Human-Robot Skill Transfer via Dynamic Movement Primitives based on Reinforcement Learning

by

Jayden Hong

B.Eng., Yonsei University, 2014

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Mechanical Engineering

Human-Robot Skill Transfer via Dynamic Movement Primitives based on Reinforcement Learning

by

Jayden Hong

B.Eng., Yonsei University, 2014

Supervisory Committee

Dr. Homayoun Najjaran, Supervisor

(Department of Mechanical Engineering)

Dr. Hong-Chuan Yang, Outside Member

(Department of Electrical and Computer Engineering)

ABSTRACT

Efficient trajectory adaptation is crucial for improving overall robot performance. The use of Reinforcement Learning (RL), despite its promise in robot motion planning, suffers from long training times and limited generalizability. Learning from Demonstrations (LfD) offers an alternative solution by transferring human-like skills to robots. However, human demonstrations may not align optimally with robot dynamics due to biomechanical differences.

To address these challenges, this thesis proposes novel frameworks that combine RL, LfD, and the Dynamic Movement Primitives (DMP) framework. The DMP framework overcomes LfD limitations but requires parameter tuning of second-order dynamics. In this work, a systematic approach is introduced to extract dynamic features from human demonstrations, enabling automatic parameter tuning within the DMP framework. These extracted features facilitate skill transfer to RL agents, leading to more efficient trajectory exploration and significantly improved robot compliance.

Additionally, the thesis presents a framework that integrates Implicit Behavior Cloning (IBC) with DMP to leverage RL training speed through human demonstrations. The framework demonstrates faster training, higher scores, and increased stability in both simulation and real robot experiments. Comparative studies highlight the advantages of the proposed method over conventional RL agents.

The findings of this thesis hold significant implications for enhancing performance and adaptability of robots in practical applications. By incorporating human expertise from demonstrations to leverage conventional RL methods, this research offers novel approaches to improving efficiency and generalizability in robot motion planning.

Table of Contents

List of Tables

List of Figures

Lay Summary

The introduction of AI-powered robots has revolutionized automation and sparked speculation about the potential for robots to replace human labor entirely. AI has showcased impressive abilities in various domains, such as chatbots (like chatGPT), music composition (like MuseNet), art creation (like DALL·E 2), vision detection, and autonomous driving. However, challenges remain in the field of robotics, particularly when it comes to equipping robots with human-like object manipulation skills.

While robots can learn and acquire skills through trial and error, similar to how we learned to walk as babies, mastering new skills without guidance from a teacher is time-consuming and demanding. As a result, researchers have explored the idea of transferring human skills and knowledge to robots. Although some progress has been made, limitations arise due to the inherent structural differences between humans and robots, making it difficult for robots to accurately mimic human motion in an adaptive manner. Instead, robots should focus on understanding the underlying intentions and core concepts behind human motion. This understanding can then be applied in their own trial-and-error learning process.

This thesis delves into the exploration of a new way of human-robot skill transfer and investigates methods to enable robots to incorporate and leverage these skills in their own practice. By effectively uncovering and applying these principles, robots can enhance their capabilities and bridge the gap between human and robotic manipulation skills.

Preface

The work presented in this thesis was conducted at the Advanced Control and Intelligent Systems (ACIS) Laboratory at the University of Victoria, under the supervision of Dr. Homayoun Najjaran.

This work represents a culmination of efforts aimed at exploring the field of human-robot interaction and skill transfer. Throughout this thesis, various aspects of human hand motion planning and the transfer of skills between humans and robots is investigated.

Chapter 3 presents the Human Demonstration Dataset, which is published online in Z. Zhang and J. Hong's work titled "Dataset of human hand motion planning" in April 2023.

Chapter 4 presents the work that has been preprinted in J. Hong, Z. Zhang, A. M. S. Enayati, and H. Najjaran's paper titled "Human-robot skill transfer with enhanced compliance via dynamic movement primitives," available on arXiv:2304.05703 in 2023.

Lastly, Chapter 5 presents the work that has been submitted for review to Transaction on Robotics. Z. Zhang, J. Hong, A. M. S. Enayati, and H. Najjaran's manucript titled "Using Implicit Behavior Cloning and Dynamic Movement Primitive to Facilitate Reinforcement Learning for Robot Motion Planning"

Acronyms

BC	Behaviour Cloning
DDPG	Deep Deterministic Policy Gradient
DFE	Dynamic Feature Extraction
DMP	Dynamic Movement Primitives
DNN	Deep Neural Network
DoF	Degree of Freedom
DRL	Deep Reinforcement Learning
IBC	Implicit Behaviour Cloning
IK	Inverse Kinematic
IL	Imitation Learning
IRL	Inverse Reinforcement Learning
LfD	Learning from Demonstration
MDP	Markov Decision Process
P2PR	Point-to-Point Reaching
RBF	Radial Basis Function
RL	Reinforcement Learning
SD	Standard Deviation
SL	Supervised Learning
VR	Virtual Reality

ACKNOWLEDGEMENTS

I would like to acknowledge the support provided by Kinova® Inc. and the Natural Sciences and Engineering Research Council (NSERC) Canada under the Grant CRDPJ 543881-19. Their financial support enabled me to conduct this research and bring my ideas to fruition.

I am deeply grateful to my supervisor, Dr. Homayoun Najjaran, for his unwavering belief in me and his invaluable support throughout my research journey. His mentorship, both intellectually and emotionally, has been invaluable, and I am grateful for the guidance and encouragement he provided every step of the way. I would also like to extend my sincere appreciation to Dr. Zengjie Zhang, who served as my academic mentor, offering invaluable insights, brilliant ideas, and a comprehensive understanding of the field. His guidance and inspiration have greatly influenced my work and helped shape my research direction. To the members of the AIARA group, namely Amir Soufi and Joel Sol, I am deeply indebted for their willingness to engage in open discussions, providing valuable perspectives whenever I needed them, and offering guidance during moments of confusion. Their input has been crucial in helping me find my path when faced with challenges. A special mention goes to Todd Charter, who generously devoted his time to engaging in lab discussions, ranging from our respective fields to even choosing board games to play next. His support and friendship have been greatly appreciated. I would also like to acknowledge the invaluable guidance and support I received from all my colleagues in the ACIS Lab. Their collective expertise, shared insights, and collaborative spirit have significantly contributed to my research journey. I am sincerely grateful for their assistance and guidance.

Lastly, but certainly the most, I extend my heartfelt thanks to my wife for her unwavering support and love. Her presence and encouragement have been a source of strength and motivation throughout this entire journey. Also, I would like to express my gratitude to my family for their trust in me as I embarked on this journey of searching for my path.

The successful completion of this thesis would not have been possible without the collective efforts, support, and guidance of the aforementioned individuals and organizations. Their contributions have been indispensable, and I am truly grateful for their presence in my life.

To see a World in a Grain of Sand
And a Heaven in a Wild Flower
Hold Infinity in the palm of your hand
And Eternity in an hour
- From Auguries of Innocence, by William Blake

DEDICATION

This thesis is dedicated to my beloved wife, Renny.

Chapter 1

Introduction

This chapter provides an overview of the importance of transferring human skills to robots in the context of motion planning. It explores the benefits and limitations of two major approaches in motion planning: Learning from Demonstration (LfD) and Reinforcement Learning (RL). Through this exploration, the need for skill transfer methodologies that connect these approaches to leverage their respective benefits becomes evident. The chapter further emphasizes the contributions of the proposed approaches in the later chapters. Additionally, an outline of the thesis is introduced, providing a roadmap for the subsequent chapters.

1.1 Motivation

The integration of AI into robotics has revolutionized the field of Factory Automation, enabling higher levels of automation and reducing the dependence on human labor. In traditional manufacturing processes, human experts are often required to teach robots the path or trajectory motion, even for minor changes in the production line or variations in the products being manufactured. This manual process is time-consuming and often halts the entire production line until the commissioning is complete. To overcome these challenges,

robots need the ability to adapt to changes in their environments and generate optimal trajectories autonomously. In order to achieve this, intelligent robots must learn skills actively instead of being explicitly programmed by experts [1, 2].

As illustrated in Fig. 1.1, motion planning is an essential problem for more complicated tasks such as grasping [3], assembly [4], and manipulation [5]. The conventional approaches used for robot motion planning [6,7] include optimization-based methods, such as trajectory optimization [8] and sequential convex optimization [9], and the sampling-based methods, including exploration trees [10], probabilistic roadmaps [11], and model predictive control [12] which highly depend on the precise model of the environment. More recent methods, such as stochastic optimization-based planning (STOMP) [13], attempt to combine the advantages of both types of approaches. Moreover, more advanced techniques, such as Learning from Demonstration (LfD) and Reinforcement Learning (RL), can enable further advancements in motion planning.

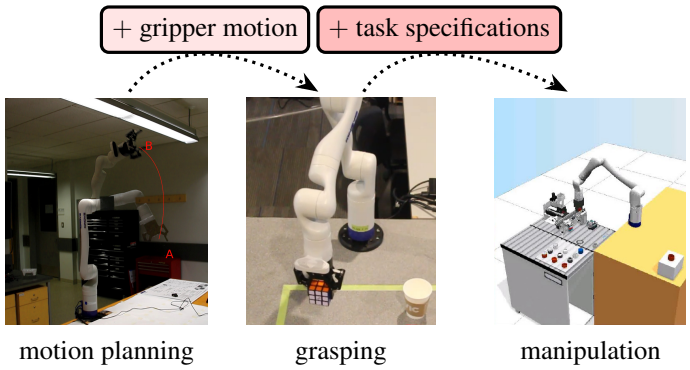


Figure 1.1: Motion planning is the essential problem of more complicated robotic tasks, e.g. grasping and manipulation.

Learning from Demonstration (LfD) for motion planning

LfD has emerged as a popular approach for transferring human knowledge to robots and enabling them to generate trajectories. LfD assumes that human demonstrations provide the ground truth for robots to imitate, making it a Supervised Learning (SL) approach. How-

ever, directly training robots using human motion has limitations due to the differences in embodiment between humans and robots. These differences can result in inefficient or undesirable trajectories when human motion is explicitly transferred to a robot. Additionally, regenerated trajectories often lack generalizability to variations in the environment. Moreover, the mechanical impedance of the robot's motor structure limits its capability to produce jerky movements, unlike the human muscle's rapid response. Therefore, in LfD, it is crucial for the robot to balance two conflicting goals simultaneously [14]: closely mimicking human behavior while maintaining system stability and safety.

To address these challenges, incorporating a dynamic system into the control model of LfD can be beneficial. By modeling complex nonlinear dynamics into a simpler one-degree-of-freedom (DoF) spring-damper dynamic system with perturbed force, robots can follow demonstrated trajectories while adhering to the compliance of the dynamic system. This dynamic system prevents noisy or jerky movements and enables the generalization of motion by extrapolating learned behaviors to new situations. The Dynamic Movement Primitives (DMP) framework has been successful in incorporating dynamic systems into LfD [15, 16]. DMP formulates the trajectory as a point attractor towards a goal position g. The gains of the DMP system correspond to the coefficients of the assumed dynamic system, such as dynamic stiffness K, damping D, and inertia K. These gains regulate the behavior of the dynamic system and play a crucial role in balancing human-likeness and system stability. Fine-tuning the gains ensures stable reproduction of the demonstrated trajectories [17].

Another technology within LfD is Behaviour Cloning (BC), where the agent learns human-like policies for robot manipulation tasks [18, 19]. The main idea of BC is to duplicate the human policy encoded in the demonstration data to a robot motion planner by SL methods [20].

While LfD has shown success in generating trajectories, the difference between human

biomechanics and robotic systems raises doubts about using human trajectories as the perfect ground truth. Human motor systems have more degrees of freedom (DoF) than their structural DoF, with multiple muscles involved in a 1-DoF motion. Muscle activations in humans are optimized to minimize factors such as muscle endurance [21], energy expenditure, or muscle fatigue [22], rather than solely focusing on achieving a specific joint movement as robots do. The maximum forces and endurance of muscles depend on posture during motion, which may cause inefficient motion in terms of duration and length of motion. For example, humans tend to avoid elbow-up posture, preferring elbow-down to minimize stress on muscles and tendons, even if it means following a longer trajectory. In contrast, robots can generate uniform torques within their range of motion, and optimal solutions for robots may differ from those for humans. Therefore, it is necessary to transfer more abstract information containing human skill rather than relying solely on explicit trajectories to ensure greater flexibility for robot manipulation.

Reinforcement Learning (RL) for motion planning

Reinforcement Learning (RL) provides another powerful approach for enabling robots to learn ideal manipulation policies through trial and error. Motion planning, a typical application of RL, requires a robot to move from an initial position to a goal position without colliding with obstacles in the environment [23]. RL offers an alternative to traditional optimization-based and sampling-based motion planning approaches that rely on precise models of the environment. RL enables an optimal planner, or an RL agent, to be trained automatically through interactions with the environment, without explicitly solving optimization problems based on the environment's model.

Compared to them, RL does not directly solve an optimization problem based on the precise model of the environment. Instead, an optimal planner, or an RL agent, can be trained automatically during interactions with the environment.

A typical case of RL is Deep Reinforcement Learning (DRL) which adopts an end-to-end learning scheme using deep neural networks [24]. It aims at constructing an all-in-one planner for highly-coupled and complicated robot tasks, especially the ones that depend on computer vision, such as grasping [25] and autonomous navigation [26]. Nevertheless, end-to-end learning suffers from the slow convergence rate of the training process and the sensitivity to environmental changes [2]. This is due to the large scale of the deep neural network and the high likelihood of overfitting. Efforts to resolve these issues include developing high-fidelity simulation platforms [27] or designing sim-to-real schemes to improve its adaptability and robustness [28].

To enhance the training speed and address overfitting, the use of heuristics in RL has been effective. Compared to end-to-end learning, exploiting heuristics can split a big RL problem into several smaller learning problems. Effective heuristic methods for RL include feature extraction layers [29], modularized neural networks [30], and hierarchical structures [31]. In [32,33], experimental studies are used to address that hierarchical-RL ensures faster convergence and better robustness than end-to-end learning. Transfer learning facilitated by heuristic mappings is used to transfer a simple-task policy to a complicated task without additional training [34]. Heuristic models, such as motion primitives [35, 36] are also widely used to simplify an RL problem by transforming a complex decision-making problem into a simpler domain.

In this sense, utilized in RL problems as well, enabling the RL agent to generate trajectories that comply with the second-order dynamic system of DMP. In RL with DMP, the RL agent explores a policy where the trajectory generated by the agent adheres to the DMP's dynamic system [37–39]. Similar to how the DMP framework in LfD maps trajectories in the task space to the forcing term in the DMP space, DMP in RL establishes a mapping between states (i.e., trajectories) and actions (i.e., forcing terms). This allows the RL agent to learn and generate trajectories that follow the desired behavior captured by the DMP.

RL with DMP provides benefits by incorporating the desirable properties of both RL and DMP. It leverages the flexibility and adaptability of RL while utilizing the dynamic system of DMP to guide the generation of compliant and smooth trajectories.

However, RL with DMP does have certain limitations. One limitation is that the human intention can only be conveyed to the RL agent through the reward signal, as opposed to directly utilizing the DMP's ability to encode and transfer human intention. In LfD with DMP, the DMP framework can effectively capture the human intention and encode it in the trajectory demonstration, but cannot utilize the benefits of RL's trial-and-error learning. In RL with DMP, the agent needs to learn the desired behavior solely through the reward signal, which can be a more indirect and potentially challenging process.

Combining LfD and RL for Skill Transfer in Motion Planning

Combining LfD and RL offers the potential to enhance skill transfer in motion planning. By integrating these approaches, the agent can explore alternative optimal strategies beyond explicit trajectories while still benefiting from the expertise and guidance of human experts implicitly. This integration enables skill transfer in motion planning, leveraging the advantages of both approaches while maintaining a balance between human-likeness and system stability provided by the Dynamic Movement Primitives (DMP) framework.

The LfD component plays a crucial role in capturing human skill and guidance. It allows the agent to learn from human demonstrations, acquiring abstract skills and capturing the intention behind the demonstrated trajectories. By observing and imitating human experts, the agent can gain valuable insights into effective motion planning strategies. On the other hand, the RL component empowers the agent to go beyond the demonstrations and explore alternative strategies by adapting its behavior to the environment. The transferred skill to the RL agent allows the improved of convergence speed, and enhanced efficiency in motion planning tasks.

The DMP framework serves as a promising connection between LfD and RL. By encoding complex nonlinear dynamics into a simpler dynamic system, the agent maintains compliance with the desired trajectories while ensuring system stability. The DMP framework provides the flexibility and adaptability necessary for skill transfer, allowing the agent to interpret human motion in terms of second-order dynamics and extract abstract features in the latent space of DMP.

In conclusion, this integrated approach enables the agent to leverage the expertise and guidance from human demonstrations, explore alternative strategies through RL, and benefit from the flexibility and adaptability provided by the DMP framework. It facilitates skill transfer in motion planning, enabling the agent to learn, adapt, and optimize its behavior effectively and in a balanced manner.

1.2 Contributions

This thesis investigates how RL agents can leverage abstract skills derived from human demonstrations, focusing on the extraction and utilization of these skills rather than relying solely on explicit paths or trajectories. By combining LfD and RL approach in motion planning, it is possible to enable the transfer of skills from human experts to robots while still benefiting from the flexibility and adaptability provided by RL. The integration of heuristic models and human demonstrations is believed to enhance the training speed and generalization capabilities of RL agents.

