathcal{H}(X|Y) = \mathbb{E}_Y[-\mathbb{E}_{X|Y} \log p_{X|Y}], \tag{5.4}$$

where $p_{X|Y}(x|y) \equiv p_{XY}(x,y)/p_Y(y)$ is the conditional probability density function of X given Y. Therefore, Equation 5.3 shows that MI represents the reduction in uncertainty about one random variable after observing the other [130]. Figure 5-1 illustrates intuitively the relationship between mutual information and entropies.

Mutual information is also intimately related to the Kullback-Leibler divergence (KL-divergence) which measures the distance between two distributions. KL-divergence is defined as

$$D_{KL}(P(z)||Q(z)) = \sum_z P(z) \log\left(\frac{P(z)}{Q(z)}\right) \tag{5.5}$$

for discrete probability distributions $P(z)$ and $Q(z)$, and

$$D_{KL}(p(z)||q(z)) = \int p(z) \log\left(\frac{p(z)}{q(z)}\right) dz \tag{5.6}$$

for continuous probability distributions $p(z)$ and $q(z)$. For jointly discrete or jointly continuous pairs of random variables (X,Y), MI is the KL-divergence between the joint distribution and the product of the marginal distributions:

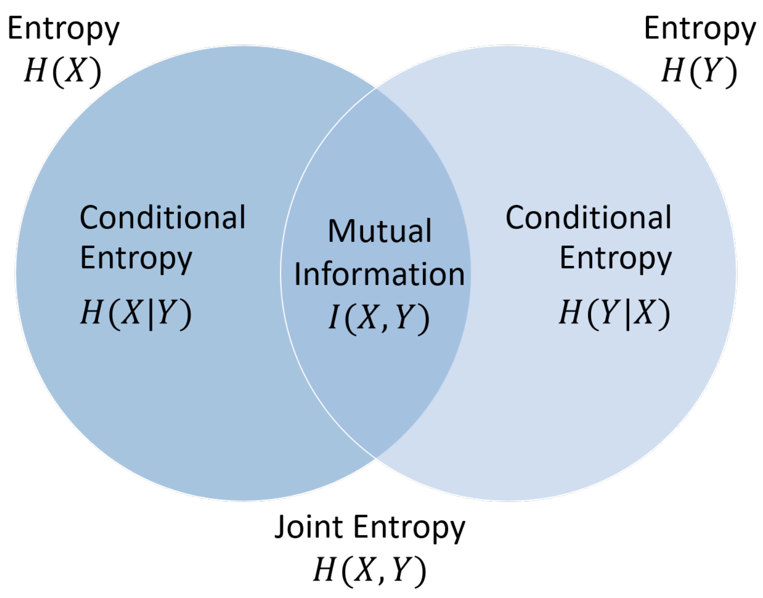


Figure 5-1: Mutual Information Illustration

$$\mathcal{I}(X;Y) = D_{KL}(p_{XY}||p_X p_Y). \tag{5.7}$$

Intuitively, Equation 5.7 means that MI can represent how close the true joint distribution is to the independent joint distribution, which allows MI to capture all dependencies between the two random variables.

5.1.2 Conditional Mutual Information

Conditional MI measures the mutual dependency between two random variables conditioned on another random variable. The conditional MI between discrete variables X and Y conditioned on Z is defined as:

$$
\begin{aligned}
\mathcal{I}(X;Y|Z) &= \sum_{z} P_Z(z) \sum_{x,y} P_{X,Y|Z}(x,y|z) \log \frac{P_{X,Y|Z}(x,y|z)}{P_{X|Z}(x|z)P_{Y|Z}(y|z)} \\
&= \sum_{x,y,z} P_{X,Y,Z}(x,y,z) \log \frac{P_Z(z)P_{X,Y,Z}(x,y,z)}{P_{X,Z}(x,z)P_{Y,Z}(y,z)} \\
&= \sum_{x,y,z} P_{X,Y,Z}(x,y,z) \log \frac{P_{X|Y,Z}(x|y,z)}{P_{X|Z}(x|z)} \\
&= \sum_{x,y,z} P_{X,Y,Z}(x,y,z) \log \frac{P_{Y|X,Z}(y|x,z)}{P_{Y|Z}(y|z)},
\end{aligned} \tag{5.8}
$$

where $P_{X,Y,Z}(x,y,z)$ is the joint probability mass function for random variables X, Y and Z, $P_{X,Y|Z}(x,y|z)$ is the joint probability mass function for X and Y conditioned on Z, $P_{X|Y,Z}(x|y,z)$ is the conditional probability mass function for X conditioned Y and Z, $P_{Y|X,Z}(y|x,z)$ is the conditional probability mass function for Y conditioned X and Z, $P_{X|Z}(x|z)$ and $P_{Y|Z}(y|z)$ are the conditional probability mass functions for X and Y conditioned on Z respectively. If X, Y and Z are continuous random variables, then the conditioned MI can be written as:

$$
\begin{aligned}
\mathcal{I}(X;Y|Z) &= \int_z \Bigg(\int_y \int_x \log \Big(\frac{p_{X,Y|Z}(x,y|z)}{p_{X|Z}(x|z)p_{Y|Z}(y|z)} \Big) p_{X,Y|Z}(x,y|z)\, dx\, dy \Bigg) p_Z(z) dz \\
&= \iiint \log \Big(\frac{p_Z(z)p_{X,Y,Z}(x,y,z)}{p_{X,Z}(x,z)p_{Y,Z}(y,z)} \Big) p_{X,Y,Z}(x,y,z)\, dx\, dy\, dz \\
&= \iiint \log \Big(\frac{p_{X|Y,Z}(x|y,z)}{p_{X|Z}(x|z)} \Big) p_{X,Y,Z}(x,y,z)\, dx\, dy\, dz \\
&= \iiint \log \Big(\frac{p_{Y|X,Z}(y|x,z)}{p_{Y|Z}(y|z)} \Big) p_{X,Y,Z}(x,y,z)\, dx\, dy\, dz,
\end{aligned} \tag{5.9}
$$

where $p_{X,Y,Z}(x,y,z)$ is the joint probability density function for X, Y and Z, and $p_{X,Y|Z}(x,y|z)$, $p_{X|Y,Z}(x|y,z)$, $p_{Y|X,Z}(y|x,z)$, $p_{X|Z}(x|z)$ and $p_{Y|Z}(y|z)$ are conditional probability density functions.

Conditional MI can also be written in terms of entropies:

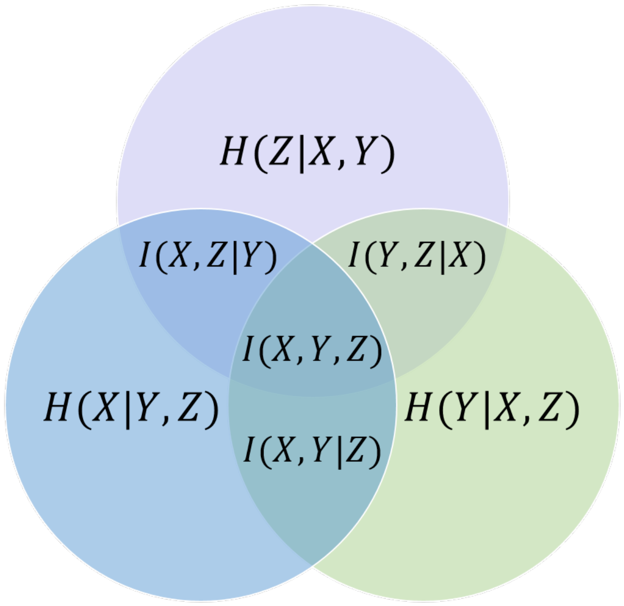


Figure 5-2: Conditional Mutual Information Illustration

$$\begin{aligned}\mathcal{I}(X;Y|Z) &= \mathcal{H}(X|Z) + \mathcal{H}(Y|Z) - \mathcal{H}(X,Y|Z) \\ &= \mathcal{H}(X|Z) - \mathcal{H}(X|Y,Z) \\ &= \mathcal{H}(Y|Z) - \mathcal{H}(Y|X,Z).\end{aligned} \tag{5.10}$$

Figure 5-2 provides an intuitive illustration of the relationship between mutual information, conditional mutual information and entropies. The approaches for estimating general MI described in Section 5.1.1 can also be applied to estimate conditional MI, but it is much more difficult to get an accurate approximation for conditional MIs, especially for continuous random variables. This is because when Z is a continuous random variable, estimating $\mathcal{I}(X;Y|Z)$ for all Z is approximately equivalent to estimating an infinite number of unconditional MIs. Closed form solution for conditional MI is only available for a very small number of distributions with linear dependencies whose closed form entropies are known. Neural function approximators have recently become powerful tools for numerically estimating conditional MIs for continuous random variables [83, 12, 64].

5.1.3 Mutual Information Computation

In general, the computation of MI is intractable. Exact computation of MI is only tractable for discrete random variables and a limited family of problems where the probability distributions are known [12]. Traditional algorithms for mutual information maximization, e.g. the Blahut-Arimoto algorithm [27], don't scale well to realistic problems with continuous state space because they typically rely on enumeration. Therefore, instead of computing the exact value of MI, a common approach for mutual information maximization in the deep reinforcement learning community is to bound MI from below and maximize the lower bound [110].

Many lower bounds of MI has been proposed that are easier to compute than the exact value of MI. One of the most commonly used lower bound is the variational lower bound of MI derived from the non-negativity of KL-divergence. Suppose X and Y are the variables that we want to estimate mutual information on, and the true conditional distribution of X conditioned on Y is $p(x|y)$, then the variational lower bound can be derived as follows:

$$\begin{aligned}
\mathcal{I}(X;Y) &= \mathbb{E}_{XY}\Big[\log\frac{p(x|y)\cdot q(x|y)}{p(x)\cdot q(x|y)}\Big]\\
&= \mathbb{E}_{XY}\Big[\log\frac{q(x|y)}{p(x)}\Big] + \mathbb{E}_{XY}\Big[\log\frac{p(x|y)}{q(x|y)}\Big]\\
&= \mathbb{E}_{XY}\Big[\log\frac{q(x|y)}{p(x)}\Big] + \mathbb{E}_{Y}\Big[D_{KL}(p(x|y)||q(x|y))\Big]\\
&\geq \mathbb{E}_{XY}\Big[\log\frac{q(x|y)}{p(x)}\Big],
\end{aligned} \tag{5.11}$$

where $q(x|y)$ is a variational approximation of $p(x|y)$, and the bound is tight when $q(x|y) = p(x|y)$. In most common reinforcement learning (RL) scenarios, the distributions we are interested in computing MI on are unknown, and we can only approximate the exact distributions through monte carlo sampling. If we use the approximated conditional distribution through sampling as $q(x|y)$, then Equation 5.11 proves that using $q(x|y)$ instead of $p(x|y)$ to estimate the MI provides a lower bound for the true MI, and the bound is tight when the approximated conditional distribution converges to the true conditional distribution, i.e. $q(x|y) = p(x|y)$. The derivation of Equation 5.11 is based on the fact that the KL-divergence is always non-negative [28], i.e. $D_{KL}(p(x|y)||q(x|y) \geq 0$.

For conditional MI $\mathcal{I}(X;Y|Z)$, the variational lower bound can be derived as:

$$\begin{aligned}
\mathcal{I}(X;Y|Z) &= \mathbb{E}_{XY|Z}\Big[\log \frac{p(x|y,z)\cdot q(x|y,z)}{p(x|z)\cdot q(x|y,z)}\Big] \\
&= \mathbb{E}_{XY|Z}\Big[\log \frac{q(x|y,z)}{p(x|z)}\Big] + \mathbb{E}_{XY|Z}\Big[\log \frac{p(x|y,z)}{q(x|y,z)}\Big] \\
&= \mathbb{E}_{XY|Z}\Big[\log \frac{q(x|y,z)}{p(x|z)}\Big] + \mathbb{E}_{Y|Z}\Big[D_{KL}(p(x|y,z)||q(x|y,z))\Big] \\
&\geq \mathbb{E}_{XY|Z}\Big[\log \frac{q(x|y,z)}{p(x|z)}\Big].
\end{aligned} \tag{5.12}$$

where $q(x|y,z)$ is a variational approximation of $p(x|y,z)$, and the bound is tight when $q(x|y,z) = p(x|y,z)$.

Other variational lower bounds of MI have also been derived based on a broader class of distance measures called f-divergence [76], of which the KL-divergence is a special case. The f-divergence between two distributions P and Q is defined as:

$$\begin{aligned}
D_f(P(z)||Q(z)) &= \int f\left(\frac{dP}{dQ}\right) dQ \\
&= \int_z f\left(\frac{p(z)}{q(z)}\right) q(z)\, dz,
\end{aligned} \tag{5.13}$$

where the generator function $f : \mathbb{R}_+ \to \mathbb{R}$ is a convex, lower-semicontinuous function satisfying $f(1) = 0$. The variational lower bound of f-divergences has been derived in [87] and [88]:

$$D_f(P(z)||Q(z)) \geq \sup_{F\in\mathcal{F}}(\mathbb{E}_{z\sim P}[F(z)] - \mathbb{E}_{z\sim Q}[f^*(F(z))]), \tag{5.14}$$

where $\mathcal{F}$ is an arbitrary class of functions $F : \mathcal{Z} \to \mathbb{R}$, and f^* is the convex conjugate of f. Equation 5.14 yields a lower bound because the class of functions $\mathcal{F}$ may only contain a subset of all possible functions, and under mild conditions on f [87], the bound is tight when:

$$F(x) = f'\left(\frac{p(z)}{q(z)}\right). \tag{5.15}$$

KL-divergence is a special case of f-divergence when the generator function $f(u) = u \log u$ [88]. Therefore, a lower bound of KL-divergence can be derived as [12]:

$$D_{KL}(P||Q) \geq \sup_{T \in \mathcal{T}} \mathbb{E}_P[F] - \mathbb{E}_Q[e^{F-1}]. \tag{5.16}$$

From Equation 5.7 we know that MI can be represented as the KL-divergence between the joint distribution and the product of the marginal distribution, hence a lower bound of MI can be derived from Equation 5.16:

$$\mathcal{I}(X;Y) \geq \sup_{F \in \mathcal{T}} \mathbb{E}_{p_{XY}}[F] - \mathbb{E}_{p_X p_Y}[e^{F-1}]. \tag{5.17}$$

where $\mathcal{F}$ is an arbitrary class of functions $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$. For conditional MI $\mathcal{I}(X;Y|Z)$, the KL-divergence lower bound can be written as:

$$\mathcal{I}_{KL}(X;Y|Z) \geq \sup_{F \in \mathcal{F}} \mathbb{E}_{p_{XY|Z}}[F] - \mathbb{E}_{p_{X|Z} p_{Y|Z}}[e^{F-1}], \tag{5.18}$$

where $\mathcal{F}$ is an arbitrary class of functions $F : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \rightarrow \mathbb{R}$.

Jensen-Shannon (JS) divergence is another special case of f-divergence. It can be expressed in terms of KL-divergence:

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M), \tag{5.19}$$

where $M = 1/2(P+Q)$. JS-divergence represents the mutual information between a random variable A associated to a mixture distribution between P and Q and a binary indicator variable B that is used to switch between P and Q. In particular, if we use P to represent the joint distribution p_{XY} and use Q to represent the product of the marginal distributions $p_X p_Y$, then:

$$p(A|B) = \begin{cases} p(x,y) & \text{if } \ B = 0, \\ p(x)p(y) & \text{if } \ B = 1. \end{cases} \tag{5.20}$$

That is, the random variable A is chosen according to the probability measure $M = (P+Q)/2$, and its distribution is the mixture distribution. Then the relationship between JS-divergence and mutual information can be derived as follows [120, 50]:

$$\begin{aligned} \mathcal{I}(A;B) &= \mathcal{H}(A) - \mathcal{H}(A|B) \\ &= -\sum M \log M + \frac{1}{2}[\sum P \log P + \sum Q \log Q] \\ &= -\sum \frac{P}{2} \log M - \sum \frac{Q}{2} \log M + \frac{1}{2}[\sum P \log P + \sum Q \log Q] \\ &= \frac{1}{2}\sum P(\log P - \log M) + \frac{1}{2}\sum Q(\log Q - \log M) \\ &= D_{JS}(P||Q). \end{aligned} \tag{5.21}$$

Therefore, if we define the Jensen-Shannon mutual information (JSMI) between two random variables X and Y as the JS-divergence between their joint distribution and the product of their marginal distributions, i.e. $\mathcal{I}_{JS}(X;Y) \equiv D_{JS}(p_{XY}||p_X p_Y)$, then Equation 5.21 shows that:

$$\mathcal{I}_{JS}(X;Y) = \mathcal{I}(A;B). \tag{5.22}$$

The advantage of using JS-divergence is that it is not only symmetric but also bounded from both below and above [64]. Although different from the commonly accepted definition of MI, JSMI is closely correlated to MI and can also represent the mutual dependence between random variables.

It is shown in [88] that JS-divergence is a special case of f-divergence when the generator function $f(u) = -(u+1)\log((1+u)/2) + u \log u$. We can then derive a lower bound of the JSMI [56, 64] according to the property of f-divergence shown in Equation 5.14:

$$
\begin{aligned}
\mathcal{I}_{JS}(X;Y) &= D_{JS}(p_{XY}||p_X p_Y) \\
&\geq \sup_{F\in\mathcal{F}} \mathbb{E}_{p_{XY}}[\log 2 - \log(1+e^{-F})] - \mathbb{E}_{p_X p_Y}[D^*_{JS}(\log 2 - \log(1+e^{-F}))] \\
&= \sup_{F\in\mathcal{T}} \mathbb{E}_{p_{XY}}[-\mathrm{sp}(-F)] - \mathbb{E}_{p_X p_Y}[\mathrm{sp}(F)] + \log 4,
\end{aligned}
\tag{5.23}
$$

where $D^*_{JS}(u) = -\log(2-\exp(u))$ is the Fenchel conjugate of JS-divergence, and $\mathrm{sp}(u) = \log(1+\exp(u))$ is the soft plus function. The Jensen-Shannon lower bound for conditional MI can be written as:

$$
\begin{aligned}
\mathcal{I}_{JS}(X;Y|Z) &= D_{JS}(p_{XY|Z}||p_{X|Z} p_{Y|Z}) \\
&\geq \sup_{F\in\mathcal{F}} \mathbb{E}_{p_{XY|Z}}[\log 2 - \log(1+e^{-F})] - \mathbb{E}_{p_{X|Z} p_{Y|Z}}[D^*_{JS}(\log 2 - \log(1+e^{-F}))] \\
&= \sup_{F\in\mathcal{F}} \mathbb{E}_{p_{XY|Z}}[-\mathrm{sp}(-F)] - \mathbb{E}_{p_{X|Z} p_{Y|Z}}[\mathrm{sp}(F)] + \log 4,
\end{aligned}
\tag{5.24}
$$

where $\mathcal{F}$ is an arbitrary class of functions $F : \mathcal{X}\times\mathcal{Y}\times\mathcal{Z} \to \mathbb{R}$. Following Equation 5.15 we can then derive that the bound for conditional JSMI is tight when:

$$
F(x) = f'\left(\frac{p(x,y|z)}{p(x|z)p(y|z)}\right). \tag{5.25}
$$

Note that Equation 5.23 is not a lower bound for the MI we defined in Equation 5.2, but since the two MIs are closely related, it is also often used to estimate the MI defined in Equation 5.2. In this thesis, we refer to the variational lower bound in Equation 5.11 as VLB, the lower bound based on KL-divergence in Equation 5.17 as KLD, and the lower bound for JS-divergence based mutual information in Equation 5.23 as JSD.