To achieve this objective, this thesis proposes two novel approaches.

Firstly, a novel framework is proposed to capture the dynamic characteristics of human motion within the DMP framework. Specifically, the framework focuses on the extraction of dynamic features, providing a new perspective on transferable skills that has not been extensively explored in existing literature.

Secondly, another novel RL method for robot motion planning, facilitated by DMP and Implicit Behaviour Cloning (IBC), is introduced. This method explores the potential of utilizing abstract skills obtained from human demonstrations, which has not been extensively investigated in existing literature.

Fig. 1.2 visually depicts the integration of the proposed frameworks, showcasing how they combine and connect key methodologies in motion planning.

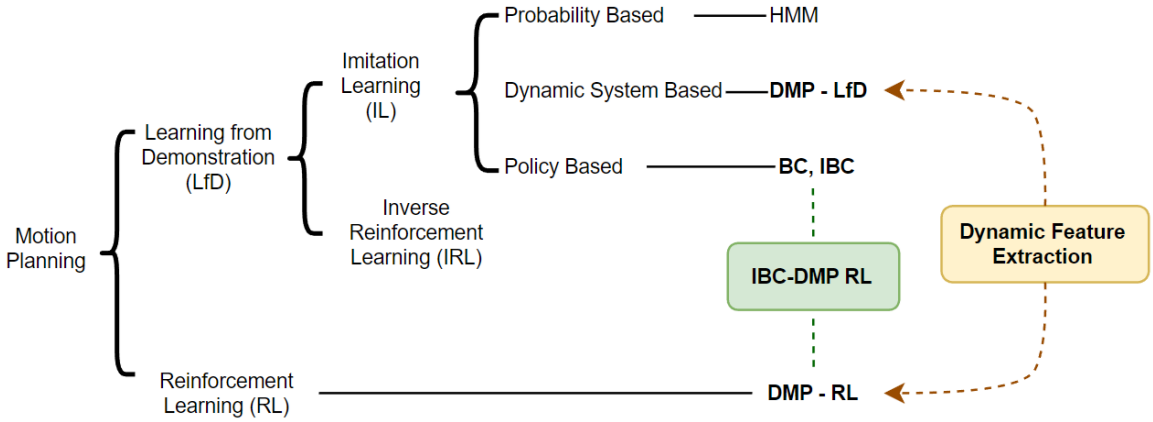


Figure 1.2: This thesis presents the two novel frameworks that connects LfD and RL approaches in motion planning: Dynamic Feature Extraction and IBC-DMP RL.

Through simulation studies, the efficacy of the proposed method and its advantages over conventional RL agents are validated in both suggested framework. Furthermore, an experimental study demonstrates the practical applicability of this approach in real-world robotic tasks.

The contributions of this thesis can be summarized as follows:

- Generated a dataset comprising human demonstrations in a point-to-point reaching task, focusing on abstract skills rather than explicit paths or trajectories. This dataset is made publicly available and serves as a valuable resource for future research on demonstration-facilitated methods for robot motion planning.

- Developed a novel framework for extracting abstract features that capture the dynamics of motion, specifically the dynamic features extraction of motion primitives.

These features can be applied to various DMP modeling approaches, including DMP-LfD and DMP-RL frameworks. This framework presents a new perspective on transferable skills, moving beyond the reliance on explicit trajectories commonly used in LfD.

- Introduced a novel framework for developing an IBC-DMP RL agent for motion planning in a robot manipulator with multiple degrees of freedom (Multi-DoF). This framework incorporates Multi-DoF DMP and IBC, aiming to improve the learning performance and generalizability of RL agents by effectively utilizing abstract skills acquired from human demonstrations.

- Conducted simulation and experimental studies to demonstrate the efficacy of the proposed methods in solving practical robotic problems. For instance, the scenario involving a robot carrying a cup full of coffee showcases the strength of adapting dynamic features extracted from human demonstrations to successfully perform delicate tasks related to velocity, acceleration, and jerk, even in simple point-to-point trajectories. Additionally, the scenario of cube stacking demonstrates the superiority of the proposed IBC-DMP RL method, which exhibits faster training convergence and higher test scores compared to conventional RL approaches.

By presenting these contributions, this thesis aims to advance the field of RL in robotics. It highlights the potential of combining heuristic models and human demonstrations to enhance the training efficiency and overall performance of RL agents, paving the way for more flexible and adaptable robotic systems.

1.3 Thesis Outline

This thesis is structured into six chapters, each contributing to the validation of claims and novel approaches to skill transfer between humans and robots in motion planning. The

overview of the main chapters is presented as follows:

Chapter 1: Introduction This chapter explores the benefits and limitations of two major approaches in motion planning for robots: Learning from Demonstration (LfD) and Reinforcement Learning (RL). It also highlights the motivation to and contributions of the proposed approach, which aims to leverage the strengths of both methods and address their respective limitations in the context of skill transfer. Furthermore, an outline of the thesis is introduced.

Chapter 2: Background This chapter provides an overview of important concepts and theories used in the thesis, including RL, Skill Transfer, and Dynamic Movement Primitives (DMP). It establishes the theoretical foundation for the subsequent chapters.

Chapter 3: Human Data Collection This chapter provides a detailed explanation of the methodology employed to collect human motion data using motion trackers. The dataset obtained in this chapter is utilized in the frameworks proposed in Chapter 4 and Chapter 5. Additionally, the dataset is openly accessible online, enabling other researchers to utilize it for similar studies.

Chapter 4: Human-Robot Dynamic Feature Transfer via DMP This chapter presents a novel framework that extracts dynamic features of motion and applies them to the RL agent within the DMP framework. The proposed method is validated through simulations and real robot experiments involving delicate point-to-point movements

Chapter 5: RL-based Robot Motion Planning Enhanced with IBC and DMP This chapter proposes another novel framework that incorporates demonstration into the training procedure of RL using Implicit Behaviour Cloning (IBC) and DMP. The proposed method demonstrates improvements in training speed and generalizability

compared to conventional RL methods. Validation is performed through simulations and pick-and-place experiments using real robots.

Chapter 6: Concluding Remarks The final chapter summarizes the main findings of the thesis, discusses the contributions of the research, and offers concluding remarks. It reflects on the importance of transferring human skills to robots and the potential impact of the proposed approaches. The chapter also identifies areas for further research and suggests future directions to enhance and expand upon the work presented in this thesis.

Chapter 2

Background

This chapter introduces the preliminary knowledge of the important concepts and theories used in the thesis, including Reinforcement Learning, Skill Transfer and Dynamic Movement Primitives. Also, we discuss the benefits and limitations of them to define the problem in Human-Robot Skill Transfer.

2.1 Reinforcement Learning (RL)

Overview

Reinforcement Learning (RL) is an approach that enables an *agent* to acquire an optimal strategy for acting in various situations based on the concept of rewards or punishments. The agent interacts with an *environment*, which responds to the agent's actions. Through this interaction, the agent gains knowledge of the current situation, known as the *state*, and receives *rewards* based on its chosen *actions*. RL involves finding the optimal *policy*, which specifies the action to be taken in each state, to maximize cumulative rewards.

Markov Decision Process (MDP)

Markov Decision Process (MDP) is a mathematical modeling that forms a foundation of RL. It assumes the case where the state transition depends only on the current state, regardless of historical states.

A MDP is represented by a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathscr{F}, \mathscr{R}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space of the agent, respectively, and $\mathscr{F} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and $\mathscr{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ are the state transition and the reward functions, and γ is the discount factor that determines the importance of future rewards. RL aims to find the optimal policy π^* that maximizes the sum of rewards over time:

$$\pi^* = \arg\max_\pi \sum_{t=0}^{T} \mathscr{R}(s_t, \pi(s_t)) \,, \tag{2.1}$$

where $T \in \mathbb{N}^+$ is the length of a trajectory $s_0 s_1 \cdots s_T$, and $s_t \in \mathcal{S}$ is the state of the MDP at time $t = 0, 1, \cdots, T$.

Bellman Equation

The value function plays a crucial role in RL. It estimates the expected return from a given state or state-action pair. The state value function $V_\pi(s)$ represents the expected return when following policy π from state s, while the action value function $Q_\pi(s, a)$ represents the expected return when taking action a in state s. These value functions satisfy the Bellman equations:

$$V_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V_\pi(s')), \tag{2.2}$$

$$Q_\pi(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') Q_\pi(s', a')) \tag{2.3}$$

where $p(s', r|s, a)$ is the probability of transitioning to state s' with reward r from state s by taking action a. The optimal value functions, denoted as $V_(s)$ and $Q_(s, a)$, are obtained by solving the Bellman Optimality Equations:

The optimal solution to the Bellman equations (2.2) and (2.3), or *Bellman Optimalityy Equations* is

$$V_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V_*(s')), \tag{2.4}$$

$$Q_*(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma \max_{a' \in \mathcal{A}(s')} Q_*(s', a')) \tag{2.5}$$

where $V_*(s')$ and $Q_*(s, a)$ are the optimal state value function and the optimal action value function, respectively. The optimal policy π^* is the policy when the optimal value function is found.

While the Bellman Optimality Equation (2.4) and (2.5) are relatively straightforward, solving direct solution to them is not trivial in many cases. In real-world problems, the state space can be extremely large or continuous, making it computationally infeasible to solve the equation directly. For example in robot motion planing, the state space can be defined as the continuous task or joint space with 6 Degree of Freedom (DoF) $\mathcal{S} := \mathbb{R}^6$, leading to infinite number of possible states known as *curse of dimensionality*. Also, the complex dynamics of the the state transition often involves non-linearity and non-convexity even in the case such that the transition possibility is deterministic. Due to these reason, iterative methods such as bootstrapping are commonly used to approximate the optimal solution.

Using Deep Neural Network (DNN) is one of the most popular and powerful approach to approximate the optimal solution, as DNN can learn any complex mapping between the states(or states and actions) and their corresponding value. The parameters of the value or/and policy approximator are iteratively updated using the interaction between the agent

and environment, until the optimal reward is steadily achieved. Typically, RL agents with larger or more complex neural networks are likely to solve more complicated tasks. On the other hand, they tend to spend longer training times and are easier to cause the overfitting problem.

On-policy and off-policy methods in RL

RL methods can be categorized into on-policy and off-policy approaches depending on whether the policy to be promoted is the one that is interacting with the environment. Both types of approaches are widely used for robot motion planning [23, 40, 41].

A typical on-policy method is Proximal Policy Optimization (PPO) which updates the policy using the policy gradient calculated from the perturbed system trajectories [42]. On-policy methods typically promote policies only after complete episodes. Also, historical policies are no longer used for policy promotion.

On the contrary, the policy update of off-policy RL methods, such as Deep Deterministic Policy Gradient (DDPG) [43], can be performed at any time. Off-policy methods can also utilize the historical data stored in the experience replay buffer to promote the current policy [44]. This means that off-policy methods can learn from multiple policies encoded in the historical data, which is its main advantage over on-policy methods. Nevertheless, the main shortcomings of off-policy methods are long training times and unstable training behaviors due to the bootstrapping effect in the initial stage of the training process. In fact, the training performance of an off-policy agent is highly dependent on the quality of the experience data. In the application of robot motion planning, it is commonly witnessed that on-policy methods outperform off-policy methods when the quality of the experience data is not sufficiently good [45, 46]. Therefore, many efforts are devoted to improving the performance of off-policy agents by refining the experience data [47, 48].

Deep Deterministic Policy Gradient (DDPG)

In this thesis, DDPG is used as the baseline model since it allows for improving the current policy using the experience data. A basic DDPG agent typically consists of two neural networks, namely an actor π_θ and a critic Q_w which are respectively used to approximate the optimal policy π^* as shown in (2.1) and the value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, where θ and w denote the parameters of the neural networks. Details about the definition of the value function can be found in [47]. The main objective of the training process for a DDPG agent is to constantly update the values of the parameters θ and w, such that the approximations $\pi_\theta \to \pi^*$ and $Q_w \to Q$ are as close as possible. Another two neural networks $\pi_{\theta'} : \mathcal{S} \to \mathcal{A}$ and $Q_{w'} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with parameters θ' and w', referred to as *target neural networks* are typically used to smooth out the approximation process. The parameters θ' and w' are iteratively updated following the updates of θ and w,

$$\theta' \leftarrow \lambda\theta' + (1-\lambda)\theta,\ w' \leftarrow \lambda w' + (1-\lambda)w, \tag{2.6}$$

where $0 < \lambda < 1$ is an interpolation factor used to average the updates of the target networks and stabilize the approximation.

The training of the critic network Q_w is a supervised learning process. The samples used to train the network are from an *experience replay buffer* $\mathcal{B}$ sized $N \in \mathbb{N}^+$, which is randomly sampled from an experience replay buffer $\mathcal{B}$. Each sample in the buffer $\mathcal{B}$ is organized as the format $\{ s_j, a_j, s'_j, r_j, d_j\}$, where $s_j \in \mathcal{S}$ and $a_j \in \mathcal{A}$ are the state and the action of the agent at a certain history instant $j = 1, 2, \cdots, N$, $s'_j = \mathscr{F}(s_j, a_j)$ is the *successive state*, $r_j = \mathscr{R}(s_j, a_j)$ is the corresponding instant reward, and d_j is a boolean termination flag that determines whether an episode terminates or not. The critic

loss function of the buffer $\mathcal{B}$ is computed as

$$\mathcal{L}_C(\mathcal{B}) = \frac{1}{N}\sum_{j=1}^{N}\left(l_j - Q_w(s_j, a_j)\right)^2 \tag{2.7}$$

where $l_j = r_j + \gamma(1-d_j)Q_{w'}(s'_j, \pi_{\theta'}(s'_j))$ is the label of sample (s_j, a_j). The actor network π_θ is also trained using the buffer $\mathcal{B}$ with the following loss function,

$$\mathcal{L}_A(\mathcal{B}) = -\frac{1}{N}\sum_{j=1}^{N} Q_w(s_j, \pi_\theta(s_j)). \tag{2.8}$$

Given the computed losses $\mathcal{L}_A$ and $\mathcal{L}_C$, the parameters of the actor and the critic networks are updated with the following gradient-based law, in an iterative and alternative manner,

$$\Delta\theta = -\alpha_\theta \nabla_\theta \mathcal{L}_A,\ \Delta w = -\alpha_w \nabla_w \mathcal{L}_C, \tag{2.9}$$

where $\Delta\theta$ and Δw are the parameter increments at each iteration, $\alpha_\theta, \alpha_w \in \mathbb{R}^+$ are the learning rates of the actor and the critic networks, and $\nabla_\theta \mathcal{L}_A$ and $\nabla_w \mathcal{L}_C$ are the gradients of the loss functions to the network parameters.

The Training Algorithm for DDPG Agent

The training procedure for a DDPG agent follows Algorithm 1. The agent initializes a replay buffer and the actor and critic networks. It then interacts with the environment, collecting experiences and storing them in the replay buffer. The agent samples a batch of experiences from the buffer and uses them to update the critic network by minimizing the mean squared error loss. The actor network is updated by computing the gradients of the expected action value with respect to the network parameters and applying gradient ascent. Target networks are periodically updated by slowly tracking the main network parameters. This process is repeated for a specified number of episodes.

Algorithm 1 DDPG Algorithm

Initialize replay buffer $\mathcal{B}$
Randomly initialize the actor network π_θ and critic networks Q_w
Initialize the target networks $\pi_{\theta'}$, $Q_{w'}$ with $\theta' \leftarrow \theta$, $w' \leftarrow w$
for $i \leftarrow 1$ **to** N_{ep} **do**
  Sample a random noise ϵ_t
  Initialize the state $s_0 = \mathcal{X}_0$
  for $t \leftarrow 0$ **to** T **do**
    Observe the state $s_t = \mathcal{X}_t$
    Sample an action $a_t = \pi_\theta(s_t) + \epsilon_t$
    Observe the successive state $s'_t = \mathcal{X}_{t+\Delta t}$
    Calculate the instant reward r_t and flag d_t
    Store $\{s_t, a_t, s'_t, r_t, d_t\}$ to $\mathcal{B}$
    Sample a random batch of N transitions $\{s_i, a_i, s'_i, r_i, d_i\}$ from $\mathcal{B}$
    Set $y_i = r_i + \gamma Q'_w(s'_i, a'_i)$ where $a'_i = \pi'_\theta(s'_i)$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q_w(s_i, a_i))^2$
    Update the actor policy using the sampled policy gradient:
      $\nabla_\theta J \approx \frac{1}{N}\sum_i \nabla_a Q_w(s,a)|_{s=s_i, a=\pi(s_i)} \nabla_\theta \pi_\theta(s)|_{s_i}$
    Update the target networks using (2.6)
  end for
end for

2.2 Skill Transfer in Motion Planning

In order to discuss the transfer of human skills to robots, it is essential to define the concept of transferable skills. In the survey conducted on skill transfer, researchers identified these skills as ”the robot's manipulation skills in a complex and dynamic environment” [49, 50]. Based on this, the approaches to achieving human skills can be broadly categorized into three main groups: Neurophysiological Signal, Reinforcement Learning, and Learning from demonstration.

Neurophysiological Signal in Skill Transfer

Neurophysiological signal-based methods aim to transfer human skills to robots by capturing and interpreting signals directly from the human body. These signals, such as electromyography (EMG) or brain activity signals, can provide valuable information about human intentions and motions. By mapping these signals to robot control commands, it becomes possible to replicate human skills in a robotic system. The approaches to transfer the stiffness of the human motion using EMG signal [51–53] has shown the possibility that dynamic characteristics can deliver human skill successfully as well as the trajectory itself in a way that the robot remained robust stability despite the unexpected external perturbation. However, this approach often requires additional devices to sense and interpret the signals, and it may have limitations in terms of signal accuracy and the optimization of human-like motion for robotic systems.

Reinforcement Learning (RL) in Skill Transfer

When it comes to transferring human skill in skill transfer using Reinforcement Learning (RL), human skill is usually conveyed through reward design. The RL agent is provided with a reward function that encourages the agent to mimic or replicate the behavior demonstrated by humans. By shaping the reward function appropriately, the agent can learn to

perform tasks in a manner similar to how a human expert would. However, the reward signal is often indirect and can be challenging to design accurately. It may not fully capture the nuances and intricacies of the human expertise being transferred. Additionally, the process of reward shaping can be time-consuming and requires careful fine-tuning to ensure that the agent acts according to the intended human skill.

Another approach to skill transfer in RL is through the use of a replay buffer that contains human demonstrations. In this method, the replay buffer is initially populated with demonstrations provided by human experts. The RL agent then learns from these demonstrations by replaying them during the training process. While utilizing the replay buffer for skill transfer seems straightforward, it has its own challenges. The explicit use of human demonstrations in the replay buffer can lead to the agent being trapped in a solution already present in the demonstrations. The agent may exploit the similar trajectories from the demonstrations without fully exploring alternative solutions that may be more efficient or effective. This restricts the flexibility and adaptability of RL algorithms.

Learning from Demonstration (LfD) in Skill Transfer

In LfD, Inverse Reinforcement Learning (IRL) and Imitation Learning (IL) are two popular and important approaches for skill transfer in motion planning. Both LfD and IRL have been extensively applied in robot motion planning and autonomous driving scenarios.

IRL is an approach that aims to invert the framework of RL by learning the reward function from given demonstrations. Since the optimal reward function often involves nonlinear terms and complexities from robot dynamics, Deep Neural Network (DNN) can be used to represent the reward function effectively. However, there are limitations to IRL. One major limitation is the potential overfitting of the learned rewards and policies to the expert demonstrations. This issue becomes more critical when there is noise or errors in the demonstration data. Overfitting can lead to a lack of generalization to new situations.

Another limitation is the ambiguity in determining the true reward function solely from demonstration data. Different reward functions can result in the same observed behavior, making it challenging to uniquely identify the underlying reward function. Additionally, even though IRL learns the reward function, the RL agent still needs to be trained using the learned reward function. This can require computational resources, especially when compared to conventional RL methods that directly use handcrafted reward functions.

IL is an approach in which an agent learns a policy or behavior by directly imitating expert demonstrations. IL involves observing and mimicking the actions and decisions of a human or expert agent to acquire specific tasks or behaviors. Various approaches can be utilized to model and interpret human behavior for learning these skills. For example, Behavior Cloning is a common IL technique that focuses on replicating expert actions based on observed states to learn policy behind the human motion. Additionally, techniques such as Hidden Markov Model(HMM) and Dynamic Movement Primitives (DMP) can be employed in IL to capture sequential aspects of human behavior or motion. HMMs offer a probabilistic modeling approach, while DMPs utilize dynamic systems to represent and reproduce complex movement patterns.

2.2.1 Behavior Cloning (BC)

BC is an important technology of IL to duplicate human policy from demonstrations. It has been widely applied to robot motion planning and autonomous driving [20, 54]. Here, behavior refers to what *actions* humans tend to take under certain *states*. Cloning means learning a new policy to fit human behaviors. BC formulates a supervised learning problem where human actions serve as the ground truth labels of the states. Then, the policy to be cloned can be trained using deep neural networks (DNN) [55] or Gaussian mixture models (GMM) [56]. When human demonstrations are recorded as movement trajectories, states refer to the position and velocity of human trajectories at a certain time and action is the

corresponding acceleration [18]. DMP has also been used to simplify the BC process [57]. Nevertheless, calculating acceleration from human demonstrations requires the second-order derivative operation which brings up differential noise to the human demonstration samples. Additional procedures such as locally weighted regression (LWR) are often used to mitigate the effects of the noise [58].

Another issue of BC is that the pre-trained policy might overfit human behaviors. Note that human likeness does not necessarily indicate the best planning policy. Thus, overfitting human behaviors may degrade the performance of robot motion planning with respect to the predefined reward. It is addressed that BC performs worse than nominal RL agents in many cases due to overfitting [59]. This motivates us to combine BC and nominal RL to develop a better robot motion planner, instead of directly using BC to generate human-like trajectories. The conventional BC-facilitated RL methods usually perform a fully separate scheme, i.e., to learn an initial policy with BC before training the agent using nominal RL methods [60–62]. Nevertheless, in such a separate scheme, BC is not fully exploited to promote the training of the RL agent. It also lacks the flexibility of achieving a balance between human likeness and the predefined reward. Recent studies made some attempts to integrate BC into the off-policy RL training using a dual-buffer structure [63, 64], which inspires us to use BC from human demonstration to leverage the training performance of RL.

BC refers to the process of training a target policy from a source policy encoded in a demonstration. Given a set of demonstrations that are stored in a buffer $\mathcal{B}$ as defined in Section 2.1, the source pattern refers to the policy $\pi_{\mathcal{B}}$ that fits the data (s_j, a_j) for all $j = 1, 2, \cdots, N$. The objective of BC is to train a policy π_μ to duplicate $\pi_{\mathcal{B}}$. Conventionally, BC renders a supervised learning problem, where the policy parameter μ is updated via a gradient-based law $\Delta\mu = -\alpha_\mu \nabla_\mu \mathcal{L}_I$, where $\Delta\mu$ is the parameter increment, $\alpha_\mu \in \mathbb{R}^+$ is

the learning rate, and $\nabla_\mu \mathcal{L}_I$ is the gradient of a loss function $\mathcal{L}_I$ defined as

$$\mathcal{L}_I(\mathcal{B}) = \sum_{j=1}^{N} \mathscr{M}^2(a_j - \pi_\mu(s_j)) \tag{2.10}$$

where $\mathscr{M} : \mathcal{A} \to \mathbb{R}_{\geq 0}$ is a non-negative function that evaluates the deviation between the two policies, which is commonly the 2-norm of vectors. Such an approach is also referred to as *explicit BC (EBC)* since the loss function (2.10) explicitly penalizes the deviation between actions of the source and the target policies. EBC is very likely to cause overfitting since it is obsessed with the fitting of two policies. It may not be successful when the demonstration data is noisy and subject to large variance. Recent study proposes an *implicit BC (IBC)* method which suggests the following loss function [65],

$$\mathcal{L}_I(\mathcal{B}) = \sum_{j=1}^{N} E_\mu(s_j, a_j), \tag{2.11}$$

where $E_\mu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ is a non-negative energy function of demo data (s_j, a_j) parameterized by μ. Thus, IBC generates the target policy through an implicit energy function, which can effectively avoid the overfitting problem. It also provides a flexible interface to combine BC with an off-policy RL agent. A technical question will be answered in Chapter 5 to showcase how to design a proper energy function to facilitate RL with IBC.

2.2.2 Dynamic Movement Primitive (DMP)

A motion primitive is a heuristic model commonly used for robot motion generation. A special motion primitive is the dynamic movement primitive (DMP) designed for robot manipulators. The mathematical formulation of DMP is a virtual second-order linear dynamic model. The state of the DMP model is the position and velocity of the robot trajectory to be generated. Its input is an actuation function to be learned. The trajectories generated by DMP are ensured with inherent smoothness and stability due to the continuity

of second-order dynamic models. DMP was originally proposed as an abstract model to describe human motions [66]. Then, it is applied to IL for the learning of human movement patterns [67] since DMP provides an elegant and simple model to depict human motions.

DMP is a virtual dynamic model used to generate desired trajectories for a moving point [68, 69], defined as follows,

$$\tau\ddot{x}_t = \alpha(\beta(x_{\mathrm{g}} - x_t) - \dot{x}_t) + \zeta_t(x_{\mathrm{g}} - x_0)f(\zeta_t), \tag{2.12a}$$

$$\tau\dot{\zeta}_t = -\omega\zeta_t, \tag{2.12b}$$

where $x_t, \dot{x}_t, \ddot{x}_t \in \mathbb{R}$ are, respectively, the time-dependent position, velocity, and acceleration of the desired trajectory to be generated, $x_0, x_{\mathrm{g}} \in \mathbb{R}$ are the initial and the goal positions of the desired trajectory, $\zeta_t \in \mathbb{R}$ is a canonical dynamic variable with a non-zero initial value $\zeta_0 \in \mathbb{R}$, $\tau \in \mathbb{R}^+$ is a constant temporal scalar that depicts the inertia of DMP, $\alpha, \beta \in \mathbb{R}^+$ are constant parameters that determine the damping and stiffness of the DMP, $\omega \in \mathbb{R}$ is a constant parameter that controls the duration of the DMP, and $f : \mathbb{R} \to \mathbb{R}$ is the actuation function that determines the shape of the generated trajectory. With a proper actuation function f, the DMP model (2.12) generates a trajectory x_t which has the continuous position, velocity, and acceleration with respect to time. Therefore, it guarantees smoothness when applied to robot motion planning. The actuation function f is solved by minimizing a certain cost. Conventionally, DMP uses a Radial Basis Function (RBF) neural network to approximate the optimal actuation function, i.e., $f(\zeta_t) = \Big(\sum_{j=1}^{M}\varphi_j(\zeta_t)\omega_j\Big)\Big/\sum_{j=1}^{M}\varphi_j(\zeta_t)$, where $M \in \mathbb{N}^+$ is the number of the RBFs, $\omega_j \in \mathbb{R}^+$ are constant weights, $\varphi_j : \mathbb{R} \to \mathbb{R}$ is a RBF $\varphi_j(x) = \exp\Big(-\frac{(x-c_j)^2}{2\sigma_j^2}\Big)$, $j = 1, 2, \cdots, M$, where $c_j, \sigma_j \in \mathbb{R}$ are constant parameters. The RBF-based actuation function can be solved using on-policy RL methods [68]. DMP is usually used to directly generate the desired trajectory for a robot end-effector in the Cartesian space, where an Inverse Kinematic (IK) algorithm is needed to map the trajectory to

the joint space [36].

When human demonstrations are unavailable, DMP can be solved using RL with a predefined cost function [70]. It is very effective to simplify a complicated robot motion planning problem via hierarchical RL [68]. DMP-facilitated RL has become a popular method for robot motion planning in recent work [69, 71, 72]. A survey on the application of DMP to robot manipulation problems is presented in [73]. Nevertheless, DMP has been mainly solved using on-policy RL methods, which makes it difficult to promote the current policy using demonstration data. Using off-policy methods to train a DMP model is still an unsolved problem.

Chapter 3

Human Data Collection

This chapter provides a detailed explanation of the methodology employed to collect human motion data. Acquiring human data is essential to leverage human skills and apply them effectively to robots. Two types of hand trajectories were recorded as demonstrations: a simple Point-to-Point Reaching (P2PR) task and a P2PR task involving obstacle avoidance. The dataset obtained in this chapter was used for both human-robot dynamic feature transfer via Dynamic Movement Primitives (DMP) in Chapter 4 and Reinforcement Learning (RL) based robot motion planning enhanced with Implicit Behaviour Cloning (IBC) and DMP in Chapter 5. The dataset is openly accessible online, allowing researchers to freely utilize it for other demonstration-assisted approaches in robot motion planning.

3.1 Human Data Acquisition

A data-recording experiment is conducted to collect human-hand motion in a point-to-point reaching (P2PR) task, which is a typical scenario in robot motion planning. The experiment is designed to capture how the human hand attempts to reach a goal position while avoiding collisions with a mid-way obstacle. An HTC VIVE tracking system, consisting of several motion trackers and a pair of Virtual Reality (VR) base stations is used to record the position

of any object attached to the trackers. The system is able to track the Cartesian pose of the rigid body in a stable sampling frequency $100\,\text{Hz}$. The HTC Vive Tracker (2018) with SteamVR 2.0 has been examined in [74] and has been found to have a tracking precision of 2 mm and a tracking accuracy of 0.007 m.

The top view of the experiment configuration is shown in Fig. 3.1. A human subject sits in front of a flat desk. The desk is positioned $0.35\,\text{m}$ vertically higher than the seat, ensuring the subject's comfortable seating. A rectangular planar motion region is marked on the desk with height $0.5\,\text{mm}$ and a width of $0.6\,\text{m}$, shown as a thick-line gray box in Fig. 3.1. The motion region is aligned with the desk edge towards the subject. The subject's seat is placed such that the hand can be comfortably positioned in the left-bottom corner of the motion region and can naturally reach any point within the motion region without standing up. Two VR base stations are placed on the diagonal corners of the workspace for superior tracking precision, as shown as camera symbols in Fig. 3.1. The coordinate of the tracking system is calibrated such that its origin coincides with the left-bottom corner of the workspace and its x-, y-, and z-axis point to the right, front, and top of the subject, respectively. The initial position, denoted as $\mathcal{I}$, is set to the position where the hand is comfortably placed at the origin. For a given goal $\mathcal{G}$ and an obstacle $\mathcal{O}$, the subject is required to move the hand from the initial position $\mathcal{I}$ to the goal $\mathcal{G}$ in a natural manner while avoiding the obstacle $\mathcal{O}$. A possible hand trajectory is illustrated as a dashed arrow in Fig. 3.1.

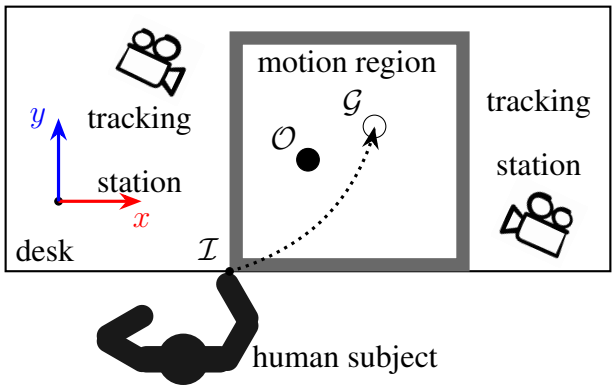
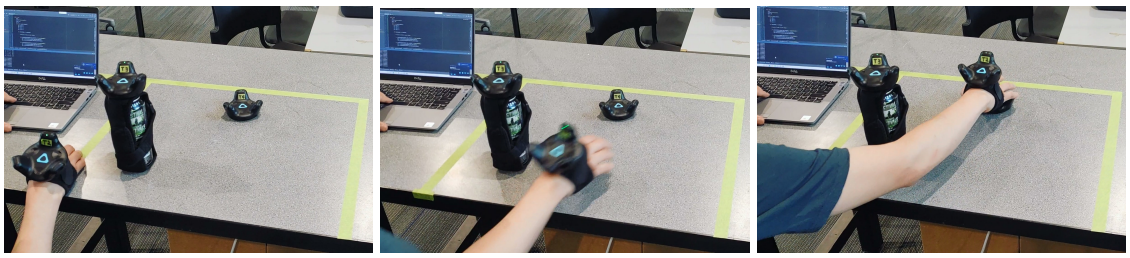


Figure 3.1: The top view of the configuration of the data recording experiment, where $\mathcal{I}$, $\mathcal{O}$, and $\mathcal{G}$ are the initial, obstacle, and goal positions, respectively.

A human demonstration dataset was recorded using the experimental setup described above. The dataset is available in [75]. To record the data, a right-handed young male played the subject. Fig. 3.2 illustrates the data recording process. As illustrated in the figure, Three trackers are utilized to mark the positions of the obstacle, the human hand, and the goal. In this experiment, a cylindrical bottle with a height of 66 mm and a diameter of 200 mm serves as the obstacle. The tracker is fixed to the top of the obstacle, resulting in a total height of 70 mm. Besides, the demonstrator ties a tracker to the wrist of the subject to track the motion of the hand. The third tracker is directly placed on the desk to serve as the goal. During the entire recording process, the subject performs 600 trials of P2PR motion repeatedly. For each trial, the goal $\mathcal{G}$ is placed at a random position in the motion region. It should be not too close to the initial position $\mathcal{I}$ to ensure the feasibility of the hand motion. The obstacle $\mathcal{O}$ is also positioned at a random point close to the mid-way between the initial position $\mathcal{I}$ and the goal $\mathcal{G}$. Note that the obstacle should be neither beside the goal nor too close to the initial position. Otherwise, it would be very difficult to produce a natural trajectory. The subject is asked to always perform the motion and avoid the obstacle in the most comfortable manner. Other than that, there are no restrictions from which direction the subject must avoid the obstacle. The subject is allowed to bend the

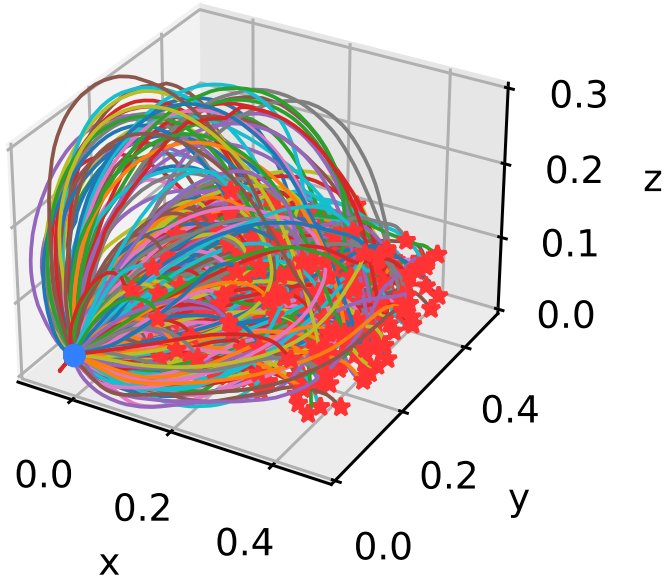
torso but cannot stand up from the seat to reach a far point.