5.1.4 Empowerment

Empowerment is an information-theoretic quantity that measures the value of the information an agent obtains in the action-observation sequences it experiences during the reinforcement learning process [83]. Empowerment $\mathcal{E}$ is defined as the maximum mutual information between a sequence of K actions $\mathbf{a}$ and the final state $\mathbf{s}'$, conditioned on a starting state $\mathbf{s}$:

$$\mathcal{E}(\mathbf{s}) = \max_{\pi} \mathcal{I}^{\pi}(\mathbf{a}, \mathbf{s}'|\mathbf{s}) = \max_{\pi} \mathbb{E}_{p(s'|a,s)\pi(a|s)} \left[\log \left(\frac{p(\mathbf{a}, \mathbf{s}'|\mathbf{s})}{\pi(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s})} \right) \right], \tag{5.26}$$

where $\mathbf{a} = \{a_1, \ldots, a_K\}$ is a sequence of K primitive actions leading to a final state $\mathbf{s}'$, $\pi(\mathbf{a}|\mathbf{s})$ is exploration policy over the K-step action sequences, $p(\mathbf{s}'|\mathbf{a}, \mathbf{s})$ is the K-step transition probability of the environment, $p(\mathbf{a}, \mathbf{s}'|\mathbf{s})$ is the joint distribution of actions sequences and the final state conditioned on the initial state $\mathbf{s}$, and $p(\mathbf{s}'|\mathbf{s})$ is the marginalized probability over the action sequence.

Empowerment indicates the amount of information contained in the action sequences $\mathbf{a}$ about the future state $\mathbf{s}'$. It provides a well-grounded, task-agnostic measure of intrinsic motivation for reinforcement learning agents. An empowerment-based agent generates an open-loop sequence of K actions into the future according to $\pi(\mathbf{a}|\mathbf{s})$ in order to compute the empowerment of its current state. When optimized using Equation 5.26, $\pi(\mathbf{a}|\mathbf{s})$ becomes an efficient exploration policy that allows for uniform exploration of the state space reachable at horizon K. However, as stated in Mohamed and Rezende [83], this open-loop exploration policy should only be used for internal planning while computing the empowerment values along the map. When the agent acts in the world, it should follow a closed-loop policy obtained by a planning algorithm using the calculated empowerment values.

Mohamed and Rezende [83] proposed to separate the state empowerment computation and the intrinsic motivated learning into two separate phases. Although this approach might work well in the grid world environments they studied, it would become infeasible in high-dimensional, continuous state space to compute the empowerment values for all states beforehand. Instead, approximations like one-step empowerment or computing empowerment of a local neighborhood during learning are needed in most practical robotics manipulation applications in order to utilize this approach. In this thesis, we make various simplifications and approximations including the one-step action simplification to make this approach work on practical robotic manipulation tasks.

5.1.5 Intrinsic Curiosity Module

Intrinsic Curiosity Module (ICM) [100] is one of the state-of-the-art novelty-driven intrinsic exploration approaches that aims at learning new skills by performing actions whose consequences are hard to predict. The intuition behind ICM is that curiosity is what drives babies to explore the environment and discover novel skills, hence it might also be able to facilitate a robotic agent in its learning process as well. In order to pursue novelty without being distracted by noises in the visual world, ICM learns a low-dimensional feature representation from the original image input using its two components: an inverse model that finds the action given the feature representation of the current state and the next state, and a forward model that predicts the feature representation of next state based on the current state and the action:

$$\begin{aligned} &\text{Inverse Model: } \hat{a}_t = g(\phi(s_t), \phi(s_{t+1})); \\ &\text{Forward Model: } \hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t). \end{aligned} \tag{5.27}$$

ICM trains an inverse model g to learn a feature encoding ϕ that captures the parts of the state space related to the consequences of the agent's actions, so that the agent will focus on the relevant part of the environment and not get distracted by other details in the camera observations. It also learns the forward model f to predict the feature encoding in the next time step, and uses the prediction error of the forward model as the intrinsic reward in order to facilitate the agent to explore the part of the state space where it can't predict the consequences of its own actions very well. ICM has shown its effectiveness in video games including VizDoom and Super Mario Bros.

In this thesis, we argue that curiosity and empowerment are two opposite concepts that both can be utilized as intrinsic motivations for robot learning. Curiosity seeks novelty in states and prefers actions that generates unpredictable consequences, whereas empowerment maximizes the amount of information obtained through a sequence of actions and favors actions that lead to predictable outcomes. We view a curiosity-driven agent as an exploring agent, and an empowerment-driven agent as a controlling agent. We argue that exploration should be prioritized at the beginning of the learning process and controllability should be prioritized during the late stage of learning in order to learn specialized skills. In this thesis, we propose a novel form of intrinsic motivation that achieves superior performance compared to other state-of-the-art intrinsic exploration approaches through integrating and balancing empowerment and curiosity.

5.2 Approach: Empowerment-based Intrinsic Motivation

We hypothesize that empowerment would be a good candidate for augmenting the sparse extrinsic rewards in manipulation tasks because it indicates the amount of information contained in the action sequence $\mathbf{a}$ about the future state $\mathbf{s}'$. Through maximizing empowerment, we are effectively encouraging the agent to influence the environment in a predictable way, which is the desired behavior in most manipulation tasks. The intuition behind the concept of empowerment can be illustrated in a robotic manipulation environment, as shown in Figure 5-3. In the column (a) of Figure 5-3, the robotic manipulator is moving around without touching the target object, which means it is not influencing the state of the environment, hence the empowerment should be zero in this scenario. In column (b), although the manipulator is interacting with the target object, the interactions are random and the consequences are difficult to predict, which means the empowerment value in this scenario will be low. In comparison, the manipulator is conducting desirable behaviors such as lifting and pushing in column (c). The influence of the robotic agent's action on the states of the environment is more predictable in this scenario, hence the empowerment value

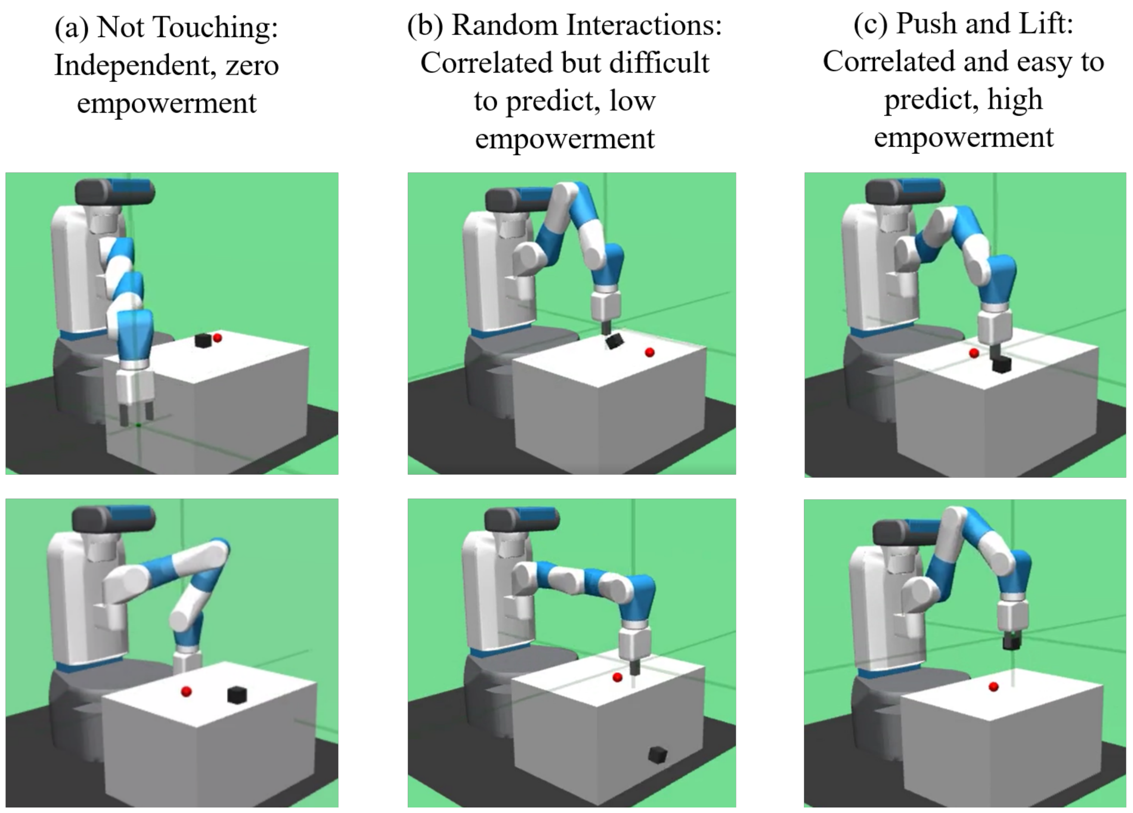

(a) Not Touching: Independent, zero empowerment
(b) Random Interactions: Correlated but difficult to predict, low empowerment
(c) Push and Lift: Correlated and easy to predict, high empowerment

Figure 5-3: Empowerment Intuition

should also also higher.

However, as a form of conditional MI for continuous variables, the computation of empowerment is especially challenging. This is because for conditional MI $\mathcal{I}(X;Y|Z)$ with continuous Z, estimating $\mathcal{I}(X;Y|Z)$ for all Z is approximately equivalent to estimating an infinite number of unconditional MIs. In this section, we discuss the approaches we take to make empowerment a feasible form of intrinsic motivation in practical robotic manipulation tasks [34].

5.2.1 Approximations to Simplify Empowerment Calculation

[83] suggest that the empowerment at each state in the state space can be calculated using an exploration policy $\pi(\mathbf{a}|\mathbf{s})$ that generates an open-loop sequence of K actions into the future (Equation 5.26), so that a closed-loop policy can be obtained by a planning algorithm using the calculated empowerment values. Although [83]

demonstrated the effectiveness of this approach in grid world environments, it is infeasible to precompute the empowerment values for all states in a high-dimensional, continuous state space. Therefore, we make a few approximations in order to make empowerment-based intrinsic motivation a practical approach. First, we use only one action step instead of an action sequence to estimate empowerment. This means we will sacrifice the globally optimality while computing empowerment, but it reduces the dimensionality and simplifies the computation significantly. Second, instead of constructing a separate exploration policy π to first compute empowerment and then plan a closed-loop policy according to empowerment, we directly optimize the behavior policy ω using empowerment as an intrinsic reward in an RL algorithm. These two approximations mean that the agent will only be looking at the one-step reachable neighborhood of its current state to find the policy that leads to high mutual information. Despite sacrificing global optimality, this approach prioritizes the policy that controls the environment in a principled way so that more extrinsic task rewards can be obtained compared to using random exploration, which help resolve the fundamental issue in sparse reward tasks.

In addition to the above two approximations, it is also important to note that in robotic manipulation tasks, we are typically not interested in the mutual dependence between robot action and robot states, and we wish to avoid the robot trivially maximizing empowerment through motion of its own body. Therefore, we assume that the state space can be divided into intrinsic states $\mathbf{s}^{in}$ (robot states) and extrinsic states $\mathbf{s}^{ex}$ (environment states), and only extrinsic states are used as $\mathbf{s}'$ when calculating empowerment. Namely, the empowerment used in this chapter is defined as:

$$\mathcal{E}(\mathbf{s}_t) \approx \mathcal{I}^{\omega}(\mathbf{a}_t, \mathbf{s}_{t+1}^{ex}|\mathbf{s}_t) = \mathcal{H}^{\omega}(\mathbf{a}_t|\mathbf{s}_t) - \mathcal{H}^{\omega}(\mathbf{a}_t|\mathbf{s}_{t+1}^{ex}, \mathbf{s}_t), \tag{5.28}$$

where ω is the behavior policy, and the relationship to Shannon Entropy is derived from Equation 5.3.

5.2.2 Maximizing Empowerment using Mutual Information Lower Bounds

Neural function approximators have become powerful tools for numerically estimating conditional MIs for continuous random variables [83, 12, 64]. However, in most RL scenarios, since exact distributions are typically unavailable and numerical estimation through sampling is required, computation of high-dimensional conditional MI remain challenging. As mentioned in Section 5.1.3, a common practice is to maximize a lower bound of MI instead of its exact value. In order to evaluate the performance of different MI lower bounds and select the appropriate bound for the experiments in this thesis, we first conduct a unit test on distributions with known conditional MI. We test the performance of the three MI lower bounds introduced in Section 5.1.3: the variational lower bound (VLB, shown in Equation 5.11), the lower bound based on KL-divergence (KLD, shown in Equation 5.17), and the lower bound for JS-divergence based mutual information (JSD, shown in Equation 5.23).

Unit Tests on Conditional Mutual Information Estimation

Consider random variables X, Y and Z, whose distributions can be described as follows:

$$\begin{aligned} Z &\sim \mathcal{N}(0, \sigma_z^2), \\ X &= Z + e, \\ Y &= \begin{cases} Z + X \cdot Z + f, & \text{if } Z > 0 \\ f, & \text{if } Z \leq 0 \end{cases} \\ e &\sim \mathcal{N}(0, 1), \\ f &\sim \mathcal{N}(0, n^2). \end{aligned} \tag{5.29}$$

In words, random variable Z is zero-mean, Gaussian distributed, random variable X linearly depends on Z and is subject to a Gaussian-distributed noise e, and random variable Y is a linear combination of X and Z with noise f only when $Z > 0$. Based on the properties of linear combinations of Gaussian distributions, we can compute the conditional distributions:

$$\begin{aligned} &X|Z \sim \mathcal{N}(z, 1), \\ &Y|X, Z \sim \mathcal{N}(z + z \cdot x, n^2), \\ &Y|Z \sim \mathcal{N}(z + z^2, n^2 + z^2). \end{aligned} \tag{5.30}$$

We know that for bivariate Gaussian distributions with correlation coefficient ρ, the mutual information can be computed as [49]:

$$\mathcal{I} = -\frac{1}{2}\log(1 - \rho^2). \tag{5.31}$$

The conditional distributions $X|Z$ and $Y|Z$ can be expressed in terms of bivariate Gaussian distributions:

$$\begin{bmatrix} X|Z \\ Y|Z \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} z \\ z + z^2 \end{bmatrix}, \begin{bmatrix} 1 & z \\ z & z^2 + n^2 \end{bmatrix}). \tag{5.32}$$

Therefore, we can compute the correlation coefficient between $X|Z$ and $Y|Z$:

$$\rho = \frac{z}{\sqrt{z^2 + n^2}}. \tag{5.33}$$

Hence, the conditional MI $\mathcal{I}(X;Y|Z)$ is:

$$\begin{aligned} \mathcal{I}(X;Y|Z) &= -\frac{1}{2}\log(1 - \frac{z^2}{z^2 + n^2}) \\ &= \frac{1}{2}\log(1 + \frac{z^2}{n^2}). \end{aligned} \tag{5.34}$$

With the theoretical conditional MI, we can then evaluate the accuracy of different estimation approaches.

Equation 5.11, Equation 5.17 and Equation 5.23 in Section 5.1.3 provide three different approaches to estimate mutual information. Here we refer to them as the VLB bound, the KLD bound and the JSD bound respectively for convenience. We conduct tests on the X, Y and Z random variables described above with $\sigma_z = 1$

Table 5.1: Comparison of Mutual Information Estimation Approaches [34]

Dimension	Theoretical Average MI	Training Data Size	Root Mean Square Error (RMSE)		
			VLB	KLD	JSD
1	0.2911	20000	0.0713	0.1661	0.1594
		40000	0.0424	0.1291	0.1242
		60000	0.0502	0.1509	0.1785
2	0.5821	20000	0.0974	0.3745	0.2578
		40000	0.1121	0.3517	0.3292
		60000	0.0942	0.2139	0.2105
3	0.8732	20000	0.1594	0.4825	0.4573
		40000	0.1508	0.4828	0.4573
		60000	0.1407	0.4129	0.3176
4	1.1643	20000	0.2222	0.5879	0.5406
		40000	0.1665	0.6092	0.4101
		60000	0.1611	0.4928	0.4326

and $n = 0.5$. We use a neural network with one hidden layer of 256 units as the MI estimator for each approach. We compare the performance of the three different estimation approaches given different variable dimensions and different sizes of training data, and the results are shown in Table 5.1. The performance of each estimation approach is evaluated based on the root mean square error (RMSE) compared to the theoretical value of MI computed through Equation 5.34.

From Table 5.1 we can see that VLB has the lowest RMSE in all the test cases on this random variable set, whereas the KLD bound performs the worst in most cases. Another finding is that increasing the size of training data generally speaking helps improve the MI estimation accuracy, however, overfitting might happen when the dimensionality of random variables is too low and the training data size is too large. The "Theoretical Average MI" column in Table 5.1 is computed through randomly sampling 10000 z values from the Z distribution and averaging their theoretical MI values, and it provides a reference when analyzing the RMSE values. From the comparison between the RMSE and the absolute values of theoretical average MI we can see that it is possible to get a relatively accurate approximation of the conditional MI through numerical estimation when the mutual dependency between random variables are simple. Despite being promising, this result doesn't mean that numerical estimations with neural networks can work well with high-dimensional random variables with complicated dependencies. In terms of estimating the conditional MI of the continuous random variables we tested on, VLB performs the best in all cases and KLD performs the worst in most cases. However, there is also no guarantee that VLB will provide the best estimations for all distributions, but it did indicate the KLD bound can't provide a stable and relatively accurate estimation even for simple distributions. Therefore, in the robotics experiments in this thesis, we consider the VLB bound and the JSD bound only for mutual information estimation. We noticed that JSD is the best performer in experiments with the Fetch robot and VLB is the best performer in experiments with the PR2 robot, hence we will only report the results with the corresponding best performer in each environment.

5.2.3 Combination with ICM to Facilitate Empowerment Computation

Another challenging issue with empowerment-based RL is that well-balanced data are not easy to obtain at the beginning of training. If we initialize the RL agent with a random policy, it will highly likely explore much more of the empty space than regions with object interactions because the interaction-free part of the state space is often much larger. However, since $\mathbf{a}_t$ and $\mathbf{s}_{t+1}^{ex}$ are independent without interactions, the training data fed into the empowerment estimation network will be strongly biased towards the zero empowerment regions, which makes it very difficult to train accurate estimation models. Therefore, it is crucial that enough training data in the interacting part of the state space can be obtained at the beginning of training in order to get accurate estimations of empowerment. We achieve this through combining empowerment with the forward model of ICM introduced in Section 5.1.5 using adaptive coefficients, which initially place more weight on ICM rewards to ensure enough well-balanced data are fed to the empowerment estimation networks, and then switches more weight to empowerment rewards to encourage the robot to learn controllable behaviors.