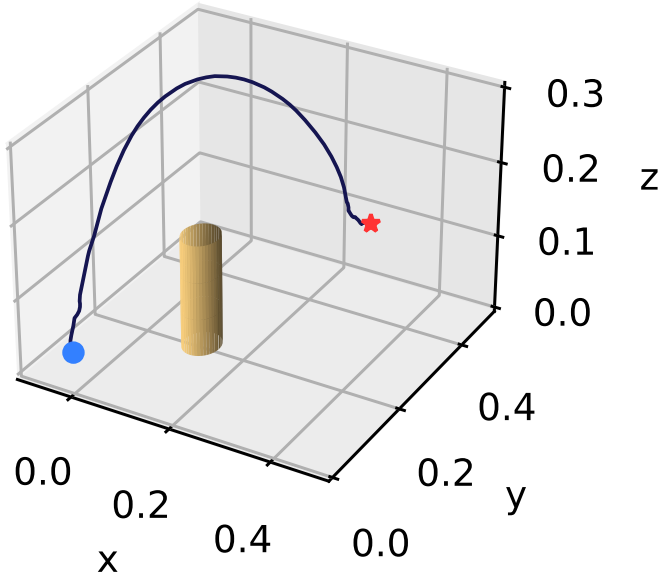
(a) (b) (c)

Figure 3.2: The illustration of the human data recording process. (a). The hand is placed at the initial position with the obstacle placed in the mid-way toward the goal tracker. (b). The hand avoids the obstacle while moving toward the goal. (c). The hand reaches the goal.

After removing the invalid motions, a total of 544 hand trajectories were obtained, corresponding to various goal positions and obstacle positions, as illustrated in Fig. 3.3a. Each trajectory contains the hand position of one P2PR motion. It can be seen that different trajectories show great variety in their shapes. Taller trajectories indicate that the hand goes over the obstacle and short ones pass around the obstacles. A common feature of these trajectories is that they all successfully avoid obstacles while maintaining the smoothness of the shape. Fig. 3.3b gives a more clear perspective of how a hand trajectory reaches the target and avoids the obstacle.



(a) All hand trajectories



(b) One hand trajectory

Figure 3.3: The illustration of the recorded human hand trajectories, where the blue dot denotes the starting position and the orange stars are the goal positions. (a) The hand trajectories of all trials. (b) A representative hand trajectory with a cylinder denoting the obstacle.

3.2 Data Preprocessing

The collected raw data includes position measurements at each timestamp of around 0.1 second. In order to generate trajectory data, it is necessary to calculate the velocity and acceleration.

The linear velocity of the trajectory can be computed as follows: $\dot{\mathbf{x}}^H_{\tau_i} = \dfrac{\mathbf{x}^H_{\tau_i} - \mathbf{x}^H_{\tau_{i-1}}}{\Delta\tau}$, $i = 1, 2, \cdots, N$, where τ_i represents the timestamp when each position data $\mathbf{x}^H_{\tau_i}$ is recorded, and N is the total number of time steps. Similarly, the acceleration of the trajectory can be calculated using: $\ddot{\mathbf{x}}^H_{\tau_i} = \dfrac{\dot{\mathbf{x}}^H_{\tau_i} - \dot{\mathbf{x}}^H_{\tau_{i-1}}}{\Delta\tau}$, $i = 1, 2, \cdots, N-1$.

Depending on the specific use cases of the dataset, the achieved trajectory can undergo denoising, regularization, normalization, or standardization.

Chapter 4

Human-Robot Dynamic Feature Transfer via DMP

This chapter presents a systematic method to extract the dynamic features from human demonstration to auto-tune the parameters in the Dynamic Movement Primitives (DMP) framework, namely *Dynamic Feature Extraction (DFE)*. In addition to its use with Learning from Demonstration (LfD), another utility of the proposed method is that it can readily be used in conjunction with Reinforcement Learning (RL) for robot training. In this way, the extracted features facilitate the transfer of human skills by allowing the robot to explore the possible trajectories more efficiently and increasing robot compliance significantly. Additionally, this chapter introduced a methodology to extract the dynamic features from multiple trajectories based on the optimization of human-likeness and similarity in the parametric space.

4.1 Related Work

In both LfD and RL with DMP framework, auto-tuning of the M, D, K parameters, or *dynamic features*, can help to increase the degree of autonomous adaptation and boost the

advantage of DMP. However, finding a proper set of dynamic features has a more significant meaning for skill transfer. As M, D, K are the assumed coefficient of virtual dynamic system to formulate DMP, they determine the spring-damping responses of the dynamic system. An inappropriate dynamic features may lead to desired motions in both frameworks. In other words, the optimal dynamic features in DMP represents the spring-damping characteristic demonstrated trajectories and delivers it to the regenerated trajectory, which can provide a more general context of the human skill other than a trajectory.

In situations where both the target forcing term and the dynamic features are unknown in the system, achieving one from another has been explored by researchers [76–78], but adapting both simultaneously remains a challenge. The methodology suggested by Y. Cohen et al. [69] to adapt DMP parameters in the parameter manifold can give a clue to the DFE. They constructed the DMP over the parameter manifold to calculate the task result (landing position) of softball throwing and performed Principal Component Analysis (PCA) and Locally Linear Embedding (LLE) to adapt the throwing angles and position n of the robot, which successfully shows the successful throwing motion in the experiment. Although they only adapted the goal position of the motion, the idea of parameterizing an objective function in the meta-parametric space can be implemented for the DFE.

In this chapter, a novel approach is proposed to extract dynamic features from human trajectories, which can be effectively utilized in both LfD and RL with DMP frameworks. The suggested extraction methodology involves comparing demonstrated trajectories and regenerated trajectories in the meta-parametric space of M, D, K to identify optimal dynamic features. These dynamic features enable DMP to regenerate trajectories stably while preserving their similarity to the original demonstrations. The rest of the chapter is organized as follows: Section 4.2 reformulates DMP and defines the problem addressed in this chapter. Section 4.3 presents the reasoning and methodology used to define the objective function to optimize and extract the relevant parameters. Section 4.4 outlines the

experimental setup and presents the results of the Dynamic Feature Extraction (DFE) process. Section 4.5 showcases the outcomes of the simulation study and how the proposed approach adapts to a robot.

4.2 Problem Definition

DMP is a category of motion primitive, a virtual dynamic model used to regulate the behavior of robotic systems subject to certain regulations or specifications. The inherent smoothness of DMP allows the robot to efficiently imitate damped human motions. The simplest DMP model defines a point-attractor system which is represented as [36]

$$\tau_y \ddot{y}_t = \alpha(\beta(y_g - y_t) - \dot{y}_t) + f(x_t), \tag{4.1a}$$

$$\tau_x \dot{x}_t = x_t, \tag{4.1b}$$

where y_t, $\dot{y}_t$, $\ddot{y}_t \in \mathbb{R}$ are respectively the position, velocity, and acceleration of the desired trajectory of the point with initial states $y_0 = 0$, $\dot{y}_0 = 0$, $y_g \in \mathbb{R}$ is the goal position which the point is supposed to reach, $x_t \in \mathbb{R}$ is the canonical state of the DMP model with an initial condition $x_0 = 1$, $\alpha, \beta \in \mathbb{R}^+$ are gain parameters, $\tau_y, \tau_x \in \mathbb{R}^+$ are the temporal scaling terms of the point dynamics (4.1a) and the canonical dynamics (4.1b), $f : \mathbb{R} \to \mathbb{R}$ is a perturbing forcing term that shapes the DMP, defined as

$$f(x_t) = \frac{\sum_{i=1}^{m} w_i \psi_i(x_t)}{\sum_{i=1}^{m} \psi_i(x_t)} x_t (y_g - y_0) \tag{4.2}$$

where, for $i = 1, 2, \cdots, m$, $w_i \in \mathbb{R}$ are real weights and $\psi_i : \mathbb{R} \to \mathbb{R}$ are a cluster of radical-basis functions (RBFs) defined as

$$\psi_i = \exp\left(-\frac{1}{2\sigma_i}(x_t - c_i)^2\right), \tag{4.3}$$

where $\sigma_i, c_i \in \mathbb{R}$ are parameters. DMP can be used to generate robot trajectories in both the joint space and the task space. In the joint space, each joint trajectory generates separate DMP forcing terms. In the task space, the Cartesian trajectories generate three DMP forcing terms, one each in the x, y, and z-axis. In this paper, the trajectory $y_t, \dot{y}_t, \ddot{y}_t$ are discussed and analyzed in the task space excluding its orientation.

The DMP model in (4.1) can be represented as the following spring-damper dynamic system

$$M\ddot{y}_t + D\dot{y}_t + K(y_t - y_g) = f, \tag{4.4}$$

where $M = \tau_y$, $D = \alpha$, and $K = \alpha\beta$ respectively denote the inertia, the damping, and the stiffness of the dynamic system. The forcing term f is approximated by the linear combination of the weighted basis functions with different distributions. Generally, locally weighted regression is used to train the weights, also known as the shape parameters. Once the DMP is trained, the weights can be reused to generate other trajectories similar to the original demonstration with different goals. In the typical DMP modelling, the internal gains α, β, τ (or dynamic features M, D, K) are assumed to have specific constant values and used to achieve the weights w_i.

Most of the DMP models with both LfD and RL assume the fine-tuned dynamic features M, D, K. However, the manual tuning of the dynamic features limits the DMP's ability to regenerate trajectories autonomously. One approach for parameter tuning is to assume a critically damped system [16] with the damping ratio:

$$\zeta = \frac{D}{2\sqrt{KM}} = 1. \tag{4.5}$$

However, relying solely on the critical damping condition for system stability may not capture the desired spring-damping characteristic of the demonstrated motion accurately. To extract the appropriate dynamic features that represent the human intention behind the

motion, optimization of a specific objective function in the meta-parametric space of M, D, and K is necessary. This objective function should take into account two key considerations:

- **Human-Likeness Criterion**: The regenerated trajectory, when provided with the same initial and goal points in the LfD with DMP framework, should closely resemble the demonstrated trajectory. Maximizing the similarity (or minimizing the distance error) between the original demonstration and the regenerated trajectory is crucial for capturing the human-likeness aspect.

- **Regeneration Stability Criterion**: The similarity of trajectories in the task space must lead to a similarity in the forcing terms (or weights) within the latent space of DMP. This criterion aligns with the fundamental assumption of the DMP framework, where similar trajectories correspond to similar forcing terms. By ensuring the regeneration stability criterion, the DMP framework can reliably reproduce the desired motion patterns.

By formulating and optimizing the objective function with these criteria in mind, it becomes possible to extract dynamic features that accurately represent the human intention behind the motion, allowing for improved skill transfer and autonomous adaptation in robot training scenarios.

4.3 Methodology

DFE from demonstrated trajectories can be considered as an optimization problem to find optimal M, D, K that minimize a certain objective function. To achieve stable motion generation that reflects human intention, the objective function of dynamic features in LfD should consider two potentially conflicting goals: human likeness and motion regeneration stability.

Human-Likeness Criterion

According to the fundamental assumption of DMP, trajectories mapped from the same forcing term in the latent space are considered similar in the task space, regardless of their goal positions and time durations. Therefore, in the framework of LfD with DMP, if the distance between the original demonstration and a regenerated trajectory is minimal, we can infer that other regenerated trajectories with different goal positions and time durations will also be similar to the original demonstration. The distance in the task space can be computed by measuring the accumulated Euclidean distance error between the demonstrated trajectory and the regenerated trajectory.

At first, the state equation of regenerated trajectory can be formulated as

$$\begin{aligned} \ddot{y}_{new} =& -\frac{D}{M}\dot{y}_{demo} - \frac{K}{M}(y_{demo} - g_{demo}) + \frac{\sum \psi_i w_i (y_{demo}(0) - g_{demo})}{\sum \psi_i M} x, \\ \dot{y}_{new} =& \ddot{y}_{new}\Delta t, \\ y_{new} =& \dot{y}_{new}\Delta t, \end{aligned} \tag{4.6}$$

where $y_{demo}(0)$ is the initial position and g_{demo} is the goal position of the demonstrated trajectory, $y_{new}, \dot{y}_{new}, \ddot{y}_{new}$ is the regenerated trajectory from the demonstrated trajectory $y_{demo}, \dot{y}_{demo}, \ddot{y}_{demo}$. As M is equivalent to temporal scaling factor τ_y, different M changes the temporal size of the DMP forcing term by changing the size of Δt in (4.6). M should be set equal to the time duration T_{demo} which scales the temporal size of the state variables of the regenerated trajectory, in order to enable the comparison of the trajectories in the same unit time duration. It is also important to note that the optimal D and K depend on the desired duration of the motion or the time scaling factor M, and that D and K should increase linearly with M. Therefore, it is concluded that the extracted dynamic features are not a single set but the ratios of M, D, K. The parametric dimension of the objective function can be decreased to 2-dimension using $D_M = \frac{D}{M}$ and $K_M = \frac{K}{M}$, where

D_M, K_M are D, M where $M = \tau$ has a unit time duration. This means that the similarity of trajectories in the task space is taken into account by temporally normalizing them to have the same duration.

Also, the spatial scaling term size of $(y_{demo}(0) - g_{demo})$ has the same size as the original demo trajectory, thus the accumulated distance error between the demonstrated trajectory and the regenerated trajectory can be as the time integral of the Euclidean distance between the demonstrated trajectory interpolated to unit time and the regenerated trajectory for each time step. Therefore, it is concluded that the human-likeness term of the objective function should be

$$d(D_M, K_M) = \int_0^1 ||y_{new}(D_M, K_M, t) - y'_{demo}(t)||\, dt, \tag{4.7}$$

$$y'_{demo} = y_{demo}(T_{demo}t) - y_{demo}(0), \tag{4.8}$$

where d is the accumulated distance error as the human-likeness term of the objective function, y'_{demo} is temporally normalized demonstrated trajectory with 1 sec duration and T_{demmo} is the duration of the demonstrated trajectory. Due to the noise in acceleration and velocity, only the position profiles are compared in the human-likeness term.

Regeneration Stability Criterion

Another important term to include in the objective function should relate to the stable representability of the demonstration trajectory. One of the primary assumptions of the DMP framework is that the similarity of the motions in the task space is conserved to the similarity of forcing terms (or the weights) in the DMP space [79]. However, this assumption may be invalid when an improper set of dynamic features are chosen, where the dynamic system cannot represent the motion robustly thereby causing unstable performance. In this sense, the similarity between the target forcing terms of multiple demonstrated trajectories

can be measured over the parametric space of dynamic features M, D, K. While the DMP framework excels in one-shot learning with a single trajectory, analyzing multiple demonstrated trajectories can identify the overlaps of the human skills among these trajectories stably as a trade-off.

The similarity in the DMP framework between the trajectories can be defined as the Standard Deviation (SD) of the trajectories' forcing terms. For example, if the trajectories are ideally similar and all from a set of regenerated trajectories sharing the same forcing term with optimal dynamic features, the SD of the forcing terms of the trajectories will be zero. On the other hand, if the dynamic system doesn't represent the original system because of an improper set of dynamic features, it may lose its capability to remain the similarity from the task space to the DMP space, thus the constructed forcing terms will be different in the DMP space even though the trajectories are similar in the task space.

Calculating the SD of the forcing terms in their original scale can cause an issue in the optimization. Since the SD becomes 0 where M, D, K are all 0, it will only give a trivial solution as a global minimum. Instead, the topological similarity of the forcing terms can be considered by standardization of the forcing terms. In the same sense that y_{demo} is temporally normalized in the accumulated distance error, f can be temporally normalized by dividing the term by M, and spatially standardized as:

$$f_{target}(i,t) = \ddot{y}_{demo}(i,t) + D_M \dot{y}_{demo}(i,t) + K_M(y_{demo}(i,t) - g_{demo}(i)), \tag{4.9}$$

$$f_{stand}(i,t) = \frac{f_{target}(i,t) - \overline{f_{target}(i)}}{\sigma(f_{target}(i,t))}, \tag{4.10}$$

where $f_{target}(i,t)$, is the target forcing term from ith demonstration with a unit duration before the approximation with basis functions, $f_{stand}(i,t)$ is the standardized target forcing term, $\overline{f_{target}(i,t)}$ and $\sigma(f_{target}(i,t))$ are the mean and the SD of the ith target forcing term over time. Then, the SD of $f_{stand,i}$ across the different demonstrations is calculated for each

time step:

$$\overline{f_{stand}(t)} = \frac{1}{N}\sum_{i=1}^{N} f_{stand}(i,t), \tag{4.11}$$

$$\sigma_f(t) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(f_{stand}(i,t) - \overline{f_{stand}(t)})^2}, \tag{4.12}$$

where N is the number of the trajectories, $\overline{f_{stand}(t)}$ and $\sigma_f(t)$ are the mean and SD of $f_{stand}(i,t)$ across the trajectories at certain time step. As there are three Cartesian DMP in x, y, z direction, $\sigma_{f_x}(t)$, $\sigma_{f_y}(t)$, $\sigma_{f_z}(t)$ can be obtained separately.

Finally, the topological similarity term of the objective function can be achieved by taking the time integral of the norm of $\sigma_{f_x}(t)$, $\sigma_{f_y}(t)$, $\sigma_{f_z}(t)$:

$$S = \int \sqrt{\sigma_{f_x}(t)^2 + \sigma_{f_y}(t)^2 + \sigma_{f_z}(t)^2}dt, \tag{4.13}$$

where S is the topological similarity. S is the function of D_M and K_M. The suggested similarity function S is designed to identify the optimal dynamic features where the forcing terms of the multiple demonstrated trajectories are similar.

In conclusion, the optimal solution of the dynamic features can be obtained by optimizing the objective function:

$$J_{obj}(D_M, K_M) = S(D_M, K_M) + k\sum_{i=1}^{N} d_i(D_M, K_M), \tag{4.14}$$

where J_{obj} is the objective function, S is the similarity between trajectories to check the stability of the DMP framework and d_i is the ith accumulated distance error for each demonstration, N is the number of the demonstration, k is the adjustment gain for the objective function to balance the impact of each term. The result of the optimization gives D and K where M is constant.

4.4 Experiment

The suggested skill transfer process, depicted in Fig. 4.1, involves multiple human demonstrations to create the forcing terms in the DMP space. Dynamic features are optimized using the suggested methodology. After extraction, these features can be used for parameter tuning in the LfD with the DMP framework or applied to the RL with DMP framework to transfer human skills.

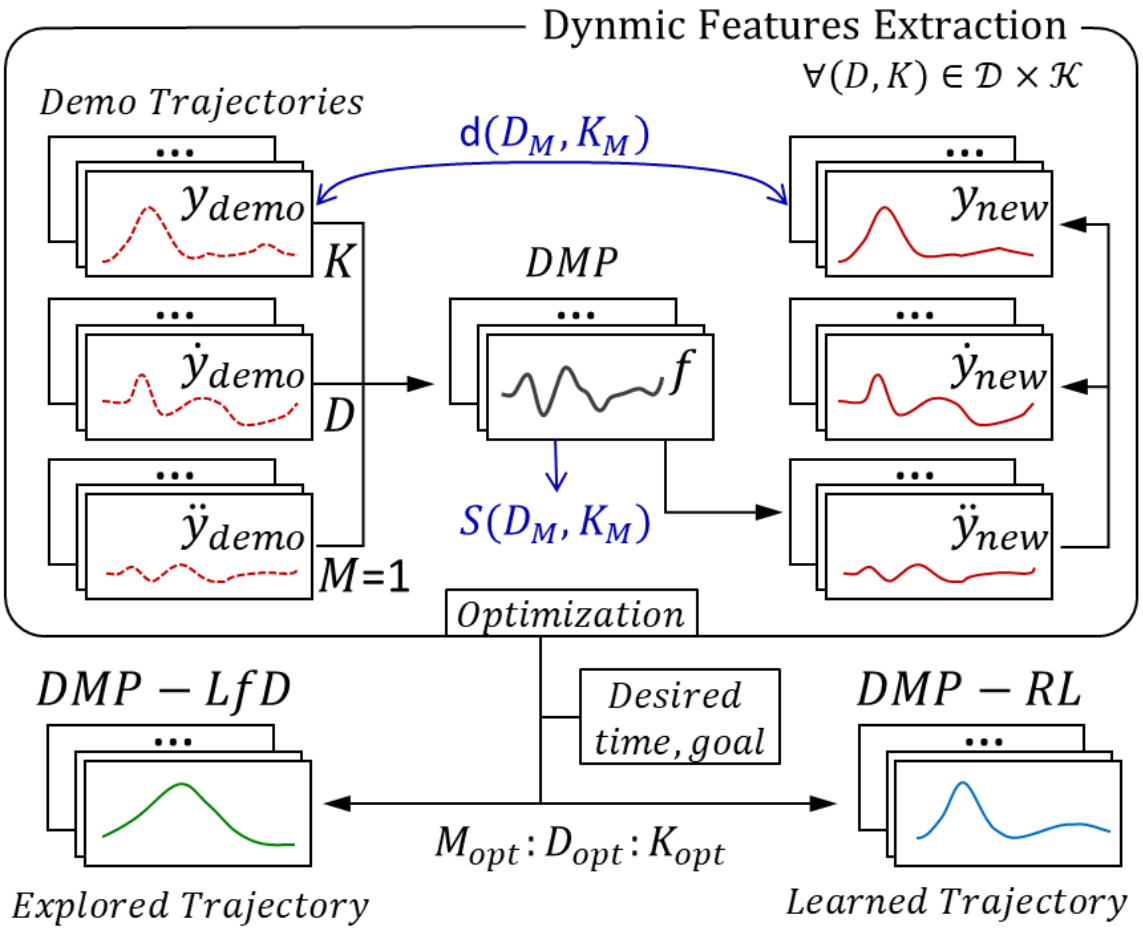


Figure 4.1: Flow of DFE and implementation to LfD and RL with DMP. $M_{opt}, D_{opt}, K_{opt}$ are the extracted dynamic features achieved by the optimization of human-likeness term d and motion stability term S.

Experimental setup

In order to extract dynamic features, 10 demonstration trajectories are randomly selected from the dataset created in Chapter 3.

When constructing the forcing term f in the methodology, it is required to reduce the noise in the velocity and the acceleration in the demonstrated trajectories, as they are achieved from the time derivatives of position data. The Savitzky-Golay filter with the 3rd-order polynomial is applied, and the window length is set to 21. The number of the BFs used in DMP is set to 100 in the experiment.

Dynamic Feature Extraction (DFE)

To observe the contribution of D_M and K_M to the two terms in the objective function, the accumulated distance error and the topological similarity (Fig. 4.2) on the parametric space of D_M and K_M axes are plotted as below. The ranges of D_M and K_M are bounded to be positive.

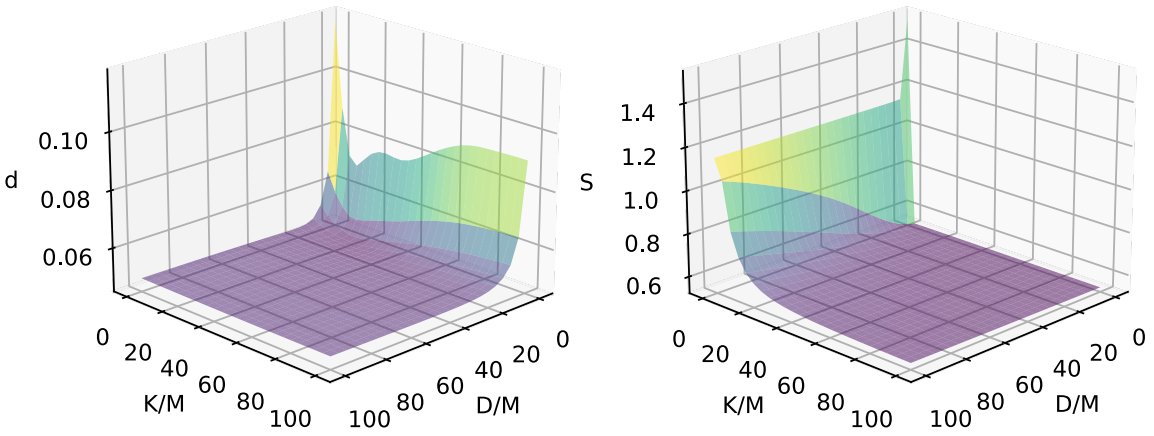


Figure 4.2: (a) The total accumulated distance error $\sum d$ of the demo trajectories and (b) their topological similarity S

As shown in the accumulated distance error, too small spring and damping coefficients D and K compared to the inertia M of a dynamic system hinder the ability to regenerate

the motion close to the original demonstration. In contrast, the similarity map of S shows that when K is too smaller than M in the dynamic system, the DMP cannot stably represent the common skill in the demonstrated trajectories even though they are similar in the task space. Also, it is observed that the higher D_M and K_M give slightly higher d and S in the plain region of the graph. Thus, the objective function including two terms can find the balance between the two goals represented by two terms of the objective function. The adjustment gain k introduced in the objective function (Eq. (4.14)) is set to 20, and the extracted ratio of the dynamic features is

$$(M : D : K)_{extracted} = 1.00 : 10.73 : 20.71 \tag{4.15}$$

with the damping ratio $\zeta = 1.28$, resulting in an over-damped dynamic system in the DMP.

Implementation to DMP frameworks

In the LfD with DMP setup, one of the demonstrated trajectories is picked to learn its motion using the extracted dynamic features. The time duration of the regenerated trajectory is set to 1 sec to compare with the temporally normalized demo trajectory. Also, the goal position is set same as the original goal position.

In the RL with DMP setup, the agent's action substitutes the forcing term with the basis functions. The action is used to calculate the next acceleration, velocity and position of the trajectory. During the episode, the reward is given as

$$r = -10^{-3}|\ddot{y}| - 10^{-7}|A|, \tag{4.16}$$

and at the end of the episode,

$$r = -10 * |y - g_{new}| \tag{4.17}$$

where A is an action, g_{new} is the goal position of the new trajectory that is the same as g_{demo}. The time duration of the new trajectory is set to 1 second. Deep Deterministic Policy Gradient (DDPG) algorithm introduced in 2.1 is used with the hidden size (64,64) for both actor and critic layers.

4.5 Results

Simulation Results

The extracted dynamic features are applied to regenerate trajectories using LfD with DMP, with the same goal position as the demonstrated trajectory. Also, the heuristic D_M, K_M in Table 4.1 are used to regenerate other trajectories for comparison. Demo 1 and Demo 2 represent randomly selected trajectories from the dataset, which are utilized for learning DMP.

Table 4.1: Performance comparison of the extracted (DFE) and the heuristic (Hrstc1 to Hrstc4) dynamic features.

				Demo1		**Demo2**	
Method	**D/M**	**K/M**	ζ	d_{mean}	a_{peak}	d_{mean}	a_{peak}
DFE	10.73	20.71	1.18	3.58mm	13.35m/s^2	7.54mm	8.64m/s^2
Hrstc1	25.00	156.25	1.00	6.24mm	19.69m/s^2	7.84mm	8.64m/s^2
Hrstc2	10.00	200.00	0.35	6.88mm	13.91m/s^2	8.75mm	8.68m/s^2
Hrstc3	100.00	20.00	11.18	5.72mm	48.95m/s^2	7.61mm	8.44m/s^2
Hrstc4	4.00	4.00	1.00	16.82mm	10.59m/s^2	7.51mm	8.51m/s^2

Fig. 4.4 and Fig. 4.3 display the regenerated trajectories, learned from Demo 1 and Demo 2 Both in Demo 1 and Demo 2, the regenerated trajectories made with the extracted dynamic features follows the demonstrated trajectories as well as the fine-tuned Hueristic1. In contrast, Heuristic2 with the high K_M and the low damping ratio ζ shows too much reactive motion to the small pause of the demonstration near goal. Hueristic3 with the high D_M and the high ζ is vulnerable to the acceleration noise in the original trajectory showing

the initial overshoot in its acceleration. Heuristic4 fails to reach to the goal position even though ζ is selected to 1 following the suggestion in [16]. Table 4.1 shows the mean distance error d_{mean} which is the accumulated distance error d_{mean} divided by the time duration of the motion, and the peak acceleration a_{peak} of each generated trajectory.

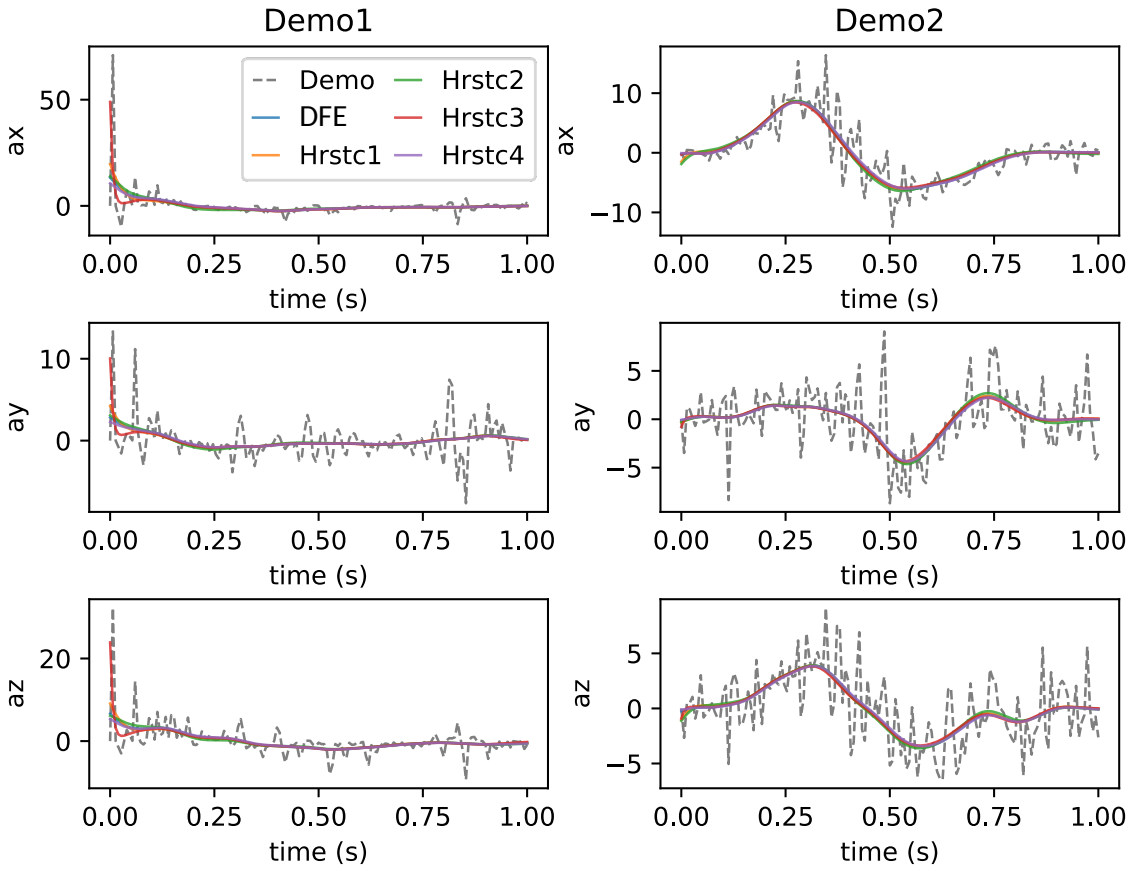


Figure 4.3: The accelerations of regenerated trajectories using the extracted dynamic feature and the heuristic features of the LfD with DMP framework.

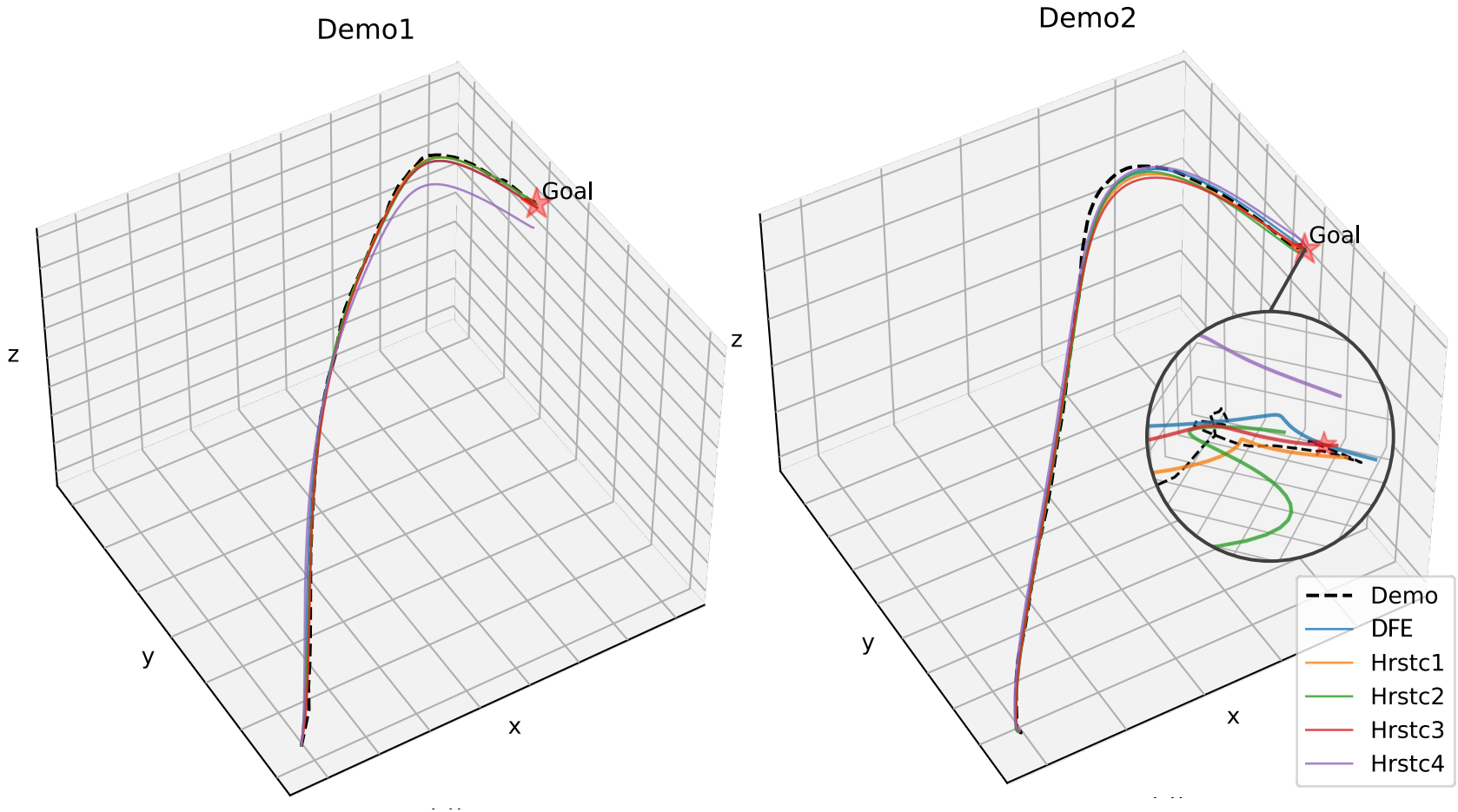


Figure 4.4: The path of regenerated trajectories using the extracted dynamic feature and the heuristic features in the LfD with DMP framework.