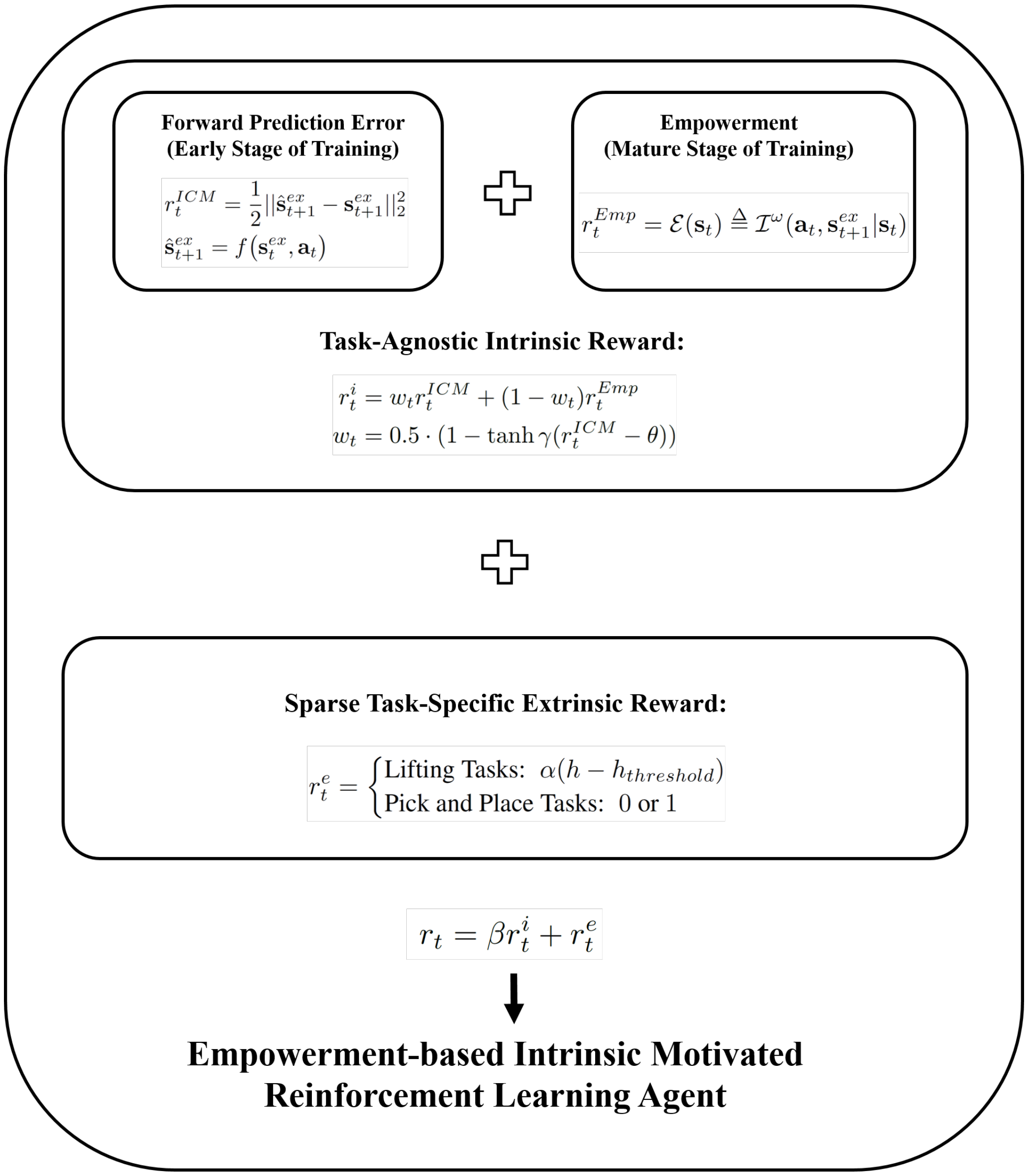

Forward Prediction Error
(Early Stage of Training)
$r_t^{ICM} = \frac{1}{2}||\hat{\mathbf{s}}_{t+1}^{ex} - \mathbf{s}_{t+1}^{ex}||_2^2$
$\hat{\mathbf{s}}_{t+1}^{ex} = f(\mathbf{s}_t^{ex}, \mathbf{a}_t)$
Empowerment
(Mature Stage of Training)
$r_t^{Emp} = \mathcal{E}(\mathbf{s}_t) \triangleq \mathcal{I}^{\omega}(\mathbf{a}_t, \mathbf{s}_{t+1}^{ex}|\mathbf{s}_t)$
Task-Agnostic Intrinsic Reward:
$r_t^i = w_t r_t^{ICM} + (1-w_t) r_t^{Emp}$
$w_t = 0.5 \cdot (1 - \tanh \gamma (r_t^{ICM} - \theta))$
Sparse Task-Specific Extrinsic Reward:
$r_t^e = \begin{cases} \text{Lifting Tasks: } \alpha(h - h_{threshold}) \\ \text{Pick and Place Tasks: } 0 \text{ or } 1 \end{cases}$
$r_t = \beta r_t^i + r_t^e$
Empowerment-based Intrinsic Motivated
Reinforcement Learning Agent

Figure 5-4: Overview of the empowerment-based intrinsic motivation approach [34]

Figure 5-4 summarizes the proposed empowerment-based intrinsic motivation approach. The core idea of this approach is to introduce a form of intrinsic reward r_t^i during RL in addition to the extrinsic task reward r_t^e which might be sparse or binary, so that the reinforcement learning process can be significantly accelerated. Note that the main difference between intrinsic reward and reward shaping is that intrinsic rewards are task-agnostic and represent the agent's motivation to explore the environment in a certain way, whereas reward-shaping is usually task-specific and it provides intermediate guidance for the agent to achieve the end goal. We use a hyper-parameter β to represent the intrinsic reward coefficient that trades off the intrinsic and extrinsic rewards during training. An adaptive coefficient w_t is determined at every training step to allocate the intrinsic reward between the empowerment reward r_t^{Emp} and the ICM reward r_t^{ICM}. w_t is close to 1 when r_t^{ICM} is very small, and is close to 0 when r_t^{ICM} is very large.

5.3 Empirical Evaluation

In this section, we present empirical evaluations of the empowerment-based intrinsic motivation approach in different robotic manipulation environments with different shapes of the target object. Experiment results demonstrate that this empowerment-based intrinsic motivation can outperform other state-of-the-art solutions to sparse-reward RL tasks and achieve higher extrinsic task rewards faster during learning. In addition, we also combine this approach with diversity-driven intrinsic motivation and show that the combination is able to encourage the manipulator to learn a diverse set of ways to interact with the object, whereas with the diversity-driven rewards alone the manipulator is only able to learn how to move itself in different directions.

5.3.1 Environment Setup

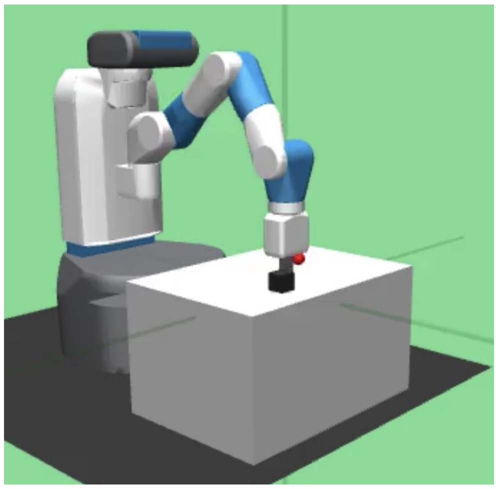
In order to evaluate the performance of the proposed empowerment-based intrinsic motivation approach in robotic manipulation tasks, we created four object-lifting tasks with different object shapes in OpenAI Gym [17] and Gazebo, as shown in

Figure 5-5, and compare the performance of our approach with other state-of-the-art intrinsic motivation approaches including Intrinsic Curiosity Module (ICM) [100] and Self-supervised Exploration via Disagreement [101] (referred to as the Disagreement method in this thesis). We also compare our approach with Hindsight Experience Replay (HER) [4] and vanilla RL without intrinsic motivation. The Gym environment uses a Fetch robot with a 25D state space (including the poses and velocities of the end-effector, the gripper and the object) and a 4D action space (including the actions of the end-effector and gripper), and the Gazebo environment uses a PR2 robot with a 38D state space (including the poses and velocities of all joints and the object) and an 8D action space (including the actions of manipulator joints). We also use the FetchPickAndPlace-V1 task provided in Gym in order to compare with HER because HER requires a goal-conditioned environment.

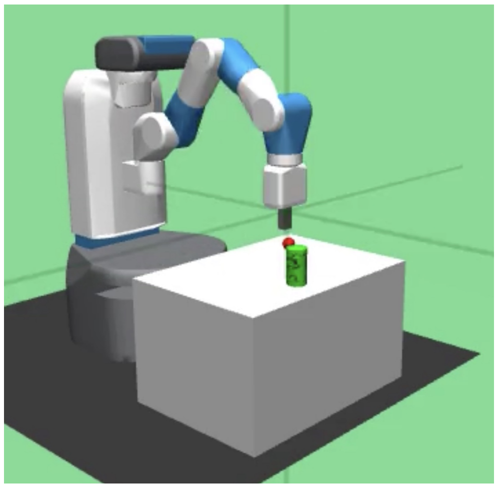
In the four object-lifting tasks, the goal is to lift up the object, and the extrinsic reward is only given when the object's height is above a threshold. In the pick-and-place task, the reward is given when the distance of the object to the goal pose is within a threshold. Our approach can be easily integrated with any standard RL algorithm, but in this thesis, we use Proximal Policy Optimization (PPO) [122] as the RL agent for all experiments to demonstrate its performance. Experiments on the Fetch robot use 60 parallel environments for training, and PR2 experiments use 40 due to its higher CPU requirement. An implementation of HER is directly adopted from OpenAI Baselines [38], and the implementation details of the other algorithms including hyperparameters and task rewards are provided in Section 5.3.2.

5.3.2 Implementation Details

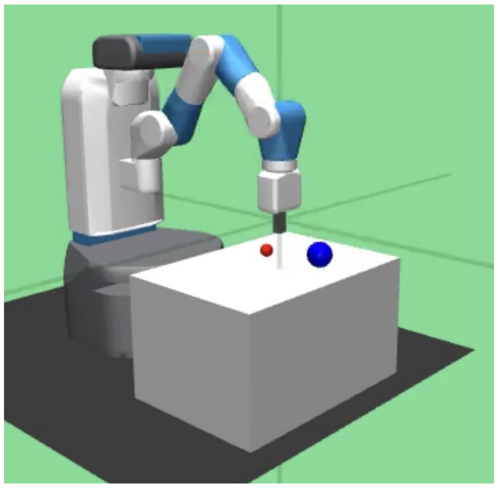
For the experiments shown in this chapter, we implemented the proposed empowerment-based intrinsic motivation approach, the ICM approach and the Disagreement approach as intrinsic rewards with an on-policy implementation of PPO. We use a three hidden-layer fully-connect neural network with (128, 64, 32) units in each layer and softsign as the activation function for both the policy network and the value network, and set $\gamma = 0.99$ and $\lambda = 0.95$ in the PPO algorithm. We use the Adam optimizer



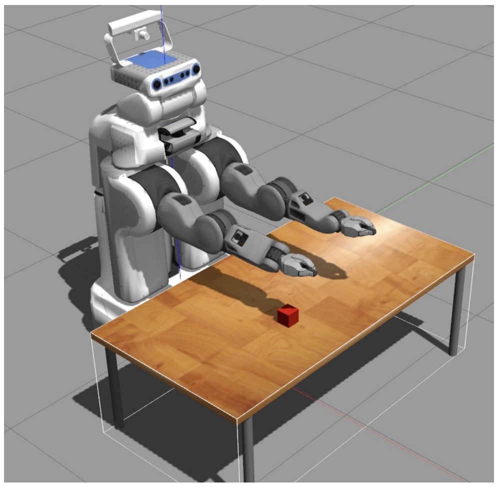
(a) Fetch with a box



(b) Fetch with a cylinder



(c) Fetch with a sphere



(d) PR2 with a box

Figure 5-5: The simulation environments used in this section include the Fetch robot in the Mujoco environment and the PR2 robot in the Gazebo environment. The robot needs to learn how to interact with objects of different shapes, including box, cylinder and sphere.

with learning rate 2e−4. All experiments shown in this chapter are conducted on a 10-core Intel i7 3.0 GHz desktop with 64 GB RAM and one GeForce GTX 1080 GPU.

ICM Implementation In the experiments in this chapter, since we assume pose estimations are available and do not directly take in image inputs, the inverse model of ICM is not necessary. In the ICM implementation, we train the forward model f by minimizing the forward loss:

$$\mathcal{L}_t^f = \frac{1}{2}||f(\mathbf{s}_t^{ex}, \mathbf{a}_t) - \mathbf{s}_{t+1}^{ex}||_2^2. \tag{5.35}$$

To compute the forward loss in the ICM approach, we use one 256-unit hidden layer in the network, and we didn't compute inverse loss because the observations in this chapter are poses instead of images. The value of the forward loss $\mathcal{L}_t^f$ is also used as the ICM intrinsic reward:

$$r_t^{ICM} = \mathcal{L}_t^f, \tag{5.36}$$

and we normalize r_t^{ICM} using running average before summing it up with the extrinsic reward to get the final reward for training the RL agent:

$$r_t = 0.01\bar{r}_t^{ICM} + r_t^e. \tag{5.37}$$

Disagreement Implementation In the Disagreement approach, we use the same network structure as in ICM and use five of these networks as the ensemble to compute the disagreement reward. We compute the forward losses for each of the five forward models in the same way as Equation 5.35, and sum up the five forward losses as the total loss to train the forward models. The intrinsic reward is calculated as:

$$r_t^{Dis} = var\{\hat{\mathbf{s}}_{t+1}^{ex,1}, \ldots, \hat{\mathbf{s}}_{t+1}^{ex,5}\}, \tag{5.38}$$

where $\hat{\mathbf{s}}_{t+1}^{ex,1}$ through $\hat{\mathbf{s}}_{t+1}^{ex,5}$ are the forward predictions made by the five forward models. We also use running average to get the normalized disagreement intrinsic reward $\bar{r}_t^{Dis}$

and then sum it up with the extrinsic reward to get the final reward for training the RL agent:

$$r_t = 0.01\bar{r}_t^{Dis} + r_t^e. \tag{5.39}$$

Empowerment Implementation For the neural network that makes empowerment prediction in the PR2 environment, we apply Gated Linear Units (GLU) [36] to improve performance. We use a neural network with four GLU layers with 256 gates each and two hidden fully-connected layers with (128, 64) units to predict $p(\mathbf{a}_t|\mathbf{s}_{t+1}^{ex}, \mathbf{s}_t)$, and calculate empowerment with the variational lower bound. Namely, we use

$$r_t^{Emp} = \log p(\mathbf{a}_t|\mathbf{s}_{t+1}^{ex}, \mathbf{s}_t) - \log p(\mathbf{a}_t|\mathbf{s}_t) \tag{5.40}$$

as the empowerment intrinsic reward so that in expectation, the empowerment reward being maximized is equivalent to the empowerment defined in Equation 5.28. In the Fetch environment, we use a neural network with six hidden fully-connected layers with (512, 512, 216, 128, 64, 32) units to approximate the T function in Equation 5.24 and calculate empowerment with the JS-Divergence approximation. In order to approximate the supremum in Equation 5.24, we use the following loss function in order to train T network:

$$\mathcal{L}_t^{Emp} = \mathrm{sp}(-T(\mathbf{a}_t, \mathbf{s}_t, \mathbf{s}_{t+1}^{ex})) + \mathrm{sp}(T(\tilde{\mathbf{a}}_t, \mathbf{s}_t, \mathbf{s}_{t+1}^{ex})) - \log 4, \tag{5.41}$$

where $\mathbf{a}_t$ is the true action executed at time step t and $\tilde{\mathbf{a}}_t$ is sampled from the policy. The empowerment intrinsic reward in the Fetch environment is:

$$r_t^{Emp} = T(\mathbf{a}_t, \mathbf{s}_t, \mathbf{s}_{t+1}^{ex}). \tag{5.42}$$

In our empowerment-based intrinsic motivation implementation, empowerment reward and ICM reward are combined through weight coefficients to ensure that the agent can collect enough data in the nonzero empowerment region to train the empowerment network well before it is used as the intrinsic reward. The weight coefficients used in this chapter are:

$$\begin{aligned} w_t^{ICM} &= 0.5 \times (1 - \tanh(200(r_t^{ICM} - 0.12))), \\ w_t^{Emp} &= 1 - w_t^{ICM}, \end{aligned} \tag{5.43}$$

where r_t^{ICM} is the forward prediction error (computed through Equation 5.35 and 5.36) averaged from all the parallel environments at time step t. These weight coefficients make sure that at the beginning of training when the robot don't have much interaction with the object, the coefficient for ICM reward is near 1 and the coefficient for empowerment reward is near 0. After the average ICM reward reaches a certain threshold, which means the robot have learned to interact with the object and the empowerment network can obtain enough meaningful data to get well trained, the coefficient for ICM reward switches to near 0 and the coefficient of the empowerment reward switches to near 1. Then this intrinsic reward and extrinsic task reward are combined as the RL algorithm reward:

$$\begin{aligned} r_t^i &= w_t^{ICM} \bar{r}_t^{ICM} + w_t^{Emp} \bar{r}_t^{Emp}, \\ r_t &= 0.01 r_t^i + r_t^e, \end{aligned} \tag{5.44}$$

where $\bar{r}_t^{ICM}$ and $\bar{r}_t^{Emp}$ are normalized using running average.