Additionally, the dynamic features are applied to RL with DMP. Fig. 4.5 shows that trajectory regeneration is more sensitive to the dynamic feature selection compared to LfD with DMP. While both the extracted dynamic features and fine-tuned Heuristic1 are able to reach the goal position robustly, the regenerated trajectories using other heuristic features either fail to reach the goal position entirely or produce ineffective trajectories.

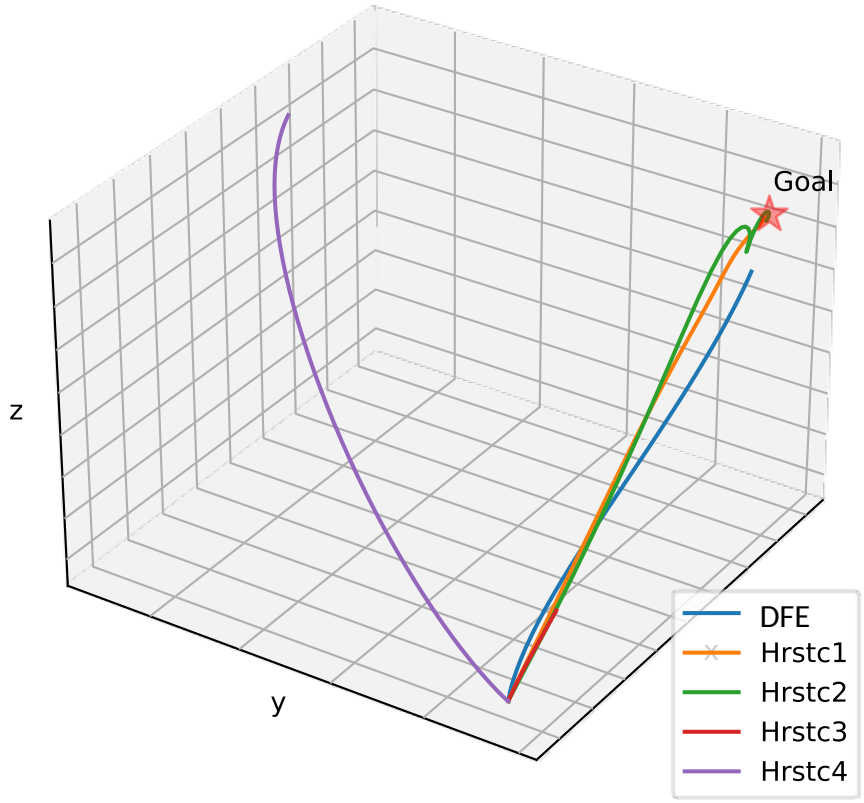


Figure 4.5: The path of regenerated trajectories using the extracted dynamic feature and the heuristic features in the RL with DMP framework.

Experimental Results

The trajectory is also executed by a Kinova® Gen,3 manipulator equipped with a two-finger Robotiq® 2F-85 gripper, as shown in Fig. 4.6. In the given task of carrying a cup full of coffee, the robot successfully completed the task without spilling the liquid when the extracted dynamic features are applied, as shown in Fig. 4.6c and Fig. 4.6d. However,

the trajectories generated by an RL agent using heuristic values, except for the fine-tuned Heristic1, failed to complete the task, as shown in Fig. 4.6a and Fig. 4.6b. This demonstrates that the dynamic features accurately represent the characteristics of dynamic motion and that the proposed method enables the robot to manipulate objects delicately, performing as well as the fine-tuned approach.

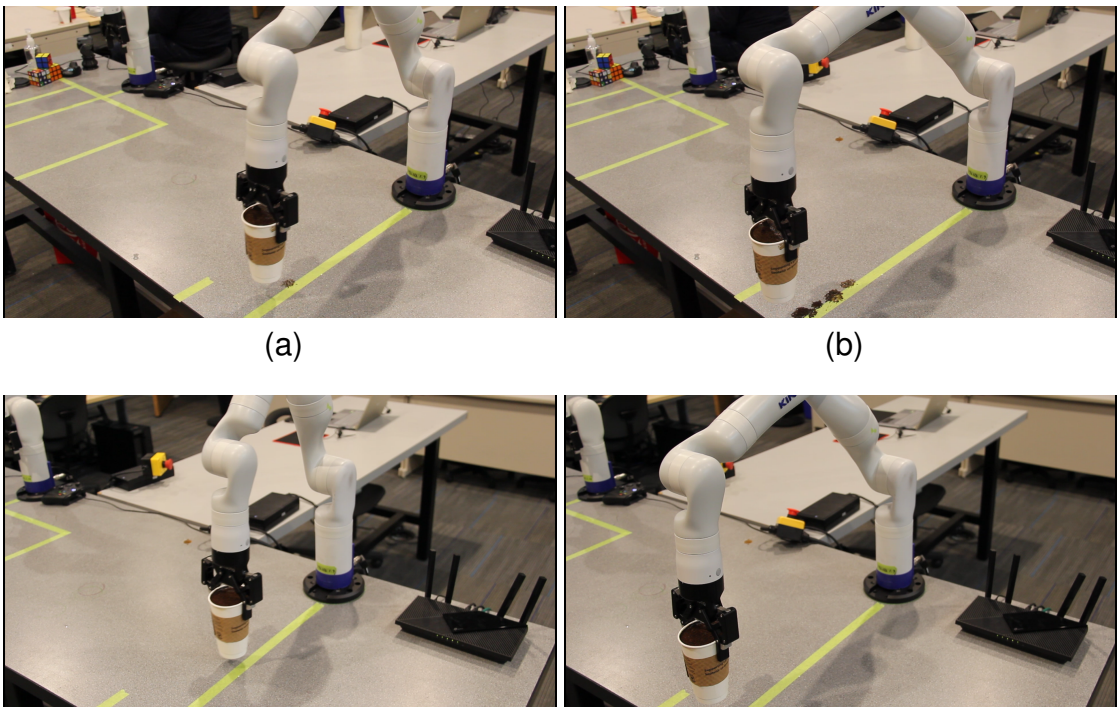


Figure 4.6: The experiment involved a real robot attempting to carry a cup of coffee. In (a) and (b), we see the mid and end positions of the robot when using Heuristic3 parameters. The generated motion failed to carry the cup without spilling coffee. However, in contrast, (c) and (d) show the mid and end positions of the robot when utilizing extracted dynamic features. Remarkably, the generated motion successfully completed the task without any coffee spills.

Chapter 5

RL-based Robot Motion Planning Enhanced with IBC and DMP

This chapter shows another approach to transfer human skills to a robot in a different use case. While Chapter 4 explores a new skill to transfer from a human to a robot other than the explicit trajectory, Chapter 5 suggest an approach to directly utilize the human data in the training process of Reinforcement Learning (RL) by the integration of Implicit Behaviour Cloning (IBC) and Dynamic Movement Primitives (DMP) framework.

5.1 Related Work

The main advantage of an off-policy RL agent, such as Deep Deterministic Policy Gradient (DDPG), is the exploitation of historical policies. This allows the exploitation of diverse policies for value promotion, which improves the exploration capability of the agent. Similar techniques are also used to propose multi-gradient-based approaches [80]. Nevertheless, the lack of sample labels leads to a *bootstrapping* process for the training of the critic network. The bootstrapping process makes the training performance very sensitive to the initial conditions, leading to low learning stability. In this chapter, a novel method

is suggested to improve the training performance of the off-policy agents using demonstration data. By properly embedding the demonstration policy in the loss functions, both the stability and convergence speed of the agent training are improved.

In RL, a common approach is to use Behaviour Cloning (BC) to obtain a decent human-like policy which then serves as the initial policy for the agent training [60–62]. The human-like policy is not necessarily to be the optimal one but is at least a decent policy for robot motion generation. This method, however, renders complete separation between BC and agent training, such that BC is not helpful in improving the performance of RL. A recent study proposed a novel method to integrate BC into the training process of an RL agent, which greatly improves the convergence speed [63]. However, the demonstration used for BC is generated by a PID controller in a simulation environment, instead of real human data. Also, BC is still performed in a *explicit* manner which directly penalizes the deviation between the cloned and the demonstration actions. This may lead to the overfitting of the demonstration policy. Recent work tries to solve this problem by proposing an *implicit* BC (IBC) method that performs BC by penalizing a certain energy function of the cloned policy, such that the cloned policy is less sensitive to the action deviations [65].

In this chapter, it is claimed that the proper usage of both heuristics and human demonstration can leverage the training speed and the generalizability of RL agents. Out of this motivation, a novel RL method for robot motion planning facilitated by DMP and IBC is proposed, which has not been investigated by existing work. The efficacy of the proposed method and its advantages over conventional RL agents are validated using simulation studies. Also, an experimental study is conducted to demonstrate its applicability to practical robotic tasks.

5.2 Framework and Problem Statement

In this section, the overall framework of the proposed IBC-DMP RL method is first introduced. Then, a novel Multi-DoF DMP model for motion planning of a high degree-of-freedom (DoF) robot is presented.

Overall Framework

The overall framework of IBC-DMP RL is illustrated in Fig. 5.1. The basic model used to generate robot trajectories is *Multi-DoF DMP*, an adapted version of the conventional DMP model introduced in Section 2.2.2. An *IBC-DMP agent* has a dual-buffer structure, where the experience replay buffer is composed of a *demonstration buffer* and an *interaction buffer*. The interaction between the recorded *human demonstration* and the *Multi-DoF DMP* generates the state and action data to be stored in the *demo buffer*. Besides, the interaction data between the *IBC-DMP agent* and the *Multi-DoF DMP* are stored in the interaction buffer. Then, the IBC-DMP agent is trained using the data stored in the experience replay buffer.

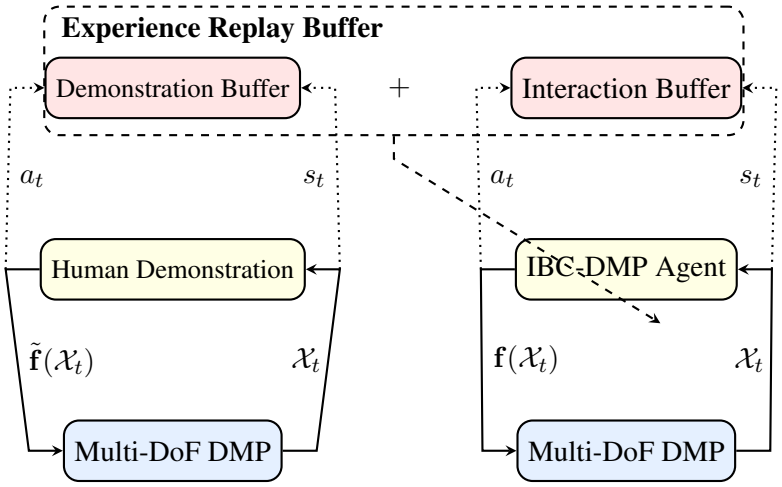


Figure 5.1: The illustration of the IBC-DMP RL framework, where $\mathcal{X}_t$ denotes the position, velocity, and acceleration of the Multi-DoF DMP model, $\tilde{\mathbf{f}}(\mathcal{X}_t)$ and $\mathbf{f}(\mathcal{X}_t)$ are the actuation functions provided by the human demonstration and the IBC-DMP agent, respectively, and s_t and a_t are the state and action data provided by human demonstration or generated by the IBC-DMP agent. Besides, the solid arrows denote the interactions, the dotted arrows indicate data storing, and the dashed arrow represents agent training.

Motion Planning Using Multi-DoF DMP

The conventional DMP in Section 2.2.2 is defined for a one-dimensional point and has been used for robot motion planning in a decoupled manner, i.e., one DMP is used to generate the trajectory for each DoF of the robot [36]. This leads to the lack of coupling among different DoFs, which restricts the flexibility of solving the optimal planning policy. To resolve this issue, in this chapter, the following Multi-DoF DMP model for the motion planning of a robot end-effector in the Cartesian space is proposed,

$$\tau\ddot{\mathbf{x}}_t = K_\alpha(K_\beta(\mathbf{x}_\mathrm{g}-\mathbf{x}_t)-\dot{\mathbf{x}}_t)+\zeta_t\|\mathbf{x}_\mathrm{g}-\mathbf{x}_0\|\mathbf{f}(\mathcal{X}_t), \tag{5.1}$$

where $\mathbf{x}_t$, $\dot{\mathbf{x}}_t$, $\ddot{\mathbf{x}}_t \in \mathbb{R}^3$ are the position, linear velocity, and acceleration of the end-effector in the Cartesian space, $\mathbf{x}_0$, $\mathbf{x}_\mathrm{g} \in \mathbb{R}^3$ are the initial and goal positions of the end-effector, K_α, $K_\beta \in \mathbb{R}^{3\times3}$ are parametric matrices, $\tau \in \mathbb{R}$ and $\zeta_t \in \mathbb{R}$ are the temporal scalar and the canonical variable, the same as (2.12), $\mathcal{X}_t = \{\mathbf{x}_t, \dot{\mathbf{x}}_t, \mathbf{x}_\mathrm{b}, \zeta_t\}$ is the state vector of the Multi-DoF DMP, where $\mathbf{x}_\mathrm{b}$ is a vector that describes the configuration of the obstacle, and $\mathbf{f} : \mathbb{R}^{10} \to \mathbb{R}^3$ is a multi-dimension actuation function to be determined. In this chapter, only a *single static obstacle* is considered for brevity. Also, a vertically positioned *cylinder* model is used to represent the obstacle. Then, the position of the obstacle is represented as a three-dimensional constant vector $\mathbf{x}_\mathrm{b} \in \mathbb{R}^3$ assigned as the Cartesian coordinate of the top surface center of the cylinder. In this sense, the DMP state can be represented as a ten-dimensional vector $\mathcal{X}_t = [\mathbf{x}_t^\top, \dot{\mathbf{x}}_t^\top, \mathbf{x}_t^\top - \mathbf{x}_\mathrm{b}^\top, \zeta_t]^\top \in \mathbb{R}^{10}$. Here, a time-variant vector $\mathbf{x}_t - \mathbf{x}_\mathrm{b}$ into the DMP state is encoded instead of the constant vector $\mathbf{x}_\mathrm{b}$ since the former provides larger diversity to the data. The incorporation of more complicated environments with multiple dynamic obstacles is beyond the scope of this paper and will be considered in future work. The multi-DoF DMP is different from the conventional DMP (2.12) in three aspects.

- The actuation function $\mathbf{f}$ does not only depend on the canonical variable ζ_t but also on the internal states of the DMP, namely $\mathbf{x}_t$ and $\dot{\mathbf{x}}_t$, and the obstacle position $\mathbf{x}_\mathrm{b}$. This helps develop a flexible policy $\mathbf{f}$ that fully incorporates the states of DMP.

- The gain of the actuation function is the absolute distance between the initial position $\mathbf{x}_0$ and the goal position $\mathbf{x}_\mathrm{g}$, $\|\mathbf{x}_\mathrm{g} - \mathbf{x}_0\|$, instead of the element-wise distance used by the conventional DMP (2.12). Such a scheme can improve the flexibility of a DMP model by fully incorporating the coupling of its different dimensions.

- The obstacle position $\mathbf{x}_\mathrm{b}$ is included in the state $\mathcal{X}_t$ of the DMP, instead of being incorporated as an additional virtual force term as the conventional DMP model. This

provides the possibility of incorporating more complicated obstacle information into motion planning.

Cost Function of Motion Planning

This chapter solves a general robot motion planning problem. Specifically, given an initial position $\mathbf{x}_0 \in \mathbb{R}^3$ and a goal position $\mathbf{x}_{\rm g} \in \mathbb{R}^3$ in the Cartesian space, generate a smooth trajectory $\mathbf{x}_t \in \mathbb{R}^3$ for $0 < t \leq T$ using the Multi-DoF DMP model in (5.1), such that $\mathbf{x}_T$ is sufficiently close to $\mathbf{x}_{\rm g}$, where $T \in \mathbb{R}^+$ is the predefined timing length of the trajectory. The generated trajectory should also ensure sufficient smoothness (minimal acceleration) and maintain a certain distance with a given static obstacle positioned in $\mathbf{x}_{\rm b} \in \mathbb{R}^2$. These requirements are encoded in the following cost function,

$$\mathcal{J}_t = \begin{cases} \sum_{i=1}^{4} \alpha_i \mathcal{J}_t^{(i)}, & 0 \leq t < T, \\ \alpha_5 \mathcal{J}_t^{(5)}, & t = T, \end{cases} \tag{5.2}$$

where $\alpha_i \in \mathbb{R}^+$ are constant parameters, $i = 1, 2, \cdots, 5$, and $\mathcal{J}_t^{(i)} \in \mathbb{R}$ are instant costs at time t, defined as

$$\begin{aligned} &\mathcal{J}_t^{(1)} = \|\ddot{\mathbf{x}}_t\|^2, \ \mathcal{J}_t^{(2)} = \|\mathbf{x}_t - \mathbf{x}_{\rm g}\|^2, \ \mathcal{J}_t^{(3)} = \eta\Big(\mathbf{x}_t^{(3)}\Big), \\ &\mathcal{J}_t^{(4)} = \eta\Big(\Big\|\mathbf{x}_t^{(1,2)} - \mathbf{x}_{\rm b}^{(1,2)}\Big\| - r_{\rm b}\Big), \text{ if } 0 < \mathbf{x}_t^{(3)} < \mathbf{x}_{\rm b}^{(3)}, \\ &\mathcal{J}_t^{(5)} = \xi(\|\mathbf{x}_t - \mathbf{x}_{\rm g}\|), \end{aligned}$$

where $\mathbf{x}_t^{(1,2)} \in \mathbb{R}^2$, $\mathbf{x}_t^{(3)} \in \mathbb{R}$, $\mathbf{x}_{\rm b}^{(1,2)} \in \mathbb{R}^2$, $\mathbf{x}_{\rm b}^{(3)} \in \mathbb{R}$ are the slides of vectors $\mathbf{x}_t$ and $\mathbf{x}_{\rm b}$, $r_{\rm b}$ is the radius of the cylinder, $\eta : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a function defined as

$$\eta(x) = \begin{cases} (x - \varepsilon_0)^{-2} - (\varepsilon_1 - \varepsilon_0)^{-2}, & \varepsilon_0 < x < \varepsilon_1, \\ 0, & x \geq \varepsilon_1, \end{cases} \tag{5.3}$$

where $0<\varepsilon_0<\varepsilon_1$ are constant parameters, and $\xi : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is a squared dead-zone function defined as

$$\xi(x) = \begin{cases} 0, & 0 \leq x \leq \varepsilon_z, \\ x - \varepsilon_z, & x > \varepsilon_z, \end{cases} \tag{5.4}$$

where $\varepsilon_z \in \mathbb{R}^+$ is the dead-zone scalar. Here, $\eta(x)$ serves as an artificial potential field function that penalizes x if it gets close to ε_0 from the positive direction. Another parameter ε_1 is the upper limit of x such that $\eta(x)$ has an effect. To avoid undefinition, $\eta(x)$ can be assigned with a large number when $x \leq \varepsilon_0$.

In the comprehensive cost function (5.2), $\mathcal{J}_t^{(1)}$ penalizes the value of the acceleration to ensure sufficient smoothness, $\mathcal{J}_t^{(2)}$ penalizes the distance between the current position $\mathbf{x}_t$ and the goal position $\mathbf{x}_{\mathrm{g}}$, aiming at fast and straightforward goal reaching, $\mathcal{J}_t^{(3)}$ and $\mathcal{J}_t^{(4)}$ attempt to keep the position $\mathbf{x}_t$ away from the ground ($z = 0$ plane) and the cylinder obstacle, respectively, using an artificial potential field method, and $\mathcal{J}_t^{(5)}$, $t = T$, penalizes the distance between the ultimate position $\mathbf{x}_T$ and the goal position $\mathbf{x}_{\mathrm{g}}$. By minimizing the comprehensive cost (5.2) for the Multi-DoF DMP model (5.1), one can get a smooth trajectory $\mathbf{x}_t$ that is able to reach the goal position $\mathbf{x}_{\mathrm{g}}$ at the ending time $t = T$ while avoiding collision with the ground and the obstacle.

Decision-Making Problem Statement

Consider that the Multi-DoF DMP model (5.1) is discretized in time in a zero-order hold manner with a discrete sampling time $\Delta t \in \mathbb{R}^+$, as follows,

$$\begin{aligned} &\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \dot{\mathbf{x}}_t,\ \dot{\mathbf{x}}_{t+\Delta t} = \dot{\mathbf{x}}_t + \Delta t \ddot{\mathbf{x}}_t, \\ &\zeta_{t+\Delta t} = \zeta_t - \frac{\Delta t}{\tau} \omega \zeta_t, \end{aligned} \tag{5.5}$$

where $\ddot{\mathbf{x}}_t$ is given by (5.1). Then, the generated trajectory $\mathbf{x}_t$ is a set of waypoints at the sampled timing points $\mathbf{t} = \{0, \Delta t, 2\Delta t, \cdots, T\}$. In this sense, the objective of motion

planning is to solve the actuation function $\mathbf{f}$ in (5.1) to minimize the accumulated cost $\sum_{t=0}^{T} \mathcal{J}_t$. This problem renders decision-making over an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathscr{F}, \mathscr{R})$, where $\mathcal{S} = \{\mathcal{X}_t | \mathbf{x}_0 \in \mathbb{R}^3,\ 0 \leq t < T\}$ is the state space, $\mathcal{A} = \{\mathbf{f}(s)\,|\forall\, s \in \mathcal{S}\}$ is the action space, $\mathscr{F}$ is the state transition characterized by (5.5), and $\mathscr{R} : -\mathcal{J}_t$ is the instant reward. In this sense, the optimal policy of $\mathcal{M}$ is equivalent to the optimal actuation function $\mathbf{f}$ which can be approximated using a neural network and solved with an off-policy RL method as introduced in Section 2.1. Besides, BC introduced in Section 2.2.1 can be used to promote the training of the RL agent with the experience replay buffer, as addressed in Section 2.2.1. Now, it is ready to state the two main objectives of this chapter: (1). transforming human demonstration into data that are compatible with the demonstration buffer, corresponding to the left column of Fig. 5.1, and (2). train the optimal policy $\mathbf{f}$ using the demonstration buffer, as the right column of Fig. 5.1. The proposed solutions to these two problems will be presented in Section 5.3 and Section 5.4, respectively.

5.3 Data Preprocessing

This section interprets how the collected human demonstration data was transformed into compatible data for the demonstration buffer in the IBC-DMP RL framework.

Velocity Normalization

The diversity of human hand trajectories is reflected by both their shapes and speeds. The shape of a trajectory prescribes in which direction the hand should move to avoid the obstacle, and its speed indicates how fast the hand moves. The diversity of the shapes is beneficial to improve the robustness of the trained planning agent against various obstacles and goal positions. However, the speeds of the hand trajectories do not contribute to the training robustness but bring up disturbances. Thus, a preprocessing procedure is needed to

normalize the trajectory speeds while retaining the diversity of the shapes. In this chapter, the average speed of a trajectory is concerned . Let $\boldsymbol{\tau} = \{0, \Delta\tau, 2\Delta\tau, \cdots, \bar{T}\}$ be the time stamps of a recorded hand trajectory $\mathbf{x}_\tau^H \in \mathbb{R}^3$, $\tau \in \boldsymbol{\tau}$, where $\Delta\tau = 0.01\,\text{s}$ is the sampling time of the VIVE tracking system. Then, the average speed is defined as $L/\bar{T}$ (m/s), where $\bar{T}$ and $L = \sum_{i=1}^{N} \|\mathbf{x}_{i\Delta\tau}^H - \mathbf{x}_{(i-1)\Delta\tau}^H\|_2$ is the total time duration and the total length of the trajectory, respectively.

The recorded trajectories is normalized to a uniform average speed $\bar{V} = 0.1\,\text{m/s}$ by assigning each trajectory a new time stamp series $\boldsymbol{\tau}' = \bar{V}\bar{T}/L \cdot \boldsymbol{\tau}$ which corresponds to a new sampling time $\Delta\tau' = \bar{V}\bar{T}/L \cdot \Delta\tau$. Then, the linear velocity of the trajectory is approximated as $\dot{\mathbf{x}}_{\tau_i'}^H = \dfrac{\mathbf{x}_{\tau_i'}^H - \mathbf{x}_{\tau_{i-1}'}^H}{\Delta\tau'}$, $i = 1, 2, \cdots, N$, and calculate the maximal speed $\hat{V} = \max_i \|\dot{\mathbf{x}}_{\tau_i'}^H\|_2$. Similar to the average speed, the maximal speed is an important index to describe the kinematic property of a trajectory. Analyzing the statistical properties of the total length, the maximal speed, and the average speed of all trajectories allows investigation into how normalization affects their diversities. Table 5.1 presents the mean values (Mean), standard deviations (Std), and extreme deviations (Max-Min). It can be observed that the average speeds of the trajectories have been normalized and the maximal speeds have been reduced. Additionally, the deviation values of the maximal speeds have also decreased, indicating a reduced influence of the diversity of maximal speeds on the recorded trajectories. However, the statistical properties of the total lengths remain unchanged due to the consistency of the trajectory shapes. As a result, the disturbance caused by varying speeds in the trajectories has been minimized while preserving the diversity of shapes.

Table 5.1: The kinematic diversity of the recorded trajectories

Feat.	Before Normalization			After Normalization		
	Mean	Std	Max-Min	Mean	Std	Max-Min
L	0.4524	0.1781	1.1706	0.4524	0.1781	1.1706
$\hat{V}$	1.1653	0.2795	1.6096	0.2009	0.0371	0.5813
L/T	0.5782	0.1332	0.9676	0.1	0	0

Buffer Generation

After the normalization process, the normalized hand trajectories is transformed into demonstration data that are compatible with the experience replay buffer using the Multi-DoF DMP model, as illustrated in the left column of Fig. 5.1. To fit the sampling rate of the DMP model Δt, the time stamp series is used as $\mathbf{t} = \{0, \Delta t, 2\Delta t, \cdots, T\}$ to interpolate the normalized trajectory $\mathbf{x}_\tau^H$, $\tau \in \boldsymbol{\tau}$. The interpolated trajectory is denoted as $\mathbf{x}_t^H$, $t \in \mathbf{t}$.

As addressed in Section 2.1, the data for the experience replay buffer have the format $\{ s_t, a_t, r_t, s'_t, d_t \}$, for a certain time $t \in \mathbb{N}^+$, where the state $s_t = \mathcal{X}_t$ is also the state of the Multi-DoF DMP model as introduced in Section 5.2.The action $a_t = \mathbf{f}(\mathcal{X}_t)$ is the output of the actuation function $\mathbf{f}$ which allows $\mathbf{x}_t$ to fit the Multi-DoF DMP model, $s' = \mathcal{X}_{t+\Delta t}$ is the successive state of s_t under the action a_t, $r_t = -\mathcal{J}_t$ is the instant reward, and d_t is a binary value to determine whether $\mathbf{x}_t$ reaches the goal $\mathbf{x}_\mathrm{g}$.

The determination of action a_t is not trivial since the actuation function $\mathbf{f}$ is not previously known. In conventional work, the action sample $a_t = \mathbf{f}(\mathcal{X}_t)$ for a given state sample $\mathcal{X}_t$ is directly computed by inverting the DMP model (5.1). This requires the acceleration $\ddot{\mathbf{x}}_t$ calculated via a twice-difference operation which brings differential noises to the action samples. As a result, the variance of the samples may be increased, which may disturb the learning process. In this chapter, a PID-based approach is used to generate actions for the state samples. Specifically, by representing the positions and velocities of a human hand trajectory as $\mathbf{x}_t^H, \dot{\mathbf{x}}_t^H \in \mathbb{R}^3$, a Multi-DoF DMP model described by (5.1) and (5.5) is used with the following actuation function to generate a trajectory $\mathbf{x}_t$ that fits the hand trajectory $\mathbf{x}_t^H$,

$$\mathbf{f}(\mathcal{X}_t) = K_\mathrm{P}(\mathbf{x}_t^H - \mathbf{x}_t) + K_\mathrm{D}(\dot{\mathbf{x}}_t^H - \dot{\mathbf{x}}_t), \tag{5.6}$$

with initial and goal conditions $\mathbf{x}_0 = \mathbf{x}_0^H$, $\mathbf{x}_\mathrm{g} = \mathbf{x}_\mathrm{g}^H$, and $\dot{\mathbf{x}}_0 = 0$, where $K_\mathrm{P} = 1500I$ and $K_\mathrm{P} = 40I$ are constant matrices, and $I \in \mathbb{R}^{3\times 3}$ is an identity matrix. The main advantage of the PID-based method is not requiring the acceleration $\ddot{\mathbf{x}}_t$. Thus, it can reduce the noise

introduced to the samples. With proper parameters K_{P}, K_{P}, the generated trajectory $\mathbf{x}_t$ coincides with the recorded hand trajectory $\ddot{\mathbf{x}}_t$ with small errors $\mathbf{x}_t^H - \mathbf{x}_t$, which ensures the efficacy of the action samples.

Having determined the action sample a_t for the state sample s_t, the successive state s_t' can also be calculated using the Multi-DoF DMP model. The reward r_t can be calculated using $-\mathcal{J}_t$ from the reward function (5.2). Then, the termination flag d_t is set to 1 if the time t equals to T or $\|\mathbf{x}_t - \mathbf{x}_{\mathrm{g}}\|_2 \leq 0.01\,\mathrm{m}$, otherwise 0. Using this method, 544 recorded hand trajectories is ultimately transformed into 123171 samples. The parameters of the Multi-DoF DMP model and the cost function $\mathcal{J}_t$ are shown in Table 5.2.

Table 5.2: The Parameters of the Multi-DoF DMP Model and the Cost

Par.	Value	Par.	Value	Par.	Value	Par.	Value
K_α	$10I$	K_β	$1.2I$	τ	0.25	ω	6
α_1	0.001	α_2	10	α_3	0.001	α_4	0.001
α_5	10^5	ε_0	0.05	ε_1	0.08	ε_z	0.01

5.4 Training of the IBC-DMP agent

Having converted the recorded hand trajectories to the demonstration data, the training method of the off-policy RL agent with demonstration-based BC is proposed. The following provides an overview of the training method and specifically interprets two important technical points of the proposed method: actor loss reshaping and critic loss refinement. Finally, the algorithm for agent training is presented.

Overview of the Training Method for IBC-DMP RL

The training process of the IBC-DMP agent is organized as a flow chart illustrated in Fig. 5.2. Similar to a traditional DDPG agent, the IBC-DMP agent consists of four forward neural networks (FNN), namely a source actor network $\pi_\theta : \mathcal{S} \to \mathcal{A}$, a target actor

network $\pi_{\theta'} : \mathcal{S} \rightarrow \mathcal{A}$, a source critic network $Q_w : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and a target critic network $Q_{w'} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, where the subscripts θ, θ', w and w' denote the parameters of these networks. The source actor and critic networks are constructed to approximate the optimal policy and the value function of the RL problem. Meanwhile, the target networks are used to improve the stability of this approximation. Besides, the IBC-DMP agent is equipped with a dual-buffer structure, which is inspired by the previous work on off-policy RL [44, 64]. The *demo buffer* is used to store the demonstration data of the human motion recorded in Chapter 3 and the *replay buffer* serves as a normal experience replay storage unit. The buffers are designed with fixed depths. Both buffers import the demonstration data from the recorded demo trials before the start of the training process. They also provide the batched data to calculate the losses for the FNNs. The training process is summarized as the following five steps.

Demonstration Data Importing

The human demonstration data generated in Section 3 are imported to the demonstration buffer and then shuffled. They are also imported to the interaction buffer to boot up the training process. The importing processes are denoted as solid arrows in Fig. 5.2. This step is executed only once at the very beginning of each training process.

Batch Sampling

During the training process, four data batches are regularly sampled from the two buffers, denoted as $\mathcal{B}^{\pi}_{\mathrm{D}}$, $\mathcal{B}^{Q}_{\mathrm{D}}$, $\mathcal{B}^{\pi}_{\mathrm{I}}$, and $\mathcal{B}^{Q}_{I}$, respectively. The subscript $\mathcal{D}$ indicates that the batches are sampled from the Demo Buffer and $\mathcal{I}$ means from the Replay Buffer. The superscripts π and Q denote that the batches are used to update the neural networks. The data in the batches are used to update the parameters of the actor and critic networks. The sampling is conducted individually and independently to eliminate the dependence between the samples. The sampling process is represented as double dashed arrows in Fig. 5.2. The sam-

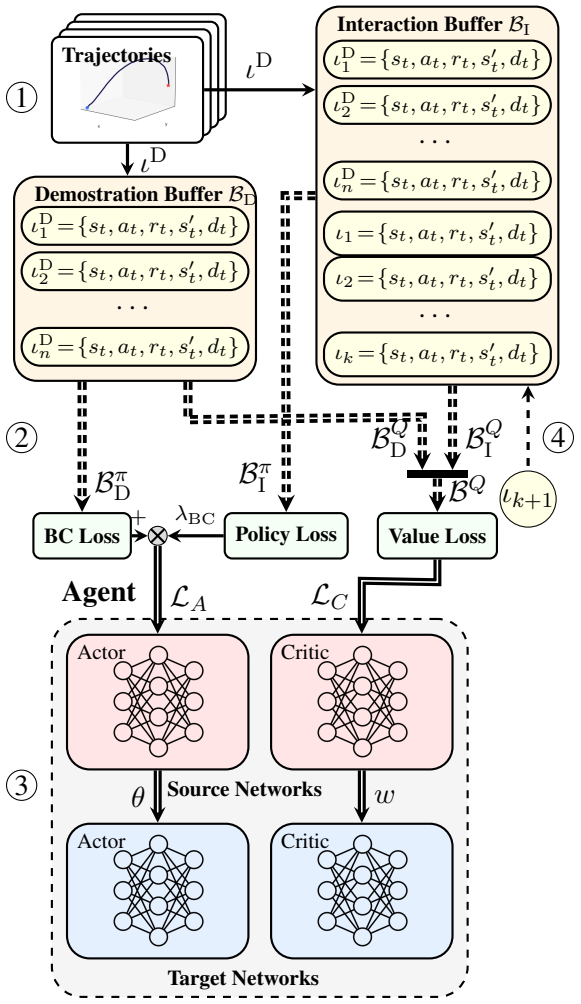


Figure 5.2: The flow chart of the training process of the IBC-DMP agent. The circled numbers indicate the critical steps of the training procedure which are explained in Section 5.4.

pling frequency of the data batches is usually the same as the update rate of the networks.