Extrinsic Task Rewards In the box-lifting task and the pick-and-place task in the Fetch environment, the object is a cube with 0.05 m edges. In the cylinder-lifting environment, the height of the cylinder is 0.1 m and the radius is 0.03 m. In the sphere-lifting environment, the radius of the sphere is 0.04 m. In both the box-lifting and sphere-lifting task, the task reward is given as Equation 5.45 under the condition that the center of the grippers is less than 0.01 m away from the center of the object. In the cylinder-lifting task, the condition for giving task reward is the same, but the reward is given as Equation 5.46. In the pick-and-place task, the task reward is 1 when the object pose is within 0.05 m of the target pose, and 0 otherwise.

$$\text{Fetch with box or sphere: } r_t^e = 50 \cdot (h - 0.01), \tag{5.45}$$

$$\text{Fetch with cylinder: } r_t^e = 500 \cdot (h - 0.01), \tag{5.46}$$

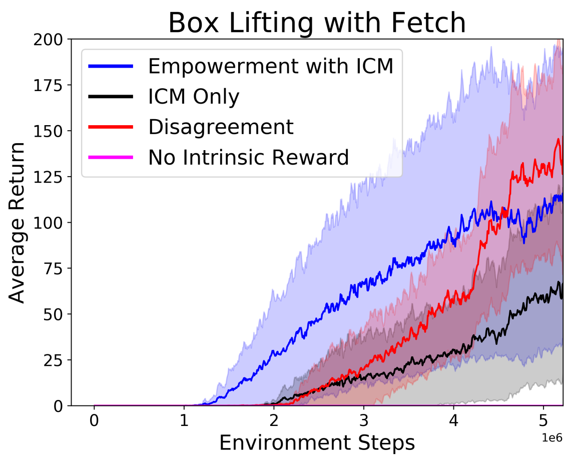
In the box-lifting task in the PR2 environment, the object is a cube with 0.06 m edges, and the task reward is given as Equation 5.47 under the condition that both grippers are in contact with the object and the object height is at least 0.012 m above the tabletop.

$$\text{PR2 with box: } r_t^e = 500 \cdot (h - 0.012). \tag{5.47}$$

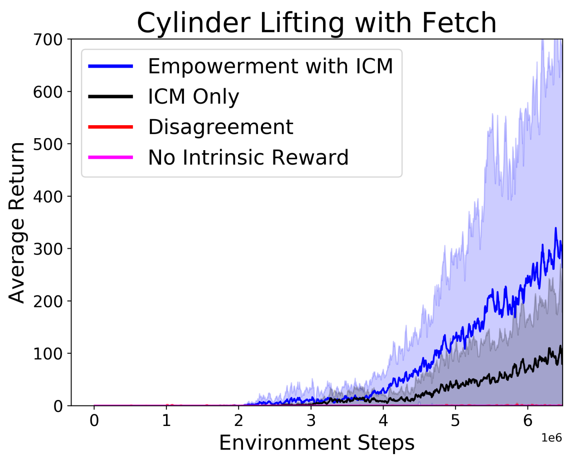
5.3.3 Experiment Results

In this section, we provide experiment results that compare the proposed empowerment-based intrinsic motivation approach with other state-of-the-art algorithms, including ICM [100], exploration via disagreement [101] (referred to as Disagreement in this thesis) and HER [4]. We use our implementation of ICM and Disagreement, and use the OpenAI Baselines implementation [38] for HER. In both ICM and Disagreement, we also make the same assumption as in the empowerment implementation that the state space can be divided into intrinsic states and extrinsic states, and only the prediction error or variance of the extrinsic states contribute to the intrinsic rewards. We run HER with 2 MPI processes with 30 parallel environments each to make sure it is equivalent to the 60 parallel environments in other experiments. Other parameters for HER are set to default. All the results in the Fetch environment are averaged over 10 different random seeds, and the results in the PR2 environment are averaged over 8 random seeds.

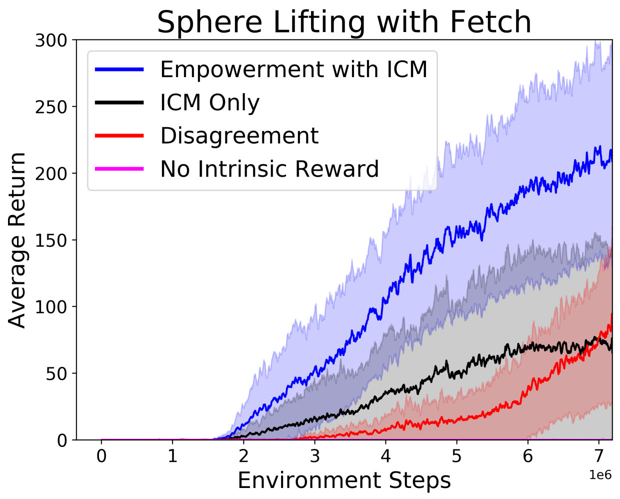
Figure 5-6(a)-(c) compare the performance of our approach with ICM, Disagreement, and PPO without any intrinsic reward in the object-lifting tasks with a Fetch robot, and Figure 5-7 compares our approach with ICM and Disagreement in box-lifting tasks with a PR2 robot. In the Fetch environment, the cylinder lifting task is



(a)



(b)



(c)

Figure 5-6: Experiment results in the Fetch environment. (a)-(c) compare the performance of the proposed empowerment-based approach (referred to as empowerment with ICM since ICM is used to help training the empowerment prediction networks) with ICM and Disagreement in object lifting tasks. The solid lines represent the mean, and the shadow areas represent the 95% confidence intervals. [34]

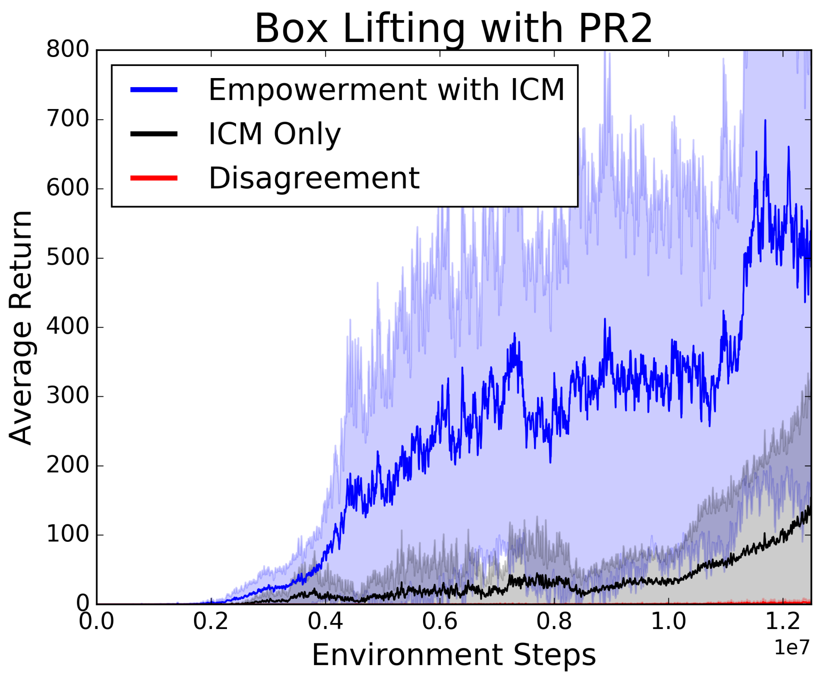


Figure 5-7: Experiment results in the PR2 environment comparing the performance of the proposed empowerment-based approach (referred to as empowerment with ICM since ICM is used to help training the empowerment prediction networks) with ICM and Disagreement in object lifting tasks. The solid lines represent the mean, and the shadow areas represent the 95% confidence intervals. [34]

much more difficult compared to box lifting and sphere lifting, thus we use a larger scale α for extrinsic lifting reward. Similarly, we also use a larger α for the box-lifting task with the PR2 robot since this environment is much higher-dimensional and hence more difficult for an RL agent. From Figure 5-6(a)-(c) we can see that the reward curve for PPO without any intrinsic reward remains almost zero, which proves that sparse reward tasks are very challenging for vanilla RL algorithms. In all four environments, our empowerment-based approach is able to help the robot achieve higher lifting rewards faster than other approaches we compared with. The Disagreement approach is able to perform better in the box lifting task with the Fetch robot after training for a long time, but it performs much worse than the other two intrinsic motivations in the cylinder and sphere lifting tasks. Another finding from Figure 5-6(a)-(c) is that the advantage of the empowerment-based intrinsic motivation is much more obvious in the cylinder and sphere lifting tasks compared to the box lifting tasks. We hypothesize that this is because the ability of "controlling" the object is much more important when there are round surfaces, since these objects are more difficult to pick up and also more likely to randomly roll around when novelty is the only intrinsic motivation. In fact, in the cylinder lifting task, our empowerment-based intrinsic motivation is the only approach that allows the agent to learn to directly pick up the cylinder from the top without knocking it down first, whereas agents trained with ICM will knock down the cylinder and then pick up radially. Videos of this behavior can be found at `https://sites.google.com/view/empowerment-for-manipulation/`. In Figure 5-7, although the confidence intervals are wider due to the smaller number of runs, we can still get the similar conclusion that our approach shows the best performance in the box lifting task in the PR2 environment.

Figure 5-8 compares the empowerment-based intrinsic motivation with HER in the Fetch pick-and-place environment. We can see that although the average success rate of HER goes up much faster, it stays at about 0.5 even after a long time of training. In fact, the maximum value dashed line in Figure 5-8 shows that none of the 10 runs of HER has reached a success rate of 0.6 or above. In contrast, although the empowerment approach is slower in the initial learning phase, in 3 out of 10 runs

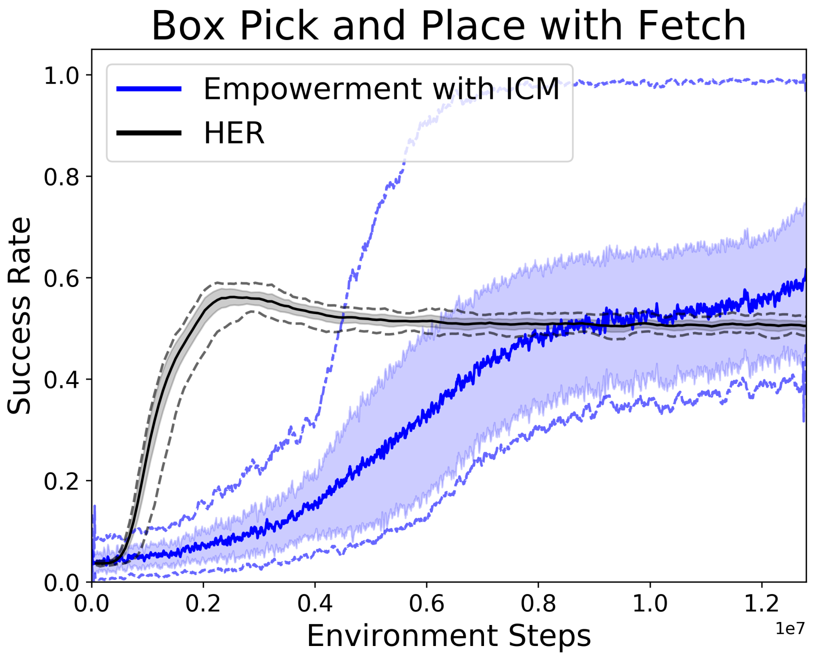


Figure 5-8: Experiment results in the Fetch environment comparing the proposed empowerment-based approach with HER in pick-and-place tasks. The solid lines represent the mean, the shadow areas represent the 95% confidence intervals, and the dashed lines in represent the maximum and minimum values. [34]

it has learned to lift up the object and reach the goals in the air accurately and quickly, and the success rate stays at about 1 in these tests. This is because in the Gym FetchPickAndPlace-V1 task, half of the goals are sampled from on the table and half are sampled in the air, thus agents that only learned to push can still reach the goals close to the tabletop and receive a success rate of about 0.5, but only agents that actually learned to pick and place will reach a success rate of 1.0.

5.3.4 Off-Policy Implementation

Our algorithm can also be used on off-policy RL algorithms but requires additional adaptation. This is because intrinsic rewards are not "ground truth" rewards and their values are not very meaningful until the neural networks are trained to predict intrinsic rewards well. Since the estimation of conditional mutual information is very challenging and the empowerment networks typically take a long time to get well trained, mixing up experiences with reward values predicted at different training steps in the same replay buffer will influence the overall performance and makes off-policy training very tricky. Therefore, we implemented an off-policy version by recomputing the intrinsic reward values with the updated network parameters every time the algorithm draws experiences from the replay buffer. We demonstrate its performance in the sphere lifting environment in Gym in Figure 5-9 and show that the off-policy implementation is much more sample-efficient.

5.4 Application: Learning a Diverse Set of Skills

Besides its advantage in solving sparse reward RL tasks, another driving force for research on intrinsic motivation is its potential in unsupervised skill discovery. Many Hierarchical Reinforcement Learning (HRL) frameworks allow RL agents to learn policies of different levels so that high-level policies only need to focus on the skill-space that low-level controllers provide instead of the raw state-space. However, the skills an end-to-end HRL system can learn are limited and they often require guidance from human-designed "curricula" [9, 116, 26]. In contrast, skills discovered by

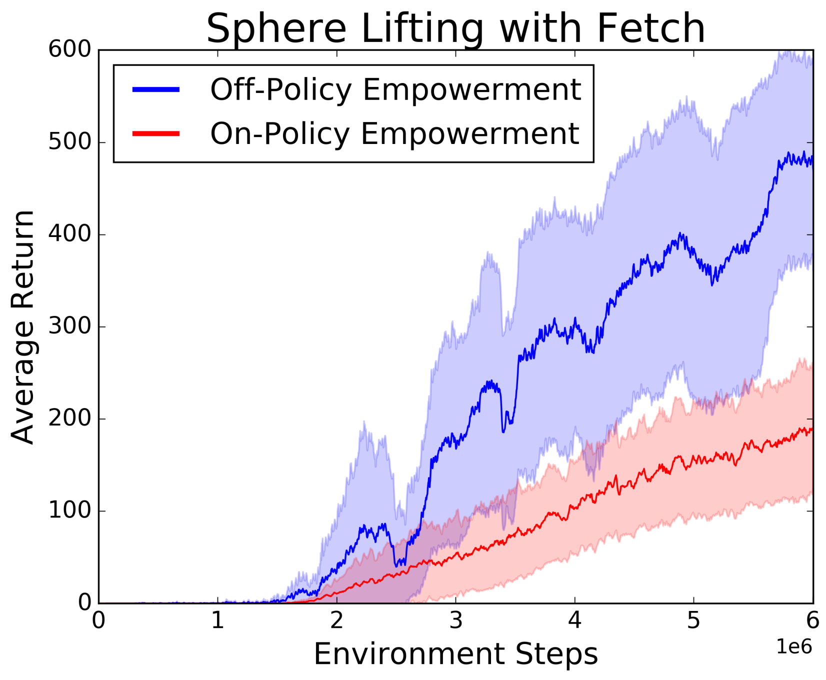


Figure 5-9: Comparison of off-policy implementation and on-policy implementation of the empowerment-based intrinsic exploration approach in the sphere lifting environment. The solid lines represent the mean of 10 experiments with different random seeds, and the shadow areas represent the 95% confidence intervals. [34]

intrinsic motivations can reduce HRL frameworks' dependence on human engineering and potentially enable them to learn more complicated tasks. Ultimately, we hope the empowerment-based intrinsic motivation proposed in this chapter can also be incorporated into a HRL framework and contribute to the learning of complicated manipulation skills, such as opening a container and stacking objects inside. In order to see what type of skills an agent can learn with our approach, we provide preliminary qualitative results combining empowerment and the Diversity is All You Need (DIAYN) approach [41] in the "Fetch with a box" environment. We evaluate this combination with two different numbers of skills in DIAYN: 3 skills shown in Figure 5-10 and 5 skills shown in Figure 5-11. In order to show the advantage of using our empowerment-based intrinsic motivation during unsupervised skill discovery, we also evaluate the performance of DIAYN only in the same task for comparison and show the results in the 5 skills scenario in Figure 5-12. From Figure 5-11 and 5-12 we can compare the skills learned by combining empowerment and DIAYN as the intrinsic reward and the skills learned with only DIAYN as the intrinsic reward. From Figure 5-12 we can see that without an intrinsic motivation that drives the agent to control the object, the skills learned through a purely diversity-driven approach are not meaningful in terms of solving manipulation tasks because they don't involve interactions with the object. In comparison, Figure 5-11 demonstrates the potential of this combined intrinsic reward in terms of learning a set of meaningful manipulation skills, including pushing the object to different directions and lifting the object up. Videos of the learned skills can be found at `https://sites.google.com/view/empowerment-for-manipulation/`.

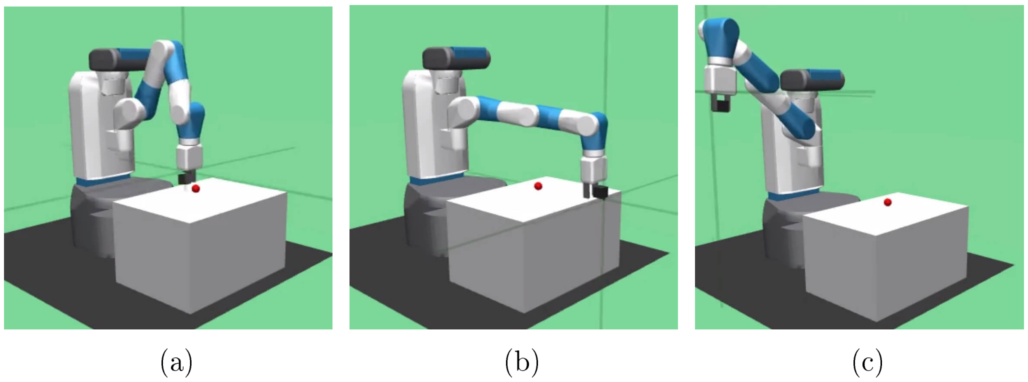


Figure 5-10: Qualitative performance of the proposed empowerment-based intrinsic motivation when combined with the diversity-driven DIAYN [41] approach in the box lifting task with a Fetch robot. (a)-(c) show the different skills learned when the number of skills in DIAYN is set to 3. [34] From the figures we can see that the robots has learned different ways of interacting with the object.

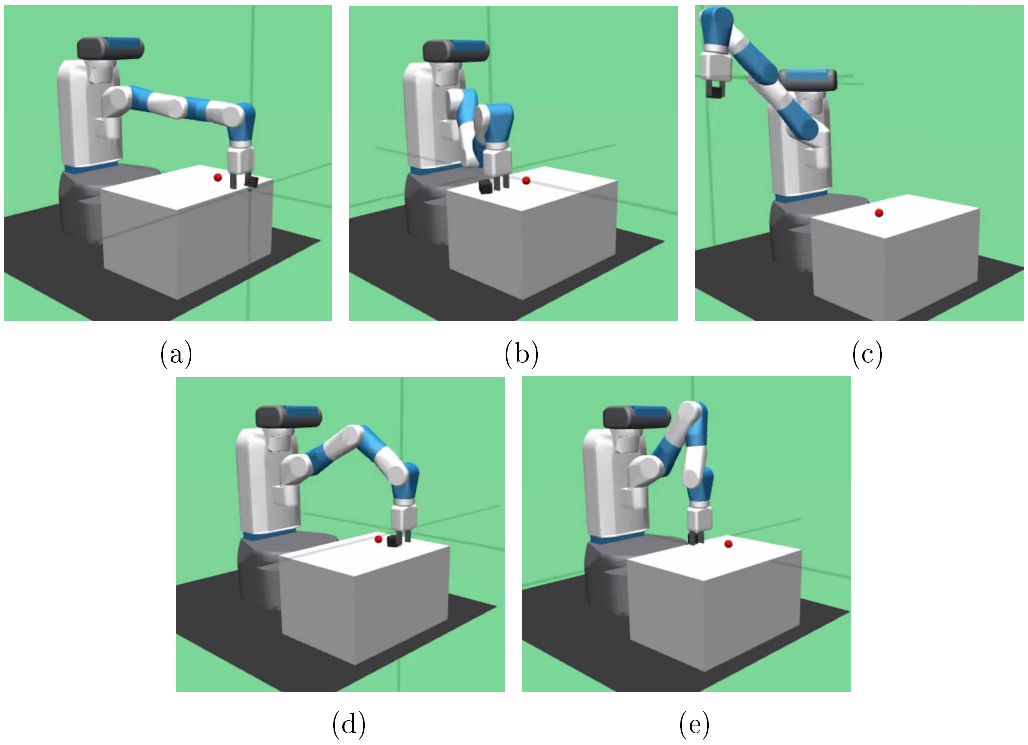


Figure 5-11: Qualitative performance of the proposed empowerment-based intrinsic motivation when combined with the diversity-driven DIAYN [41] approach in the box lifting task with a Fetch robot. (a)-(e) show the different skills learned when the number of skills in DIAYN is set to 5. [34] From the figures we can see that the robots has learned different ways of interacting with the object.

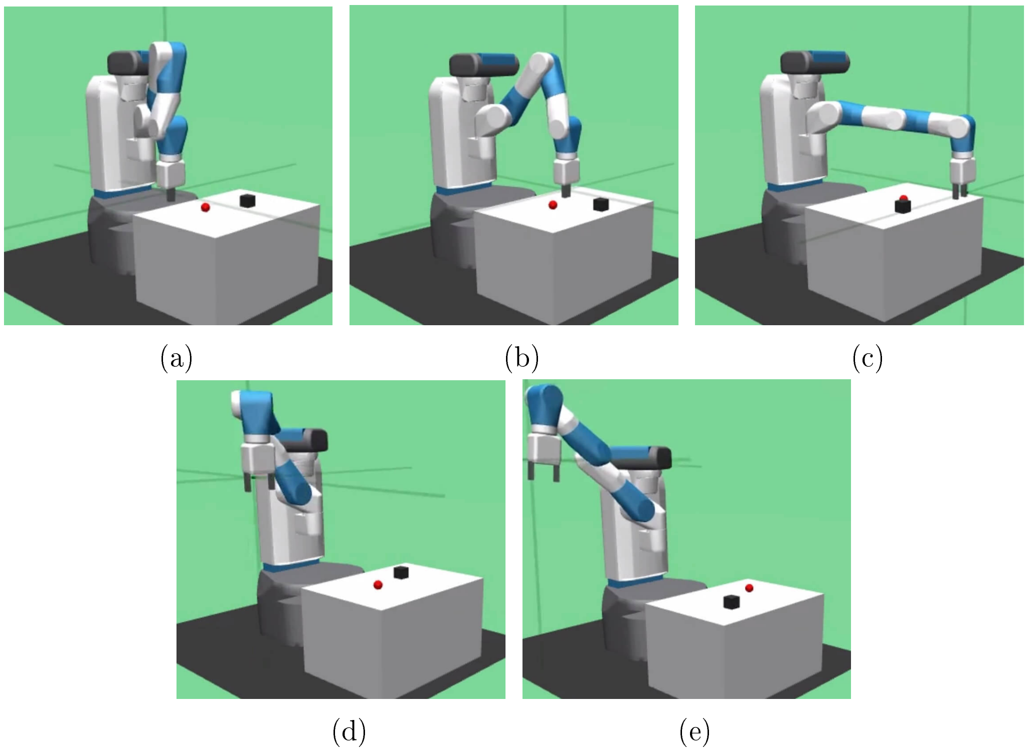


Figure 5-12: Different skills learned with DIAYN [41] without the empowerment-based intrinsic motivation in the box lifting task with a Fetch robot when the number of skills is set to 5. [34] From the figures we can see that the skills the robot has learned do not involve interactions with the object.

Chapter 6

Automatic Curricula via Expert Demonstrations

This chapter presents the third contribution of this thesis: Automatic Curricula via Expert Demonstrations (ACED). Section 6.1 explains the key concepts necessary for understanding the ACED approach, and Section 6.2 introduces the ACED algorithm. Section 6.3 describes the empirical evaluation environments and the implementation details, Section 6.4 presents the empirical evaluation results of ACED in a block pick-and-place task and a block stacking tasks, and Section 6.5 compares ACED with a simple combination of behavior cloning and reinforcement learning to demonstrate the significance of curriculum learning. In Section 6.6, we propose a combination of ACED and empowerment-based intrinsic motivation and evaluate it in block pick-and-place tasks.

6.1 Preliminaries

ACED aims at solving robotic manipulation tasks by providing a small number of demonstration trajectories and formulating a sequence of curriculua by creating RL tasks using demonstration states. The ACED approach presented in this Chapter can work with any standard RL algorithm, though in this thesis we only demonstrate its performance using Proximal Policy Optimization (PPO) [122] with multiple parallel rollout workers.

6.1.1 Behavior Cloning

Given expert demonstration trajectories $\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$ where each trajectory includes state-action pairs, i.e. $\tau_e = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T)$, the objective of Behavior Cloning (BC) is to learn a mapping from states to actions through supervised learning in order to imitate expert behaviors. Due to BC's demand for a large amount of demonstrations and its poor generalization performance in unseen states, it is often used to pre-train the policy network for other imitation learning or RL approaches as a warm start instead of as a standalone imitation learning approach. The ACED method introduced in this chapter can also be combined with BC by using it to pre-train policy networks, and we present this combination in Section 6.2. We compare the performances of ACED with or without BC in Section 6.4. Note that demonstration trajectories with state-action pairs are only required by BC, and if ACED is used without BC, then demonstration trajectories with only states are sufficient.

6.2 Approach

In order to solve long-horizon manipulation tasks with binary rewards, ACED constructs a curriculum by sampling states from expert demonstration trajectories as initializations for each training episode, where the samples initially come from near the end of the demonstration trajectories and gradually move forward as the agent improves its performance [30].

The intuition of ACED is shown in Figure 6-1 with a block stacking example. The robot is trying to stack a yellow block on top of a green block at a pre-specified location on the tabletop, and it has received a demonstration trajectory that accomplished the task. From the demonstration trajectory we can see that the world state is moving closer and closer towards the goal state as time passes despite the back-and-forth in the Euclidean space, and states towards the end of the demonstration trajectory are closer

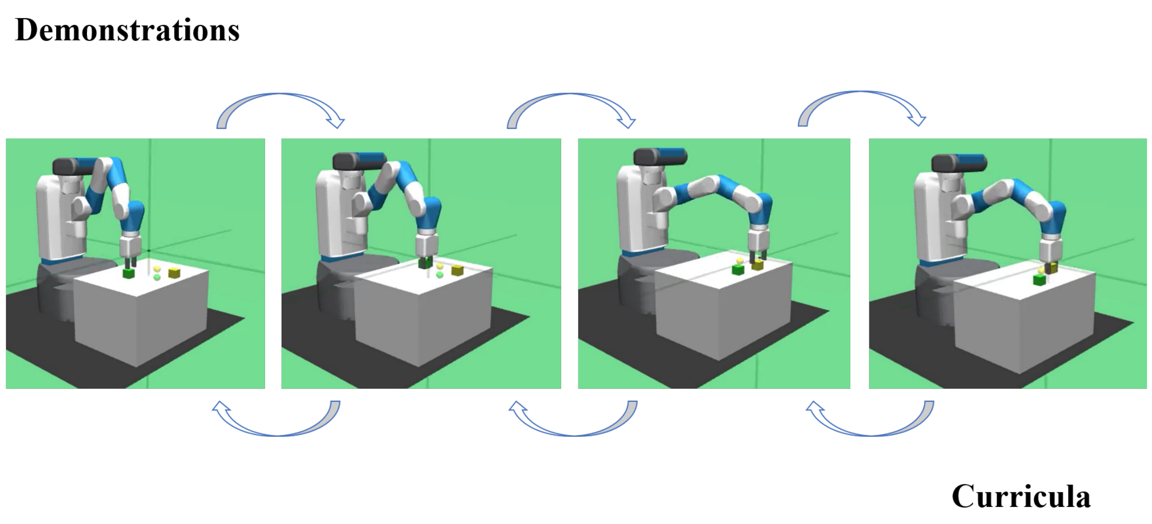


Figure 6-1: ACED Intuition

to the goal state within the task space compared to the states towards the beginning. In other words, if the green block is already placed at its goal and the yellow block is already in the robot's hand, then the robot is solving a much easier task compared to the ones trying to accomplish the same task with randomly placed blocks to start with. Therefore, if we take the states along this demonstration trajectory to initialize training episodes, then we can naturally form a set of curricula with increasing task difficulties if we start RL training with states closer to the end of demonstration and move towards the beginning as the RL agent improves its performance. This is the key intuition behind the ACED approach proposed in this chapter.

In ACED, we first collect a set of expert demonstration trajectories $\mathcal{T}$ and represent each trajectory $\tau_e \in \mathcal{T}$ as a discrete sequence of states at each time step: $\tau_e = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_t, \ldots, \mathbf{s}_{T-1}, \mathbf{s}_T)$, where the initial state $\mathbf{s}_0$ is randomly sampled from the initial state distribution S_0 and the final state $\mathbf{s}_T$ has a probability of $p_{success}$ to reach a goal state randomly sampled from the goal distribution S_g, i.e. $\mathbb{P}(\mathbf{s}_T = \mathbf{s}_g) = p_{success}, \mathbf{s}_g \in S_g$. We refer to $p_{success}$ as the expert success rate. Each trajectory τ_e is then evenly divided into C_{max} sections, where section-C_{max} denotes the section at the beginning of τ_e (near $\mathbf{s}_0$) and section-1 denotes the one at the end (near $\mathbf{s}_T$). C_{max} is a hyperparameter referred to as the *total number of curricula.* Figure 6-2 provides an illustration of an example demonstration trajectory and its

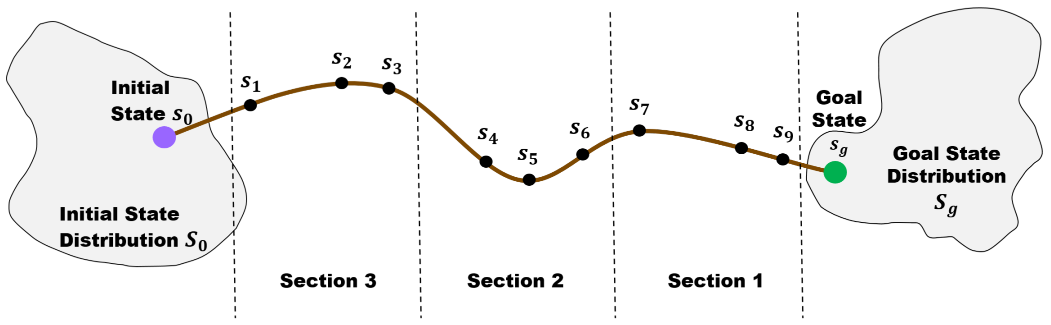


Figure 6-2: Example of demonstration trajectory segmentation: an expert demonstration trajectory can be divided into sections where larger section number indicates being closer to the initial state. The total number of sections is also called the total number of curricula C_{max}, and in this example $C_{max} = 3$. Normal rollout workers randomly sample an initial state from the initial state distribution S_0 and a goal state from the goal state distribution S_g. For curriculum rollout workers, the environments are reset based on the curriculum number C: curriculum-C tasks reset the environment to a section-C state on a randomly selected demonstration trajectory. ACED starts training with $C = 1$ and gradually moves reset states towards the beginning of demonstration trajectories by increasing C. When ACED switches to normal rollout workers, the reset states are drawn from the actual S_0 the target task specifies, and this is when it starts to generalize to unseen initializations. [30]

segmentation. In Figure 6-2, a demonstration trajectory is divided into three segments, which corresponds to a total number of curricula of $C_{max} = 3$. Section-1 represents the section closest to the goal state distribution S_g and Section-3 represents the one closest to the initial state distribution S_0. A key assumption made in ACED is that all expert demonstrations are reasonable solutions to the problem and don't contain unnecessary detours, which guarantees that two states that are close in the demonstration trajectory are also close in the task space.

We consider two different versions of the ACED algorithm: ACED with BC and ACED without BC. Algorithm 2 describes the overall framework of ACED with BC. Given the expert demonstration set $\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$, we can pretrain the network parameters of the policy π of the RL algorithm with BC (line 1). We refer to this version of the algorithm as ACED with BC, and if Algorithm 2 is executed without line 1, then we call it ACED without BC. The performances of the two versions are compared in pick-and-place tasks in Section 6.4.1. We use a set of parallel environments to generate rollout data for RL training. We refer to the original environment that resets to initial states sampled from S_0 as the *normal rollout worker*, and the curriculum-based environment that resets to demonstration states the *curriculum rollout worker*. At every environment step, the normal rollout workers simply take as input the action predicted by the policy network $\pi(\mathbf{s}_t)$ and returns the state at the next time step $\mathbf{s}_{t+1}$ after simulating for one step. Details of the curriculum rollout worker is described in Algorithm 3. The main difference between the curriculum rollout workers and the normal rollout workers is that, at the beginning of each episode, curriculum rollout workers initialize the environment to states sampled from demonstration states, and normal rollout workers initialize the environment with states sampled from the initial state distribution S_0.

At the beginning of training, all parallel environments are set to be curriculum rollout workers (as shown in Algorithm 2 line 3), and the switch from curriculum rollout workers to normal rollout workers are controlled by *curriculum number* $C \in \{1, 2, \ldots, C_{max}\}$. At each iteration, Algorithm 2 will collect training data using rollout workers and optimize the policy using the RL algorithm of choice (line 6 - 8). C is initialized to be 1, and every t iterations the algorithm will check the average return from the most recent n episodes and compare it with a threshold ϕ (line 9 - 15). When the average return exceeds the threshold ϕ, we add 1 to the current curriculum number C if it hasn't reached C_{max}, or switch to the normal rollout worker if C has reached C_{max}.

At each rollout, as shown in Algorithm 3, the curriculum rollout worker will first randomly select a demonstration trajectory τ_e and divide it into C_{max} sections with equal number of states (line 3). Based on the current curriculum number C, the curriculum rollout worker resets the environment to a randomly selected state from section-C on the demonstration trajectory τ_e (line 4 and 5). It will then rollout a trajectory using the current policy and return it to Algorithm 2. In our implementation, each rollout worker keeps track of its own curriculum number and the switch from a curriculum rollout worker to a normal rollout worker is independent from other parallel workers' C value.

Algorithm 2: Automatic Curriculum Learning with Demonstrations

```
Input:
$T$: number of iterations
$C_{max}$: number of curricula
$\mathcal{T} = \{\tau_1, \ldots, \tau_N\}$: demonstration trajectories
$\phi$: curriculum switching threshold for average return
$t$: period for checking average return
$n$: number of episodes used to compute average return
Output:
$\pi$: policy
1  Initialize policy parameters with Behavior Cloning Algorithm
2  Initialize curriculum number $C \leftarrow 1$
3  Initialize rollout worker $W \leftarrow$ CurriculumRolloutWorker
4  Initialize experience $\mathcal{E} \leftarrow \{\}$
5  for $i = 1, 2, \ldots, T$ do
6      $\tau \leftarrow W$.rollout$(C, C_{max}, \mathcal{T}, \pi)$
7      Add $\tau$ to $\mathcal{E}$
8      Send $\mathcal{E}$ to RL Algorithm and update $\pi$
9      if $i \mod t == 0$ then
10         $R \leftarrow$ Evaluate the average return on the most recent $n$ episodes
11         if $R \geqslant \phi$ then
12             if $C < C_{max}$ then
13                 $C \leftarrow C + 1$
14             else
15                 $W \leftarrow$ NormalRolloutWorker
16             end
17         end
18     end
19 end
```

Algorithm 3: CurriculumRolloutWorker

```
Input:
C: current curriculum number
C_max: total number of curricula
T = {τ_1, ..., τ_N}: demonstration trajectories
π: current policy
Output:
τ: rollout trajectory
1 Randomly select a trajectory from demonstrations $\tau_e \in \mathcal{T}$
2 $num\_transitions = \text{len}(\tau_e) - 1$ /* Make sure to not sample the goal state */
3 $interval = \text{RoundDown}(num\_transitions / C_{max})$ /* Divide the demonstration
    trajectory $\tau_e$ into $C_{max}$ intervals */
4 $index = \text{RandInt}(interval) + interval \times (C_{max} - C)$ /* Randomly select a state
    from the segment of the demonstration trajectory corresponding to the
    current curriculum $C$ */
5 $\mathbf{s}_{init} = \tau_e[index]$
6 $\tau = \text{Rollout}(env, \mathbf{s}_{init}, \pi)$
```

6.3 Experiment Setup

We evaluate the ACED approach on two tasks in the Fetch environment in OpenAI Gym [17]: a pick-and-place task and a block stacking task, as shown in Figure 6-3. The pick-and-place task is adapted from Gym directly and the block stacking task is adapted from [70]. The goal of the pick-and-place task is to move a block randomly placed on the tabletop to a goal pose that could be either in the air or on the tabletop, and the goal of the stacking task is to move two randomly placed blocks to their corresponding goal pose where the yellow block is stacked on top of the green block on the tabletop. The majority of results presented in this section use Proximal Policy Optimization (PPO) [122] as the RL algorithm, and we demonstrate ACED's off-policy performance with Deep Deterministic Policy Gradient (DDPG) [77] in the pick-and-place task. In the experiments in this section, we compare the performance of ACED with BC, ACED without BC, RL + BC without ACED, and other state-of-the-art automatic curriculum generation approaches [45, 118]. We refer to the approach presented in [45] the reverse curriculum method, and the one presented in [118] the Montezuma's Revenge method.

The demonstration trajectories in our experiments are generated from a hand-

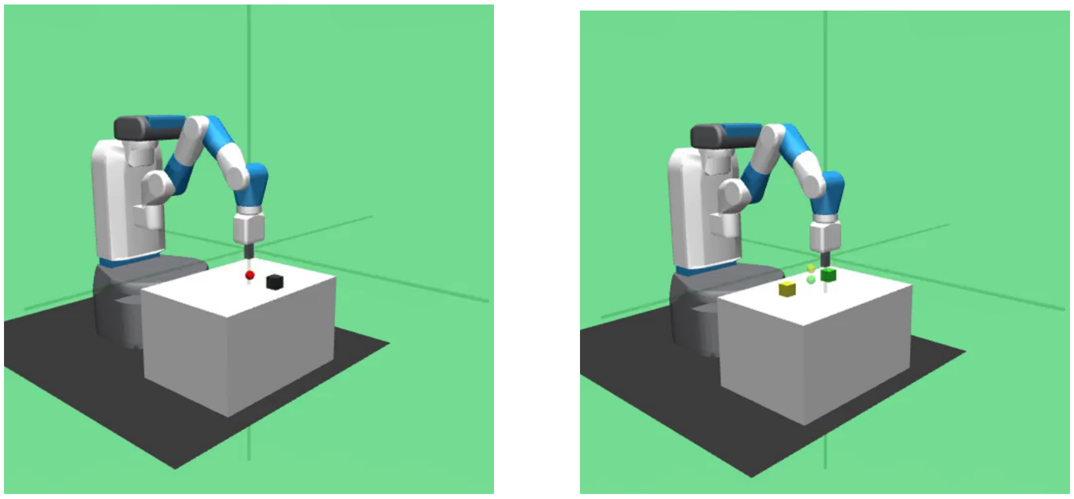

(a) Fetch pick-and-place environment (b) Fetch block stacking environment

Figure 6-3: Simulation environments for ACED evaluation

coded straight-line policy, i.e. the robot is instructed to follow a straight-line trajectory to reach the object, grasp the object and then follow a straight-line trajectory to reach the goal. For both tasks, we use a binary reward function, where $r = 1$ indicates all blocks are within the distance threshold to their corresponding goal poses and $r = 0$ otherwise. The episode is terminated immediately if $r = 1$ is obtained even if the maximum episode length hasn't been reached. Additional implementation details are presented in Section 6.3.1. In the results presented in this section, we mainly focus on the convergence performance and the success rate, but selected examples of the learning curves are shown in Section 6.4.3.

6.3.1 Implementation Details

All experiments shown in this section are conducted on a 10-core Intel i7 3.0 GHz desktop with 64 GB RAM and one GeForce GTX 1080 GPU. In our implementation of the PPO algorithm [122], we use a three hidden-layer fully-connect neural network with (128, 64, 32) units in each layer for both the policy network and the value network, and set $\gamma = 0.99$ and $\lambda = 0.95$. We noticed that PPO training can be unstable due to the frequent curriculum switches especially in the block stacking environment, and found that in order to prevent collapsing during training, it is very helpful to use a small importance ratio clipping parameter in PPO (denoted as ϵ in [122]) together with an optimizer with small learning rates and gradient clipping. In pick-and-place tasks, we set $\epsilon = 0.2$ and use the Adam optimizer with a learning rate of 2e-4 without gradient clipping. In stacking tasks, we set $\epsilon = 0.05$ and use the Adam optimizer with a learning rate of 1e-4 and gradient clipping-by-norm with a clipping factor of 0.05. In our DDPG [77] implementation, the learning rate is set to 2e-4, and the same policy network is used as in the PPO implementation. The target update period in DDPG is set to 5.

In the ACED algorithm, we use $\phi = 0.9$ as the curriculum switching threshold in pick-and-place tasks, and $\phi = 0.85$ in block stacking tasks. The average return checking period is set to $t = 120$, and the number of episodes used to compute the average return is set to $n = 3$. 60 parallel rollout workers are used in both tasks. In pick-and-place tasks, the threshold for the object's distance to the goal to assign reward $r = 1$ is 0.05, and in stacking tasks the threshold is set to 0.04. The maximum number of steps in an episode is set to 50 in pick-and-place tasks, and 100 in block stacking tasks. In BC, an Adam optimizer with the learning rate of 2e-4 is used, and the loss function is negative log likelihood.

In the reverse curriculum [45] implementation, we use 1000 random start states and a time horizon of 5 time steps for the Brownian motion. 200 old sampled start states are appended to the new start states at each training step. In the our implementation of the Montezuma's Revenge method [118], we randomly select one demonstration trajectory and set the curriculum switching threshold also to $\phi = 0.85$. Both the reverse curriculum method and the Montezuma's Revenge method are implemented with the same PPO algorithm as used in the ACED implementation.

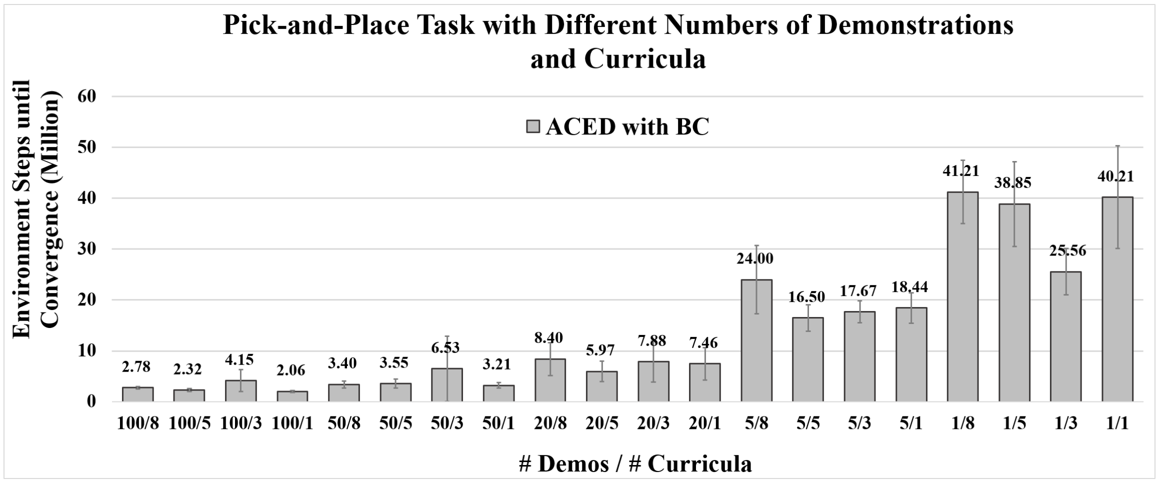


Figure 6-4: Number of environment steps ACED with BC takes to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval. [30]

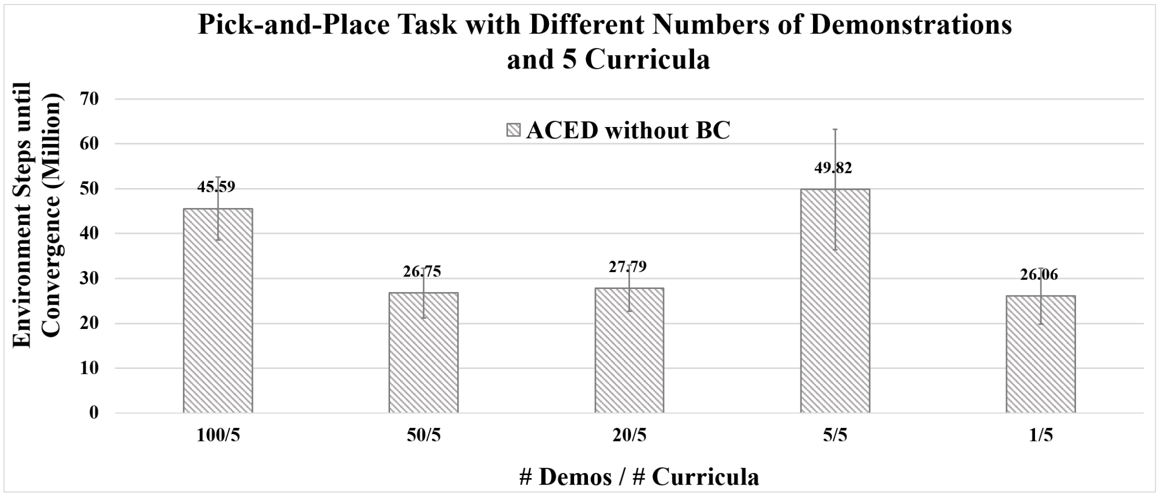


Figure 6-5: Number of environment steps ACED without BC takes to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval. [30]

Table 6.1: Pick-and-Place Success Rate [30]

Algorithm	Number of Curricula[1]	Number of Demonstrations				
		$\lvert\mathcal{T}\rvert = 100$	$\lvert\mathcal{T}\rvert = 50$	$\lvert\mathcal{T}\rvert = 20$	$\lvert\mathcal{T}\rvert = 5$	$\lvert\mathcal{T}\rvert = 1$
ACED with BC	$C_{max} = 8$	99%	100%	99%	97%	96%
	$C_{max} = 5$	96%	99%	99%	100%	95%
	$C_{max} = 3$	100%	99%	100%	100%	99%
	$C_{max} = 1$	100%	99%	100%	98%	99%
	Average[2]	98.8%	99.3%	99.5%	98.8%	97.3%
ACED without BC	$C_{max} = 5$	100%	95%	100%	93%	97%
BC Policy[3]		60%	54%	24%	2%	4%
Expert Demonstrations		92%	98%	95%	80%	100%

[1] For each set of experiment, we have 10 runs with different random seeds. For each run, we rollout 10 trajectories with the policy at convergence and compute the success rate, hence each entry is computed from a total of 100 rollout trajectories.
[2] The average success rate for $C_{max} = 8$, $C_{max} = 5$, $C_{max} = 3$ and $C_{max} = 1$.
[3] The success rate of the initial policy pre-trained by BC evaluated on 100 rollout trajectories.

6.4 Empirical Evaluation Results

6.4.1 Pick-and-Place Tasks

In this section, we compare the performance of ACED with BC and ACED without BC on pick-and-place tasks in the Fetch environment. In order to study the influence of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}, we test ACED with BC with $|\mathcal{T}| = 100, 50, 20, 5, 1$ and $C_{max} = 8, 5, 3, 1$. ACED without BC is only tested with $|\mathcal{T}| = 100, 50, 20, 5, 1$ and $C_{max} = 5$, since $C_{max} = 5$ is the best performer in ACED with BC experiments in terms of convergence speed. Here we define convergence as having a training success rate of stably above 90% and being able to accomplish most tasks during test time. We compare their convergence performance during training with PPO in Figure 6-4 and Figure 6-5 and their success rate performance during testing in Table 6.1. We also tested ACED with DDPG with $|\mathcal{T}| = 5$ and $C_{max} = 5$, and the average steps to convergence is 5.07 million and the success rate is 100%, proving that ACED can be applied to off-policy RL algorithms and achieve higher sample efficiency.

The horizontal axis in Figure 6-4 and Figure 6-5 represents the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}, and the vertical axis represents the number of environment steps the training takes to converge averaged from 10 runs with different random seeds. From Figure 6-4 we can see that for ACED with BC, $|\mathcal{T}|$ has a significant impact on the number of environment steps it takes to converge. One potential reason that can cause this is, with a more diverse set of initializations during the curriculum training phase, ACED will better generalize to unseen random initial states and goal states after switching to normal rollout worker. Another potential reason is that BC is usually less prone to overfitting when the number of demonstrations is large, hence it should be able to generate better initial policies during pre-training with a larger $|\mathcal{T}|$. In comparison, the number of total curricula has less impact on the convergence performance of ACED with BC, but choosing a reasonable C_{max} can help accelerate training especially when the number of demonstrations is small. For the pick-and-place task we tested on, $C_{max} = 5$ generally performs better across different $|\mathcal{T}|$ values in terms of both the mean and the 90% confidence interval.

If we compare the convergence performance of ACED with BC and ACED without BC in Figure 6-4 and Figure 6-5, we can see that ACED without BC generally takes longer to converge except for when there is only 1 demonstration trajectory. This shows that BC pre-training can provide a good initial policy and accelerate ACED training when sufficient demonstration trajectories are provided. However, when $|\mathcal{T}|$ is too small, BC pre-training might adversely affect ACED's performance. Another observation from Figure 6-5 is that the convergence performance of ACED without BC does not show a clear trend as $|\mathcal{T}|$ decreases, which means that the increasing trend we see in ACED with BC experiments is more likely to have been caused more by BC rather than ACED itself. One explanation for this observation is that, despite ACED's better generalization performance when $|\mathcal{T}|$ is large, the curriculum training phase itself can become more challenging and takes longer to converge with a larger $|\mathcal{T}|$. This is because ACED faces a more diverse set of initializations when there are more demonstration trajectories. Our experiments show that without BC pre-training, ACED actually performs the best with only 1 demonstration in pick-and-place tasks.

Table 6.1 compares the success rate of ACED during test time with the success rate of expert demonstrations and behavior cloning. We can see that even though the expert demonstrations aren't perfect (i.e. mostly have a success rate of less than 100%), ACED is able to learn the pick-and-place task with better-than-expert performance. This is because our approach doesn't rely on the expert policy except for the BC pre-training, and it instead tries to come up with its own policy that reaches the goal from states along the demonstration trajectories. Therefore, even with sub-optimal demonstrations, ACED can still achieve better-than-expert performance. On the other hand, with policies trained only by BC, the success rate is much lower especially when the number of demonstrations is small, proving that our approach utilizes expert demonstrations in a much more effective way than BC does. Another observation from Table 6.1 is that ACED generally achieves higher success rate with more demonstration trajectories, but it is notable that even with only 1 demonstration trajectory, it can still achieve a success rate of 96%. This is very encouraging because unlike many other imitation learning approaches that require a large number of demonstrations in order to work effectively, ACED can succeed with as few as 1 demonstration.

6.4.2 Block Stacking Tasks

ACED with BC is also evaluated in block stacking tasks with $|\mathcal{T}| = 100, 20$ and $C_{max} = 12, 8$, and its performance is compared with other state-of-the-art automatic curriculum methods including reverse curriculum [45] and the Montezuma's Revenge method [118]. Unfortunately, neither of the baseline approaches are able to successfully learn the stacking task (i.e. converge), hence we cannot compare with their convergence performance. We provide more detailed discussion on the comparison of their learning progress performance in Section 6.4.3. Table 6.2 presents the average number of environment steps ACED with BC takes to converge in each set of experiments, and compares its success rate with the initial policies pre-trained by

Table 6.2: Block Stacking Performance [30]

Number of Demonstrations		$\|\mathcal{T}\| = 100$		$\|\mathcal{T}\| = 20$	
ACED with BC[1]	Total Curriculum Number	$C_{max} = 12$	$C_{max} = 8$	$C_{max} = 12$	$C_{max} = 8$
	Convergence Env Steps[2]	213.08	169.88	143.02	119.79
	Success Rate[3]	100%	100%	96%	100%
BC Policy Success Rate[4]		0%		0%	
Expert Demonstration Success Rate[5]		84%		85%	

[1] The performance for each set of experiment for ACED with BC is averaged from 5 runs with different random seeds.

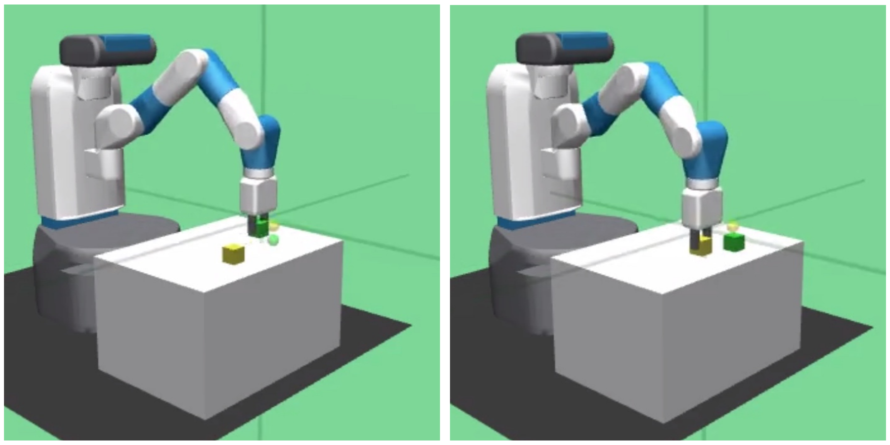
[2] Presented in millions. The training for stacking tasks is more unstable, so we separated the training process into two sections: 1) use curriculum rollout workers to train until all parallel workers reach $C = C_{max}$ and 2) set all parallel workers to be normal rollout workers and train until convergence. The convergence environment steps presented here are the sum of the two sections.

[3] The success rate for ACED with BC is evaluated with 10 rollout trajectories per random seed (50 rollout trajectories in total).

[4] The success rate of the initial policy pre-trained by BC evaluated on 100 rollout trajectories.

[5] The demonstrations for stacking have much lower success rates because there are two blocks in the scene and one block might obstruct the straight line policy the moves the other block to its goal pose. Since the straight line policies are open-loop, the agent can't recover from such failures.

BC and with the expert demonstrations. Interestingly, we observe that the number of environment steps until convergence is much lower when $|\mathcal{T}| = 20$ compared to when $|\mathcal{T}| = 100$. We believe this is because ACED with BC has converged to two different policies for the two sets of experiments with different $|\mathcal{T}|$ values. From the recorded videos we found that in all 10 runs with $|\mathcal{T}| = 100$ (including $C_{max} = 12$ and $C_{max} = 8$), the policies ACED with BC converged to are similar to the demonstrations, where the robot first picks up the green block and places it onto the goal, and then picks up the yellow block and places it onto the green block. However, in all 10 runs with $|\mathcal{T}| = 20$, the policy ACED with BC converged to takes a different route: it first places the yellow block on top of the green block, and then picks up the two blocks together to place them onto the goal. Figure 6-6 illustrates the two different policies visually by showing two representative frames from each rollout video. This finding shows that, with a smaller number of demonstrations, ACED with BC has more flexibility to come up with novel solutions instead of following the demonstra-


(a) Solution policy when $|\mathcal{T}| = 100$

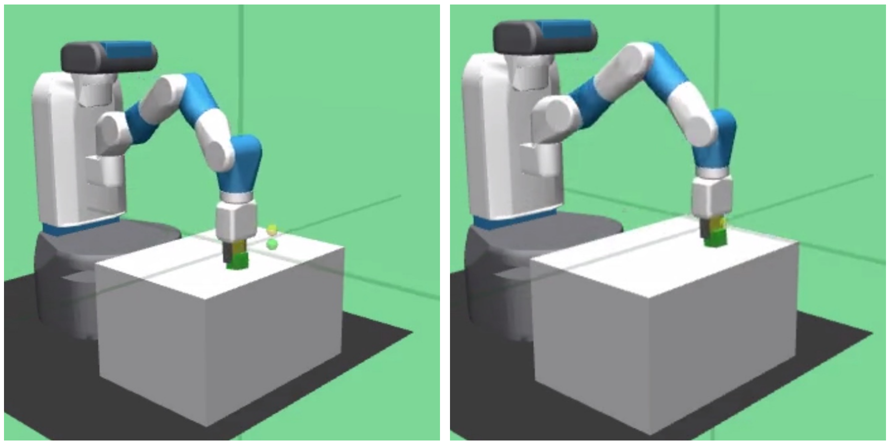

(b) Solution policy when $|\mathcal{T}| = 20$

Figure 6-6: Visualization of two different stacking policies ACED with BC converged to with different $|\mathcal{T}|$ values. [30]

tion trajectories. Because the $|\mathcal{T}| = 20$ solution takes fewer steps, it is easier to train and is more robust when generalizing to new initial and goal poses. We also observed that the $|\mathcal{T}| = 20$ training curves experience less performance drop when switching from curriculum rollout workers to normal rollout workers. We believe this is why ACED with BC converges faster when trained with 20 demonstrations.

Another finding from Table 6.2 is that for the block stacking task we tested on, $C_{max} = 8$ has better performance than $C_{max} = 12$ in terms of both convergence speed and success rate. Compared to the number of demonstrations, the number of total curricula has less impact on ACED with BC's solutions and training performance, and different C_{max} numbers didn't cause the solution policies to differ qualitatively.

6.4.3 Learning Curves

In order to show the learning progress and curriculum switches when using ACED, we use the pick-and-place task with 5 demonstration trajectories as an example to compare the learning curves of different algorithms, as shown in Figure 6-7. Since each experiment is terminated after convergence, the lengths of the learning curves may vary. For each algorithm, we select one run whose convergence environment step is close to the mean for all 10 runs instead of directly using the mean in order to clearly show the learning progress and the curriculum switches. From Figure 6-7 we can see that for all ACED runs with PPO, the first few curricula are usually much easier than the last few and the majority of training time is spent on training the last few curricula. Without BC, the performance drop during curriculum switches is more obvious. If we compare the performance of ACED with DDPG and ACED with PPO, we can observe that ACED achieves a much higher sample efficiency with the off-policy DDPG.

All ACED runs are able to converge to almost 100% success rate, whereas vanilla PPO without ACED is not able to achieve a success rate higher than 10% during training. In addition to the comparison with vanilla PPO, we also compare ACED with Hindsight Experience Replay (HER) [4] in the block pick-and-place task. We use the OpenAI Baseline [38] implementation of HER with 2 MPI processes with 30 parallel environments each to make sure it is equivalent to the 60 parallel environments in other experiments. Other parameters for HER are set to default. However, all 10 runs with HER are only able to achieve a success rate of about 50%, and we show one representative learning curve in Figure 6-7. This is because in the Gym FetchPickAndPlace-V1 task, half of the goals are sampled from on the table and half are sampled in the air, thus agents that only learned to push can still reach the goals close to the tabletop and receive a success rate of about 50%, but only agents that actually learned to pick and place will reach a success rate of 100%.

We show the comparison of ACED with two state-of-the-art automatic curriculum generation methods [45, 118] in the block stacking task in Figure 6-8. We refer to

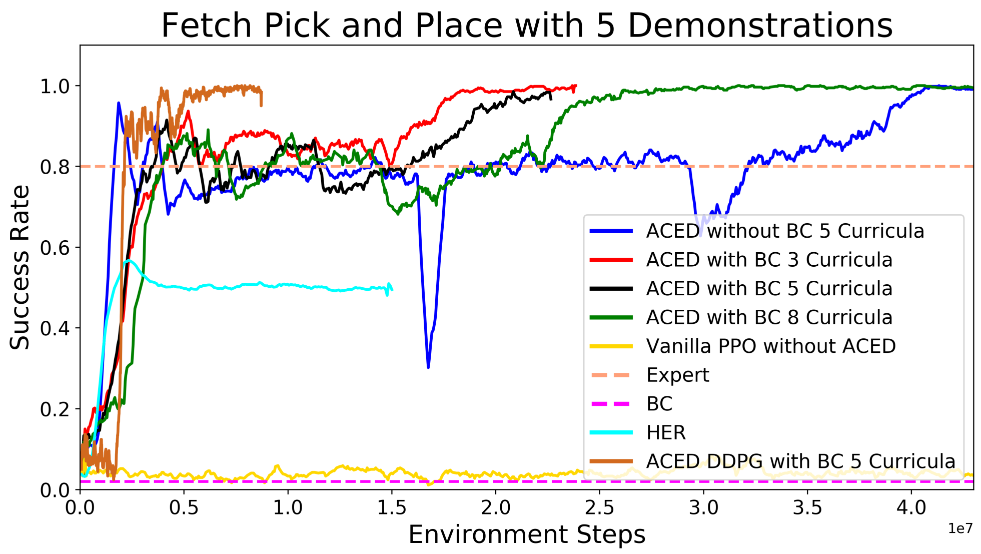


Figure 6-7: Learning curves of different algorithms in the pick-and-place environment with 5 demonstration trajectories. The horizontal axis represents the number of environment steps during training and the vertical axis represents the success rate. Expert and BC success rates are represented by dash lines because they didn't have training processes and their success rates remain constant.

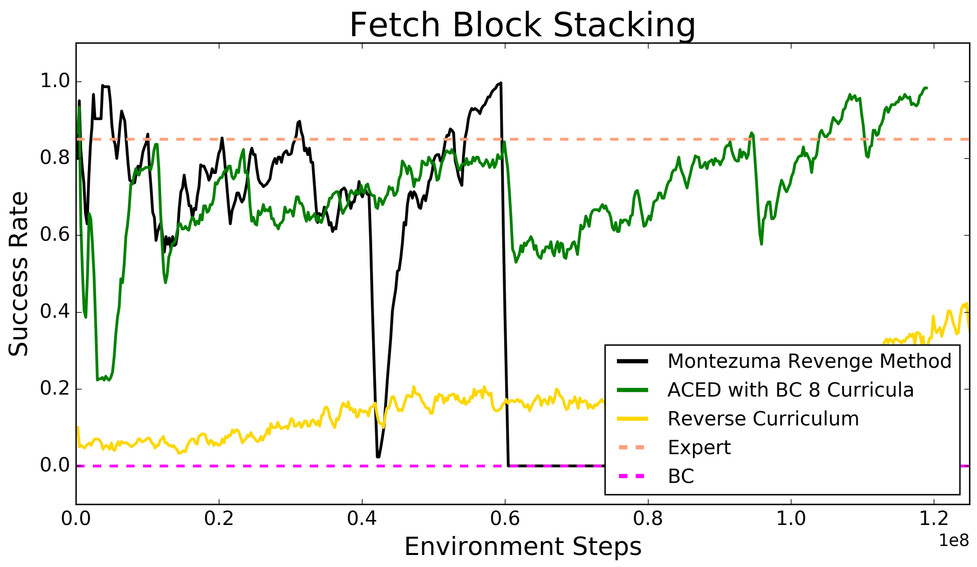


Figure 6-8: Learning curves of ACED with BC, the reverse curriculum method [45], and the Montezuma's Revenge method [118] in the block stacking task.

the approach presented in [45] the reverse curriculum method, and the one presented in [118] the Montezuma's Revenge method. In Figure 6-8, ACED is implemented with PPO and is provided with 20 demonstrations and BC pre-training. As we mentioned earlier, neither of the two baseline automatic curriculum methods are able to converge in the block stacking tasks. In all runs of the reverse curriculum method, the start state distribution has not moved to the true S_0 after 400 million environment steps of training, and achieves 0% success rate during testing. In all runs of the Montezuma's Revenge method, moving through the curriculum rollout workers are relatively quick, but the learning curve drops to zero and never goes back up once it switches to the normal rollout worker. This shows that in challenging tasks in continuous state space, policies training using a fixed demonstration without randomization struggle to generalize to the entire initial state distribution S_0 and goal distribution S_g. Figure 6-8 shows the learning curves of one representative run for each of the curriculum generation methods.

6.5 Comparison with BC + RL without ACED

In order to further demonstrate the role of curriculum learning in ACED, this section compares the performance of ACED with an algorithm that only uses BC pre-trained policies to initialize the RL agent but doesn't use curriculum learning during RL training. We referred to the RL algorithm that uses BC to pre-train the policy but doesn't use ACED as "BC + RL". The same PPO algorithm and BC pre-trained policy initializations as in the ACED experiments are used in all experiments presented in this section. We summarizes the performance of both ACED with BC and BC + RL in Table 6.3 for convenient comparison, but the ACED with BC data in Table 6.3 are the same as the ones presented in Figure 6-4 and Table 6.1. As shown in Table 6.3, BC + RL only works better than ACED with BC when $|\mathcal{T}| = 100$, whereas its performance in terms of both convergence speed and success rate is worse than that of ACED with BC when $|\mathcal{T}| = 50$ and $|\mathcal{T}| = 20$. When $|\mathcal{T}| = 5$ or $|\mathcal{T}| = 1$, BC + RL is not able to learn pick-and-place and none of the runs converged to a success

Table 6.3: Pick-and-Place Comparison with BC + RL

Algorithm[1]			$\|\mathcal{T}\| = 100$	$\|\mathcal{T}\| = 50$	$\|\mathcal{T}\| = 20$	$\|\mathcal{T}\| = 5^2$	$\|\mathcal{T}\| = 1^2$
ACED with BC	Convergence Env Steps (Million)	$C_{max} = 8$	2.78	3.40	8.40	24.00	41.21
		$C_{max} = 5$	2.32	3.55	5.97	16.50	38.85
		$C_{max} = 3$	4.15	6.53	7.88	17.67	25.56
		Average[3]	3.08	4.49	7.41	19.39	35.21
	Success Rate	$C_{max} = 8$	99%	100%	99%	97%	96%
		$C_{max} = 5$	96%	99%	99%	100%	95%
		$C_{max} = 3$	100%	99%	100%	100%	99%
		Average[3]	98.3%	99.3%	99.3%	99%	96.7%
BC + RL	Convergence Env Steps		2.47	14.66	13.58	(67.67)	(61.73)
	Success Rate		100%	97%	99%	(37%)	(53%)

[1] For each set of experiment except for BC + RL with $|\mathcal{T}| = 5$ and $|\mathcal{T}| = 1$, we have 10 runs with different random seeds and the entries in the table are averaged from all runs. For BC + RL with $|\mathcal{T}| = 5$ and $|\mathcal{T}| = 1$, we only conducted 3 runs each due to their long training time. For each run, we rollout 10 trajectories with the policy at convergence, and we compute the success rate by taking the average of all rollout trajectories for all runs.

[2] The entries for BC + RL with $|\mathcal{T}| = 5$ and $|\mathcal{T}| = 1$ are in brackets because none of these experiments have actually converged to a success rate of 100% during training. They instead converged to around 50% because they have only learned to push the block to the goal when the goal pose is on the tabletop, but they failed to learn how to pick up the block and lift them to the goal poses that are in the air.

[3] The average success rate for $C_{max} = 8$, $C_{max} = 5$ and $C_{max} = 3$.

rate of 100%. Recorded videos show that when $|\mathcal{T}| = 5$ and $|\mathcal{T}| = 1$, BC + RL can only learn to push the block to goal poses that are on the tabletop, but failed to learn pick-and-place when the goal pose is in the air. Since the goal pose has a 50% probability of being in the air in the pick-and-place environment, all the runs have converged to a success rate of around 50% during training.

We also evaluated BC + RL in the block stacking environment, but results show that none of the runs with $|\mathcal{T}| = 100$ or $|\mathcal{T}| = 20$ can converge to a success rate of 100%. In fact, the training curves remain zero throughout the entire training progress for all runs with BC + RL. The comparison between ACED with BC and

BC + RL shows that ACED is especially helpful in scenarios where the target task is complicated or the number of demonstrations is small.

6.6 Combination with Empowerment-based Intrinsic Motivation

In Chapter 5, we proposed an empowerment-based intrinsic motivation approach that can effectively augment the sparse extrinsic task rewards during learning and alleviate the exploration challenges faced by RL agents in robotic manipulation tasks. In this section, we investigate if empowerment-based intrinsic motivation is able to further improve the RL agent's performance when expert demonstrations are provided. Specifically, we evaluate the combination of ACED with BC and empowerment-based intrinsic motivation in block pick-and-place tasks and compare its performance with vanilla ACED.

In order to combine ACED with empowerment-based intrinsic motivation, we preserve all the BC pre-training procedures and the curriculum switching mechanisms in ACED while augmenting the RL reward with intrinsic rewards shown in Figure 5-4. The implementation of empowerment rewards and ICM rewards is the same as Section 5.3.2, with the weight coefficients:

$$\begin{aligned} w_t^{ICM} &= 0.5 \times (1 - \tanh(200(r_t^{ICM} - 0.08))), \\ w_t^{Emp} &= 1 - w_t^{ICM}. \end{aligned} \tag{6.1}$$

We present the convergence speed performance of this combination in Figure 6-9, and the success rate performance in Table 6.4. For easy comparison with Figure 6-4, we also combined some of the experiment results for ACED with empowerment and ACED without empowerment in the same bar chart in Figure 6-10. Compared to the success rate performance shown in Table 6.1, the combination of empowerment-based intrinsic rewards and ACED shows slightly higher success rate in Table 6.4. From the comparison between Figure 6-9 and Figure 6-4 we can see that the general

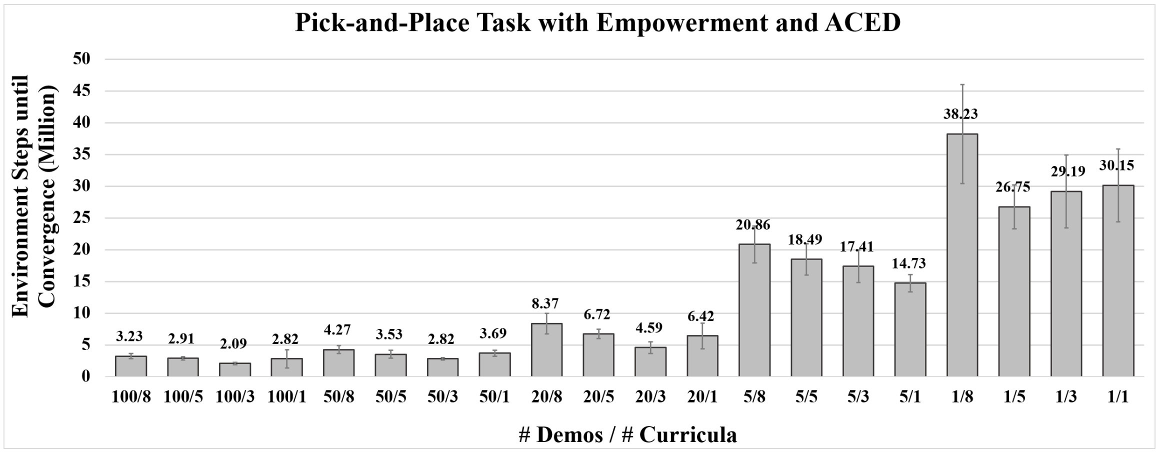


Figure 6-9: Number of environment steps the combination of ACED with BC and empowerment-based intrinsic motivation takes to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval.

Table 6.4: ACED with Empowerment Success Rate in Pick-and-Place Tasks

Algorithm	Number of Curricula[1]	Number of Demonstrations				
		$\|\mathcal{T}\| =$ 100	$\|\mathcal{T}\| =$ 50	$\|\mathcal{T}\| =$ 20	$\|\mathcal{T}\| = 5$	$\|\mathcal{T}\| = 1$
ACED with BC and Empowerment	$C_{max} = 8$	99%	99%	99%	99%	97%
	$C_{max} = 5$	100%	100%	99%	100%	99%
	$C_{max} = 3$	100%	100%	100%	99%	98%
	$C_{max} = 1$	99%	100%	100%	100%	98%
	Average[2]	99.5%	99.75%	99.5%	99.5%	98%

[1] For each set of experiment, we have 10 runs with different random seeds. For each run, we rollout 10 trajectories with the policy at convergence and compute the success rate, hence each entry is computed from a total of 100 rollout trajectories.

[2] The average success rate for $C_{max} = 8$, $C_{max} = 5$, $C_{max} = 3$ and $C_{max} = 1$.

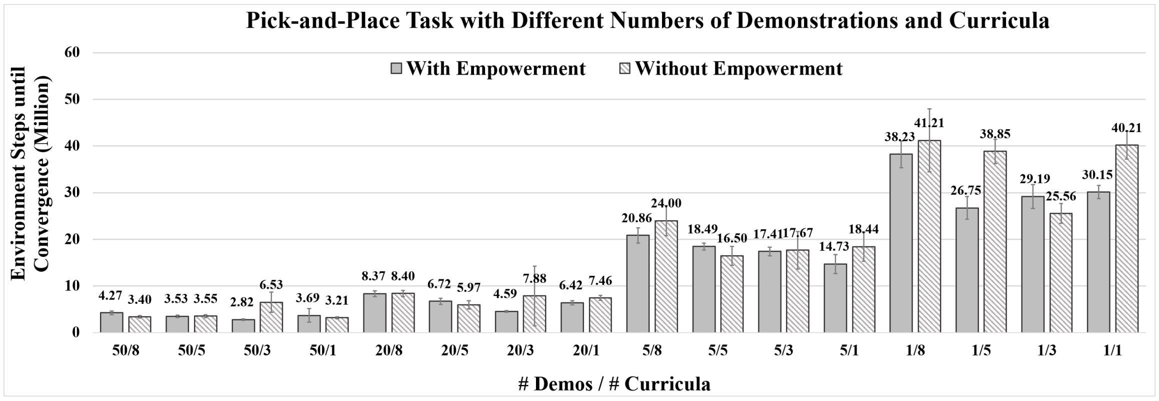


Figure 6-10: Number of environment steps ACED with empowerment and ACED without empowerment take to train pick-and-place tasks with PPO until convergence with different values of the number of demonstration trajectories $|\mathcal{T}|$ and the total number of curricula C_{max}. The bars represent the mean of 10 runs with different random seeds and the error bars represent the 90% confidence interval.

trends of ACED with empowerment and vanilla ACED are similar as the number of demonstrations increases. The improvement in terms of convergence speed caused by the augmented rewards is much more significant when the number of demonstration trajectories $|\mathcal{T}|$ and the number of curricula C_{max} are low. When $C_{max} = 8$, ACED with empowerment actually shows a higher number of environment steps until convergence. These findings align with our expectations because empowerment-based intrinsic motivation is designed for tasks with sparse extrinsic rewards. When the number of demonstration trajectories and the number of curricula are low, the RL agents are learning tasks where extrinsic rewards are more difficult to obtain at the beginning of training, hence augmenting the task rewards with empowerment-based intrinsic rewards will help accelerate the learning progress. However, when the number of demonstration trajectories and the number of curricula are high, the RL agents are initialized with policies that are closer to the task solutions, but are presented with tasks with much shorter horizons. This means that it is relatively easy for the agents to obtain extrinsic tasks rewards, hence augmenting them with intrinsic rewards will actually distract the agent and slow down the learning progress.

6.7 Discussion

In this chapter, we presented ACED, an imitation learning method that leverages expert demonstrations to extract a set of curricula automatically in order to accelerate learning. We evaluated the influence of the number of demonstrations and the number of curricula on the performance of the ACED algorithm in a pick-and-place task and a block stacking task. Experiment results show that ACED can achieve more than 97% success rate on pick-and-place tasks while learning with only one demonstrations, whereas other imitation learning methods cannot effectively learn when the number of demonstrations is so low. We also observed qualitatively different policies while learning the block stacking task with different numbers of demonstrations, showing that ACED is not simply imitating the expert strategies and can innovate on more effective policies on its own. The comparison with other automatic curriculum learning methods shows that ACED can learn challenging tasks that other state-of-the-art approaches can't learn. Additionally, we combined the empowerment-based intrinsic motivation method proposed in Chapter 5 with ACED and demonstrated in empirical experiments that the combination is able to achieve higher success rate and faster convergence compared to ACED alone. One major advantage of ACED is that it is a very intuitive idea that can be easily combined with most existing learning from demonstration methods, including adding demonstrations to replay buffers [85] and introducing GAN-based imitation rewards [57], in order to learn more complex tasks with higher sample efficiency for expert demonstrations, for example stacking multiple blocks into block towers or tasks involving tool using.

Chapter 7

Conclusion

This chapter first reviews the main content of this thesis, then summarizes the main contributions and discusses the potential directions for future work.

7.1 Summary

This thesis considered the challenges faced by home support robots when solving manipulation tasks. Specifically, we focus on two different scenarios: accomplishing known tasks quickly and safely, and learning new tasks with a minimal amount of human supervision. Both scenarios are often encountered when executing daily tasks while supporting human customers at home. For example, the robot might need to fetch the same set of items from a kitchen cupboard everyday, hence knowing how to complete pick-and-place task and only needing to call a motion planning algorithm will be very helpful in this case. On the other hand, the robot might need to learn some new skills when meeting a new customer, for example, cleaning up the living room or cooking a new dish. In these cases, it is very important that the robot is capable of learning new tasks quickly and requiring a minimal amount of customer supervision.

In this thesis, a fast-reactive chance-constrained motion planning approach was proposed in Chapter 4 to tackle challenges in the first scenario, and two Reinforcement Learning (RL) based approaches were proposed in Chapter 5 and Chapter 6 to facilitate success in the second scenario. Leveraging a combined sampling-based and optimization-based deterministic motion planning framework as well as supervised-learning techniques for collision risk estimation, our motion planning approach, learning-based p-Chekov, is able to satisfy chance constraints while maintaining high planning speed in known environments. On the other hand, the two RL-based approaches this thesis proposed focus on learning from scratch without human supervision and fast learning with a handful of human demonstrations respectively. In Chapter 5, we introduced empowerment-based intrinsic motivation, a form of intrinsic motivation which encourages RL agents to explore the sections of the state space that might contain high rewards when learning robotic manipulation tasks from scratch. In Chapter 6, we proposed Automatic Curricula via Expert Demonstrations, an imitation learning algorithm that efficiently utilizes human demonstrations through constructing a sequence of curricula.

7.2 Main Contributions

In this section, we summarizes the three main contributions of this thesis:

1. Learning-based P-Chekov
2. Empowerment-based Intrinsic Motivation
3. Automatic Curricula via Expert Demonstrations (ACED)

7.2.1 Learning-based P-Chekov

Learning-based p-Chekov is a fast-reactive chance-constrained motion planning approach that can provide safe motion plans for robotic manipulators in known environments. It includes a planning phase that generates initial feasible solutions for the robotic manipulator to execute, and an execution phase where chance-constraints are reallocated in order to further optimize the planning phase solutions. Learning-based p-Chekov propagates process noises and observation noises along the nominal trajectory generated by a deterministic motion planner in order to estimate the *a priori* probability distribution of robot states, and decomposes the joint chance constraint into allowed collision risk bounds at discrete waypoints. The risk estimation component then predicts the collision risk during execution based on the estimated state distributions, and then compares them with the allocated risk bounds to extract "conflicts" where the risk bounds are violated. These conflicts are fed back to deterministic Chekov to guide it to generated safer nominal trajectories. After resolving all the conflicts, the solution trajectory is passed into execution phase and an Iterative Risk Allocation (IRA) component will improve the solution through reallocating risk bounds.

Built on top of a combined sampling-based and optimization-based motion planning framework Chekov, it inherits Chekov's advantage in reacting fast to plan requests and its ability of generating smooth motion plans. By applying supervised-learning techniques to collision risk estimation, learning-based p-Chekov moves significant amount of the computation to off-line and improves upon the previously proposed sampling-based p-Chekov which heavily relies on on-line Monte Carlo sampling and collision risk estimation and requires a large amount of computation even with a small number of samples per time step. We compare the performance of a variety of supervised-learning models and conclude that neural networks are the best performers in terms of both training performance and testing performance. We demonstrate in empirical experiments in realistic application scenarios that learning-based p-Chekov with neural networks is able to provide smooth motion plans that satisfy pre-specified chance-constraints while significantly accelerating the planning speed. The comparison with sampling-based p-Chekov shows that learning-based p-Chekov can reduce planning time by 50% - 70% while maintaining similar chance constraint satisfaction rate.

7.2.2 Empowerment-based Intrinsic Motivation

How to efficiently explore the state space in order to learn target tasks with high-dimensional continuous state and action spaces has been a key challenge in RL.

Inspired by how human babies are intrinsically curious about the surrounding environment, many intrinsic motivation approaches have been proposed to facilitate the learning process of RL agents. The empowerment-based intrinsic motivation approach proposed in this thesis leverages recent advances in both mutual information maximization and intrinsic novelty-driven exploration in order to guide RL agents explore high-reward regions of the state space when learning robotic manipulation tasks. Through maximizing the mutual dependence between robot actions and environment states, namely the empowerment, this intrinsic motivation helps the agent to focus more on the states where it can effectively "control" the environment instead of the parts where its actions cause random and unpredictable consequences. Despite the challenges posed by conditional mutual information maximization with continuous high-dimensional random variables, we are able to successfully train neural networks that make reasonable predictions on empowerment with the help of novelty-driven exploration methods at the beginning of the learning process.

Empirical evaluations in different robotic manipulation environments with different shapes of the target object demonstrate the advantages of this empowerment-based intrinsic motivation over other state-of-the-art solutions to sparse-reward RL tasks. In addition, we also combine this approach with diversity-driven intrinsic motivation and show that the combination is able to encourage the manipulator to learn a diverse set of ways to interact with the object, whereas with the diversity-driven rewards alone the manipulator is only able to learn how to move itself in different directions. This approach can be easily integrated into any RL algorithm to accelerate their learning progress, or be combined with approaches like Hindsight Experience Replay (HER) and imitation learning to further improve their performance.

7.2.3 Automatic Curricula via Expert Demonstrations

A key challenge that constrains imitation learning's wide application in robotics tasks is its high demand on human demonstrations. In order to tackle this issue, this thesis proposes ACED, an RL approach that combines ideas from both imitation learning and curriculum learning in order to solve challenging robotics manipulation tasks with sparse reward signals. Through resetting the training episodes to states along demonstration trajectories, ACED is able to control the difficulty of the tasks by moving the reset states from the end of the demonstration to the beginning based on the learning progress of the RL agent. This procedure naturally forms a curriculum and makes challenging exploration problems feasible to learn. One main advantage of ACED is that it only requires demonstration states and not actions when deployed on its own. ACED can also be intuitively combined with many existing imitation learning approaches to utilize expert demonstrations more efficiently, including adding demonstrations to replay buffers [85], introducing GAN-based rewards [57], and using behavior cloning to pre-train the policies. In Chapter 6, a version of ACED with policies pre-trained via behavior cloning is compared with ACED on its own as an example. We evaluate the performance of ACED on block pick-and-place tasks and stacking tasks, and show that pick-and-place can be learned with as few as 1 demonstration and stacking can be learned with 20 demonstrations. We also analyzed the impact of the number of demonstration trajectories and the total number of curricula on ACED's performance, and discovered that ACED can learn novel solutions that are very distinct from expert demonstrations when the number of demonstrations is small.

7.3 Discussion and Future Work

Despite the contributions made in this thesis, many challenges remain to be solved in order for home support robots to successfully accomplish household tasks requested by potential customers. In the first scenario described in Section 7.1, in order to turn common tasks to motion planning problems, a scene understanding approach is also needed to convert visual observations into simulation environments for the motion planning as well as to locate the target object's pose within the global environment. Many existing scene understanding and pose estimation approaches, such as [74] and [113], are good candidates for this application. In the second scenario described in Section 7.1, in addition to a scene understanding approach, sim-to-real transfer algorithms, e.g. [133] and [106], are also necessary for the robot to transfer skills it learned in simulation to real-world skills.

In future work, the learning-based p-Chekov algorithm can be further improved by incorporating minor changes to environment objects' poses during off-line risk estimator training. This can be achieved by assigning movable objects and training conditional collision risk estimators that are conditioned on the actual poses of the movable objects in the scene. Additionally, the skills discovered using empowerment-based intrinsic motivation and unsupervised skill discovery approaches can serve as primitive actions in a HRL framework in order to simplify the learning of long-horizon complex tasks. ACED can also be extended to utilize semantically meaningful segmentations of expert demonstrations instead of evenly divided sections in order to better accommodate the RL tasks that are best solved through subtasks with highly imbalanced lengths of action sequences. Furthermore, since the number of demonstrations needed for ACED to solve a task highly depends on the complexity of the task, incorporating an interactive framework where the robot can query for more demonstrations when the tasks are more challenging to learn is another future work direction towards user-friendly home support robots.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1964.

[3] Ron Alterovitz, Thierry Siméon, and Kenneth Y Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. In *Robotics: Science and systems*, volume 3, pages 233–241, 2007.

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[6] Oktay Arslan and Panagiotis Tsiotras. Machine learning guided exploration for sampling-based motion planning algorithms. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2646–2652. IEEE, 2015.

[7] Anna Atramentov and Steven M LaValle. Efficient nearest neighbor searching for motion planning. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 1, pages 632–637. IEEE, 2002.

[8] Brian Axelrod, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Provably safe robot navigation with obstacle uncertainty. *The International Journal of Robotics Research*, 37(13-14):1760–1774, 2018.

[9] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[10] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[11] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[12] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Devon Hjelm, and Aaron Courville. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 530–539, 2018.

[13] Richard Ernest Bellman. Dynamic programming. 1957.

[14] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[15] Lars Blackmore, Hui Li, and Brian Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference, 2006*, pages 7–pp. IEEE, 2006.

[16] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE transactions on Robotics*, 26(3):502–517, 2010.

[17] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[18] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.

[19] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019.

[20] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[21] Julien Burlet, Olivier Aycard, and Thierry Fraichard. Robust motion planning using markov decision processes and quadtree decomposition. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2820–2825. IEEE, 2004.

[22] Chao Chen, Markus Rickert, and Alois Knoll. Motion planning under perception and control uncertainties with space exploration guided heuristic search. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 712–718. IEEE, 2017.

[23] Rohan Chitnis, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning quickly to plan quickly using modular meta-learning. In *2019 IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pages 7865–7871. IEEE, 2019.

[24] François Chollet et al. Keras. https://keras.io, 2015.

[25] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[26] Cédric Colas, Pierre-Yves Oudeyer, Olivier Sigaud, Pierre Fournier, and Mohamed Chetouani. Curious: Intrinsically motivated modular multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 1331–1340, 2019.

[27] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[28] Imre Csiszár, Paul C Shields, et al. Information theory and statistics: A tutorial. *Foundations and Trends® in Communications and Information Theory*, 1(4):417–528, 2004.

[29] Siyu Dai. Probabilistic motion planning and optimization incorporating chance constraints. Master's thesis, Massachusetts Institute of Technology, 2018.

[30] Siyu Dai, Andreas Hofmann, and Brian Williams. Automatic curricula via expert demonstrations. *arXiv preprint arXiv:2106.09159*, 2021.

[31] Siyu Dai, Andreas Hofmann, and Brian Williams. Fast-reactive probabilistic motion planning for high-dimensional robots. *SN Computer Science*, 2(6):1–39, 2021, Reproduced with permission from Springer Nature.

[32] Siyu Dai, Matthew Orton, Shawn Schaffert, Andreas Hofmann, and Brian C Williams. Improving trajectory optimization using a roadmap framework. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[33] Siyu Dai, Shawn Schaffert, Ashkan Jasour, Andreas Hofmann, and Brian C Williams. Chance constrained motion planning for high-dimensional robots. In *Proceedings of the 2019 IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2019.

[34] Siyu Dai, Wei Xu, Andreas Hofmann, and Brian C. Williams. An Empowerment-based Solution to Robotic Manipulation Tasks with Sparse Rewards. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[35] Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *The Journal of Machine Learning Research*, 17(1):3190–3239, 2016.

[36] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941, 2017.

[37] Charles Dawson, Andreas Hofmann, and Brian Williams. Fast certification of collision probability bounds with uncertain convex obstacles. *arXiv preprint arXiv:2003.07792*, 2020.

[38] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

[39] Morris L Eaton. Multivariate statistics: a vector space approach. *JOHN WILEY & SONS, INC., 605 THIRD AVE., NEW YORK, NY 10158, USA, 1983, 512*, 1983.

[40] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. In *International Conference on Learning Representations*, 2018.

[41] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

[43] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[44] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[45] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pages 482–495, 2017.

[46] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[47] Alexandre Galashov, Siddhant Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojtek M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations*, 2019.

[48] Arthur Gelb. *Applied optimal estimation.* MIT press, 1974.

[49] IM Gel'Fand and AM Yaglom. Calculation of amount of information about a random function contained in another such function. *Eleven Papers on Analysis, Probability and Topology*, 12:199, 1959.

[50] Jacob Goldberger and Yaniv Opochinsky. Information-bottleneck based on the jensen-shannon divergence with applications to pairwise clustering. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3507–3511. IEEE, 2019.

[51] Anirudh Goyal, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Sergey Levine, and Yoshua Bengio. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*, 2019.

[52] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320. JMLR. org, 2017.

[53] Jung-Su Ha, Hyeok-Joo Chae, and Han-Lim Choi. Approximate inference-based motion planning by learning and exploiting low-dimensional latent variable models. *IEEE Robotics and Automation Letters*, 3(4):3892–3899, 2018.

[54] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1846–1855, 2018.

[55] Francis Begnaud Hildebrand. *Introduction to numerical analysis.* Courier Corporation, 1987.

[56] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[57] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4572–4580, 2016.

[58] Wassily Hoeffding, Herbert Robbins, et al. The central limit theorem for dependent random variables. *Duke Mathematical Journal*, 15(3):773–780, 1948.

[59] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[60] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094. IEEE, 2018.

[61] Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 15–21. IEEE, 2019.

[62] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

[63] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2469–2478. PMLR, 2018.

[64] Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. Emi: Exploration with mutual information. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3360–3369, 2019.

[65] Youngjin Kim, Wontae Nam, Hyunwoo Kim, Ji-Hoon Kim, and Gunhee Kim. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In *International Conference on Machine Learning*, pages 3379–3388, 2019.

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

[67] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135.

[68] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[69] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland., 2008.

[70] John B Lanier, Stephen McAleer, and Pierre Baldi. Curiosity-driven multi-criteria hindsight experience replay. *arXiv preprint arXiv:1906.03710*, 2019.

[71] Alex Lee, Yan Duan, Sachin Patil, John Schulman, Zoe McCarthy, Jur Van Den Berg, Ken Goldberg, and Pieter Abbeel. Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5660–5667. IEEE, 2013.

[72] David Lenz, Markus Rickert, and Alois Knoll. Heuristic search in belief space for motion planning under uncertainties. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2659–2665. IEEE, 2015.

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

[74] Li-Jia Li, Richard Socher, and Li Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2036–2043. IEEE, 2009.

[75] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[76] Friedrich Liese and Igor Vajda. On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory*, 52(10):4394–4412, 2006.

[77] 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.

[78] Wei Liu and Marcelo H Ang. Incremental sampling-based algorithm for risk-aware planning under motion uncertainty. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2051–2058. IEEE, 2014.

[79] Brandon Luders, Mangal Kothari, and Jonathan How. Chance constrained rrt for probabilistic robustness to environmental uncertainty. In *AIAA guidance, navigation, and control conference*, page 8160, 2010.

[80] Brandon D Luders, Sertac Karaman, and Jonathan P How. Robust sampling-based motion planning with asymptotic optimality guarantees. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 5097, 2013.

[81] Yuanfu Luo, Haoyu Bai, David Hsu, and Wee Sun Lee. Importance sampling for online planning under uncertainty. *The International Journal of Robotics Research*, 38(2-3):162–181, 2019.

[82] Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Andrew Zisserman, Raia Hadsell, et al. Learning to navigate in cities without a map. In *Advances in Neural Information Processing Systems*, pages 2419–2430, 2018.

[83] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

[84] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[85] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[86] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[87] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[88] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

[89] Masahiro Ono and Brian Williams. An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure. 2008.

[90] Masahiro Ono and Brian C Williams. Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3427–3432. IEEE, 2008.

[91] Masahiro Ono, Brian C Williams, and Lars Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research*, 46:511–577, 2013.

[92] Takayuki Osa, Voot Tangkaratt, and Masashi Sugiyama. Hierarchical reinforcement learning via advantage-weighted information maximization. In *International Conference on Learning Representations*, 2019.

[93] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[94] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[95] Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

[96] AB Owen. Monte carlo theory, methods and examples (book draft), 2014.

[97] Jia Pan, Sachin Chitta, and Dinesh Manocha. Probabilistic collision detection between noisy point clouds using robust classification. In *Robotics Research*, pages 77–94. Springer, 2017.

[98] Jia Pan and Dinesh Manocha. Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. *The International Journal of Robotics Research*, 35(12):1477–1496, 2016.

[99] Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[100] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787, 2017.

[101] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*, 2019.

[102] Sachin Patil, Yan Duan, John Schulman, Ken Goldberg, and Pieter Abbeel. Gaussian belief space planning with discontinuities in sensing domains. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6483–6490. IEEE, 2014.

[103] Sachin Patil, Gregory Kahn, Michael Laskey, John Schulman, Ken Goldberg, and Pieter Abbeel. Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In *Algorithmic foundations of robotics XI*, pages 515–533. Springer, 2015.

[104] Sachin Patil, Jur Van Den Berg, and Ron Alterovitz. Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3238–3244. IEEE, 2012.

[105] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[106] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[107] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. In *International Conference on Learning Representations*, 2018.

[108] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1527–1533. IEEE, 2017.

[109] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[110] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[111] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[112] CE Rasmussen and C Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[113] Xinyi Ren, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Abhishek Gupta, Aviv Tamar, and Pieter Abbeel. Domain randomization for active pose estimation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7228–7234. IEEE, 2019.

[114] Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alexander Peysakhovich, Kyunghyun Cho, and Joan Bruna. Backplay:" man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

[115] RethinkRobotics. Baxter http://www.rethinkrobotics.com/baxter/, 2012.

[116] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pages 4341–4350, 2018.

[117] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings*

of the fourteenth international conference on artificial intelligence and statistics, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[118] Tim Salimans and Richard Chen. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

[119] Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphael Marinier, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2019.

[120] Elad Schneidman, William Bialek, and Michael J Berry. Synergy, redundancy, and independence in population codes. *Journal of Neuroscience*, 23(37):11539–11553, 2003.

[121] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Citeseer, 2013.

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

[123] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[124] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised skill discovery. In *International Conference on Learning Representations*, 2020.

[125] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.

[126] Wen Sun, Sachin Patil, and Ron Alterovitz. High-frequency replanning under uncertainty using parallel sampling-based motion planning. *IEEE Transactions on Robotics*, 31(1):104–116, 2015.

[127] Wen Sun, Luis G Torres, Jur Van Den Berg, and Ron Alterovitz. Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In *Robotics Research*, pages 685–701. Springer, 2016.

[128] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[129] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[130] MTCAJ Thomas and A Thomas Joy. *Elements of information theory*. Wiley-Interscience, 2006.

[131] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[132] Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

[133] 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. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[134] Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. Safe reinforcement learning via curriculum induction. *Advances in Neural Information Processing Systems*, 33, 2020.

[135] Jur Van Den Berg, Pieter Abbeel, and Ken Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[136] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[137] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[138] Dequan Wang, Coline Devin, Qi-Zhi Cai, Fisher Yu, and Trevor Darrell. Deep object-centric policies for autonomous driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8853–8859. IEEE, 2019.

[139] Hongqiang Wang, Jie Chen, Henry YK Lau, and Hongliang Ren. Motion planning based on learning from demonstration for multiple-segment flexible soft robots actuated by electroactive polymers. *IEEE Robotics and Automation Letters*, 1(1):391–398, 2016.

[140] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

[141] Xuesu Xiao, Jan Dufek, and Robin R Murphy. Robot risk-awareness by formal risk reasoning and planning. *IEEE Robotics and Automation Letters*, 5(2):2856–2863, 2020.

[142] Kelvin Xu, Siddharth Verma, Chelsea Finn, and Sergey Levine. Continual learning of control primitives: Skill discovery via reset-games. *Advances in Neural Information Processing Systems*, 33, 2020.