Network Updates

The sampled data batches are used to calculate the losses of the actor network and the critic network using loss functions $\mathcal{L}_A$ and $\mathcal{L}_C$, respectively. In this chapter, novel loss functions for the training of the neural networks of the IBC-DMP agent is proposed, which will be introduced in Section 2.2.1. With the loss functions $\mathcal{L}_A$ and $\mathcal{L}_C$, the parameters of the source networks are updated using the gradient-based law in (2.9). Then, the target

networks are updated using (2.6). This process is denoted by double arrows in Fig. 5.2.

Experience Storing and Forgetting

Similar to all off-policy RL agents with experience replay, the IBC-DMP agent also stores its interaction data with the environment in the interaction buffer, at every sampling instant. As shown in Fig. 5.2, the latest interaction data is always added to the tail of the interaction buffer, right after the demonstration data. The storing process is represented as a dashed arrow. The interaction buffer has a fixed size such that the old data is forgotten and replaced by the new data. Thus, the demonstration data are ultimately purged from the interaction buffer and lose their impacts on the data batches $\mathcal{B}_{\mathrm{D}}^{Q}$ and $\mathcal{B}_{\mathrm{I}}^{Q}$ as the learning proceeds.

Reshaped Actor Loss Based on IBC

One of the critical technical points of the proposed IBC-DMP agent is the proper design of the loss functions to train the networks. In this chapter, a reshaped loss function for the actor network is proposed as

$$\mathcal{L}_A(\mathcal{B}_{\mathrm{D}}^{\pi}, \mathcal{B}_{\mathrm{I}}^{\pi}) = \hat{\mathcal{L}}_A(\mathcal{B}_{\mathrm{I}}^{\pi}) + \lambda_{\mathrm{BC}} \mathcal{L}_{BC}(\mathcal{B}_{\mathrm{D}}^{\pi}) \,, \tag{5.7}$$

where $\hat{\mathcal{L}}_A(\mathcal{B}_{\mathrm{I}}^{\pi})$ has the same form as a conventional actor loss function defined in (2.8), $\mathcal{L}_{BC}(\mathcal{B}_{\mathrm{D}}^{\pi})$ is a novel IBC loss function and $\lambda_{\mathrm{BC}} \in \mathbb{R}^{+}$ is the BC ratio to adjust the proportion of λ_{BC} in the overall actor loss $\mathcal{L}_A$. For any data buffer $\mathcal{B}$ sized $n \in \mathbb{N}^{+}$, the IBC loss is calculated as

$$\mathcal{L}_{BC}(\mathcal{B}) = -\tfrac{1}{n} \textstyle\sum_{j=1}^{n} E_w(\pi_\theta | s_j, a_j), \tag{5.8}$$

where $E_w(\pi_\theta | s_j, a_j)$ is an energy function of the current policy π_θ defined as

$$E_w(\pi_\theta | s_j, a_j) = -\mathrm{ReLU}(Q_w(s_j, a_j) - Q_w(s_j, \pi_\theta(s_j))) \,, \tag{5.9}$$

where $\{s_j, a_j\}$ are the j-th state and action samples of $\mathcal{B}$, $j = 0, 1, \cdots, n$, and $\mathrm{ReLU}(x) = \max(x, 0)$, $x \in \mathbb{R}$, is a Rectified Linear Unit (ReLU) function.

The actor loss function in (5.7) consists of two parts. The first part $\hat{\mathcal{L}}_A$ is defined on the interaction data batch $\mathcal{B}_{\mathrm{I}}^{\pi}$ and has the same form as the conventional actor loss function of an off-policy RL agent. The second part $\mathcal{L}_{BC}$ is a novel IBC-based loss function defined on the demonstration data batch defined on the interaction data batch $\mathcal{B}_{\mathrm{D}}^{\pi}$. It is used to penalize the current policy π_θ if produces a worse action $\pi_\theta(s_j)$ than the demonstration action a_j. As a result, it forces the policy π_θ to perform better than the demonstration policy during the training process. The extent of this effect is controlled by the BC ratio λ_{BC}. In this sense, BC is seamlessly integrated into the training process of the actor network, which leads to a flexible manner of updating the policies.

The form of the BC loss function (5.8) is inspired by the EBC-based RL in previous work which penalizes the deviation between the current policy π_θ and the demonstration policy [63]. Nevertheless, (5.8) adopts the IBC technology which penalizes a certain energy function of the current policy [65]. In this chapter, the energy function is selected as the deviation between the value functions of the current policy and the demonstration policy. Meanwhile, a ReLU operation is exerted to the deviation since no penalty is needed if the current policy performs better than the demonstration policy. The IBC-based loss function is dedicated to improving the value of the current policy instead of its similarity to the demonstration policy. For example, the penalty may not be exerted even if the two policies a_j and $\pi_\theta(s_j)$ are not the same, as long as $\pi_\theta(s_j)$ is superior to a_j according to the current value Q_w. Therefore, the IBC-based loss function is expected to have a better capability of avoiding overfitting the demonstration policy compared to the EBC-based one.

Refined Critic Loss

The IBC loss in (2.8) depends on the parameter w of the critic network Q_w. If the critic network Q_w is not well trained, the BC loss $\mathcal{L}_{BC}$ may not be able to accurately capture the penalty that should be imposed on the current policy π_θ, leading to an invalid loss that does not help the policy update. This is especially likely to occur in the initial stage of the

learning process. To solve this problem, a refined data batch $\mathcal{B} = \mathcal{B}_{\mathrm{D}}^{Q} \cup \mathcal{B}_{\mathrm{I}}^{Q}$ is used to compose the loss function for the critic network, $\mathcal{L}_C\left(\mathcal{B}_{\mathrm{D}}^{Q} \cup \mathcal{B}_{\mathrm{I}}^{Q}\right)$, where the form of $\mathcal{L}_C$ is defined in (2.7). Here, $\mathcal{B}_{\mathrm{D}}^{Q} \cup \mathcal{B}_{\mathrm{I}}^{Q}$ is referred to as a refined data batch since it has a larger proportion of demonstration data compared to an interaction data batch $\mathcal{B}_{\mathrm{I}}^{Q}$ of the same size. The ratio between the sizes of $\mathcal{B}_{\mathrm{D}}^{Q}$ and $\mathcal{B}_{\mathrm{I}}^{Q}$ is referred to as the refining factor λ_{RF}. Here, human demonstration data are assumed to be very likely to have higher values than random data. Therefore, a refined data batch with a larger λ_{RF} is likely to better describe the true critic loss and more helpful to the booting up of the training of the critic network. However, an overlarge λ_{RF} may lead to overfitting of the human demonstration. In this chapter, a constant λ_{RF} is selected. For better performance, the demo batch $\mathcal{B}_{\mathrm{D}}^{Q}$ can be gradually eliminated from $\mathcal{B}^{Q}$ as the learning proceeds, which renders a decreasing refining factor. This can be an interesting topic for future work.

The Training Algorithm for An IBC-DMP Agent

The training procedure of the IBC-DMP agent is formulated as Algorithm 2, where N is the total number of episodes during the training process, T is the ending time of an episode, and $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ is a random variable for a certain time $t = 0, 1, \cdots, T$. Algorithm 2 inherits the following common points from the conventional off-policy RL method.

- Initialization of the four neural networks (lines 2 and 3).
- The random sampling of actions (line 9).
- Storing interaction data (line 13).
- Sampling data from the interaction buffer (line 16).
- The update methods for the networks (lines 19 and 20).

Meanwhile, Algorithm 2 is different from the conventional off-policy RL agent in the following aspects.

- Initialization of the buffers $\mathcal{B}_{\mathrm{D}}$ and $\mathcal{B}_{\mathrm{I}}$ (line 1).
- A Multi-DoF DMP as the environment model (line 10).
- Sampling data from the demonstration buffer (line 15).
- The computation of loss functions (lines 17 and 18).

Algorithm 2 The demo IBC-DMP DDPG algorithm

Input: demonstration buffer $\mathcal{B}_{\mathrm{D}}$ and interaction buffer $\mathcal{B}_{\mathrm{I}}$
Output: trained policy parameter θ

1: Import demonstration data to buffers $\mathcal{B}_{\mathrm{D}}$, $\mathcal{B}_{\mathrm{I}}$
2: Randomly initialize the source networks π_θ, Q_w
3: Initialize the target networks $\pi_{\theta'}$, $Q_{w'}$ with $\theta' \leftarrow \theta$, $w' \leftarrow w$
4: **for** $i \leftarrow 1$ **to** N **do**
5: Sample the initial position $\mathbf{x}_{\mathrm{i}}$, the obstacle position $\mathbf{x}_{\mathrm{b}}$, and the goal position $\mathbf{x}_{\mathrm{g}}$
6: Initialize the state $s_0 = \mathcal{X}_0$
7: **for** $t \leftarrow 0$ **to** T **do**
8: Observe the state $s_t = \mathcal{X}_t$
9: Sample an action $a_t = \pi_\theta(s_t) + \epsilon_t$
10: Update the DMP model using (5.5)
11: Observe the successive state $s'_t = \mathcal{X}_{t+\Delta t}$
12: Calculate the instant reward r_t and flag d_t
13: Store $\{s_t, a_t, s'_t, r_t, d_t\}$ to $\mathcal{B}_{\mathrm{I}}$
14: **if** *UPDATE* is **true then**
15: Sample random batches $\mathcal{B}_{\mathrm{D}}^{\pi}$, $\mathcal{B}_{\mathrm{D}}^{Q}$ from $\mathcal{B}_{\mathrm{D}}$
16: Sample random batches $\mathcal{B}_{\mathrm{I}}^{\pi}$, $\mathcal{B}_{\mathrm{I}}^{Q}$ from $\mathcal{B}_{\mathrm{I}}$
17: Calculate the actor loss $\mathcal{L}_A(\mathcal{B}_{\mathrm{D}}^{\pi}, \mathcal{B}_{\mathrm{I}}^{\pi})$ using (5.7)
18: Calculate the critic loss $\mathcal{L}_B\left(\mathcal{B}_{\mathrm{D}}^{Q} \cup \mathcal{B}_{\mathrm{I}}^{Q}\right)$ using (2.7)
19: Update the source networks π_θ, Q_w using (2.9)
20: Update the target networks $\pi_{\theta'}$, $Q_{w'}$ using (2.6)
21: **end if**
22: $s_t \leftarrow s'_t$
23: **end for**
24: **end for**

5.5 Evaluation in Simulation

In this section, the efficacy of the proposed IBC-DMP RL agent is evaluated using a simulation study. A point-to-point reaching case in the three-dimensional task space is considered, where the end-effector of a robot is expected to move from a fixed initial position $\mathbf{x}_{\mathrm{i}} = (0,\ 0,\ 0.05)$ m to an arbitrary goal position $\mathbf{x}_{\mathrm{g}}$ while avoiding the collision with a cylinder obstacle placed at an arbitrary position $\mathbf{x}_{\mathrm{b}}^{(1,2)}$. The radius and height of the obstacle are $r_{\mathrm{b}} = 0.035$ m and $\mathbf{x}_{\mathrm{b}}^{(3)} = 0.16$ m which are the same as the demonstration recording experiment in Section 3. Note that assigning a fixed initial position $\mathbf{x}_{\mathrm{i}}$ does not lose the generality since it is always possible to use a coordinate transformation to transform any initial position in practice to $\mathbf{x}_{\mathrm{i}}$. Also, in this simulation study, the robot end-effector is assumed as a point and only consider its position. The experimental study in Section 5.6 will show how to deploy the trained policy to the end-effector of a robot manipulator in a pick-and-place task. The code for the simulation study is implemented based on the Spinningup baseline programs [81] and written in Python. All training and test processes are performed on a Thinkpad laptop workstation with Intel(R) Core(TM) i7-10750H CPU at 2.60 GHz. The programs and data of the simulation studies are published at [82].

Agent Training

With the demonstration data collected in Section 3, Algorithm 2 is proposed to train an IBC-DMP agent for the point-to-point reaching task. The size of the demo buffer $\mathcal{B}_{\mathrm{D}}$ is $N_{\mathrm{D}} = 123171$, which contains all eligible demonstration samples. The size of the experience replay buffer $\mathcal{B}_{\mathrm{I}}$ is $N_{\mathrm{R}} = 10^6$ which is sufficiently large to include both the demonstration samples and the history samples during the training process. Other hyper-parameters of training are listed in Table 5.3, where N_{D}^{π}, N_{D}^{Q}, N_{I}^{π}, and N_{I}^{Q} are the sizes of the data batches $\mathcal{B}_{\mathrm{D}}^{\pi}$, $\mathcal{B}_{\mathrm{D}}^{Q}$, $\mathcal{B}_{\mathrm{I}}^{\pi}$, and $\mathcal{B}_{\mathrm{I}}^{Q}$. The configuration corresponds to a refining factor $\lambda_{\mathrm{RF}} = 9$. The actor and the critic are three-layer forward neural networks in which the neuron numbers are 64,

128, and 64. The actuation functions of the networks are Rectified Linear Unit (ReLU) functions. The parameters of the networks are randomly initialized.

Table 5.3: The Hyper-Parameters of the Training of the IBC-DMP Agent

Par.	Value	Par.	Value	Par.	Value	Par.	Value
γ	0.99	σ	0.1	T	5	Δt	0.02
N	500	λ	0.995	α_θ	10^{-3}	α_w	10^{-3}
N_D^Q	450	N_I^Q	50	N_D^π	100	N_I^π	100

Each training episode is a run of the Multi-DoF DMP model (5.5) from the fixed initial position $\mathbf{x}_\mathrm{i}$ to a random goal position $\mathbf{x}_\mathrm{g}$ uniformly sampled from $\mathbf{x}_\mathrm{g} \in \mathcal{P}_\mathrm{g}$, where $\mathcal{P}_\mathrm{g} = \{P_\mathrm{g},\ \Delta P_\mathrm{g}\}$ is a polyhedral region with $P_\mathrm{g} = (0.30\ 0.35\ 0.08)$ m being its center coordinate and $P_\mathrm{g} + \Delta P_\mathrm{g}$ being its vertexes, where $\Delta P_\mathrm{g} = (\pm 0.05\ \pm 0.05\ \pm 0.01\ )$ m. The purpose of uniform sampling is to improve the robustness of the trained agent against the perturbation of the actual goal positions. Also, the obstacle position is randomly sampled by $\mathbf{x}_\mathrm{b} \in \mathcal{P}_\mathrm{b}(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{g})$ to improve the robustness of the agent to obstacle positions, where $\mathcal{P}_\mathrm{b}(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{g}) = \{P_\mathrm{b}(\mathbf{x}_\mathrm{i}, \mathbf{x}_\mathrm{g}), \Delta P_\mathrm{b}\}$ is a polyhedral sampling region that depends on the initial position $\mathbf{x}_\mathrm{i}$ and the sampled goal position $\mathbf{x}_\mathrm{g}$, with $P_\mathrm{b} = \dfrac{\mathbf{x}_\mathrm{i} + \mathbf{x}_\mathrm{g}}{2}$ being the geometry center and $P_\mathrm{b} + \Delta P_\mathrm{b}$ being the vertexes, where $\Delta P_\mathrm{b} = (\pm 0.05\ \pm 0.05\ \pm 0.02)$ m. The obstacle sampling space $\mathcal{P}_\mathrm{b}$ is set generally in the mid-way between the fixed initial position $\mathbf{x}_\mathrm{i}$ and the goal sampling space $\mathcal{P}_\mathrm{g}$ to intentionally create challenging environmental configurations for the agent training.

The three IBC-DMP agents is trained with different BC ratios $\lambda_\mathrm{BC} = 1, 1.5, 2$ to evaluate the influence of IBC on agent training. To show the advantage of the IBC-DMP agent, a comparison study is conducted with a conventional agent without IBC. The experience replay buffer of the no-IBC agent only has one interaction buffer. For a fair comparison, the interaction buffer is filled with random samples generated from the multi-DoF DMP model (5.5) using random actions. Apart from this, all other training parameters of the no-IBC agent are the same as the IBC-DMP agents. Moreover, for each agent, $M = 10$ policies

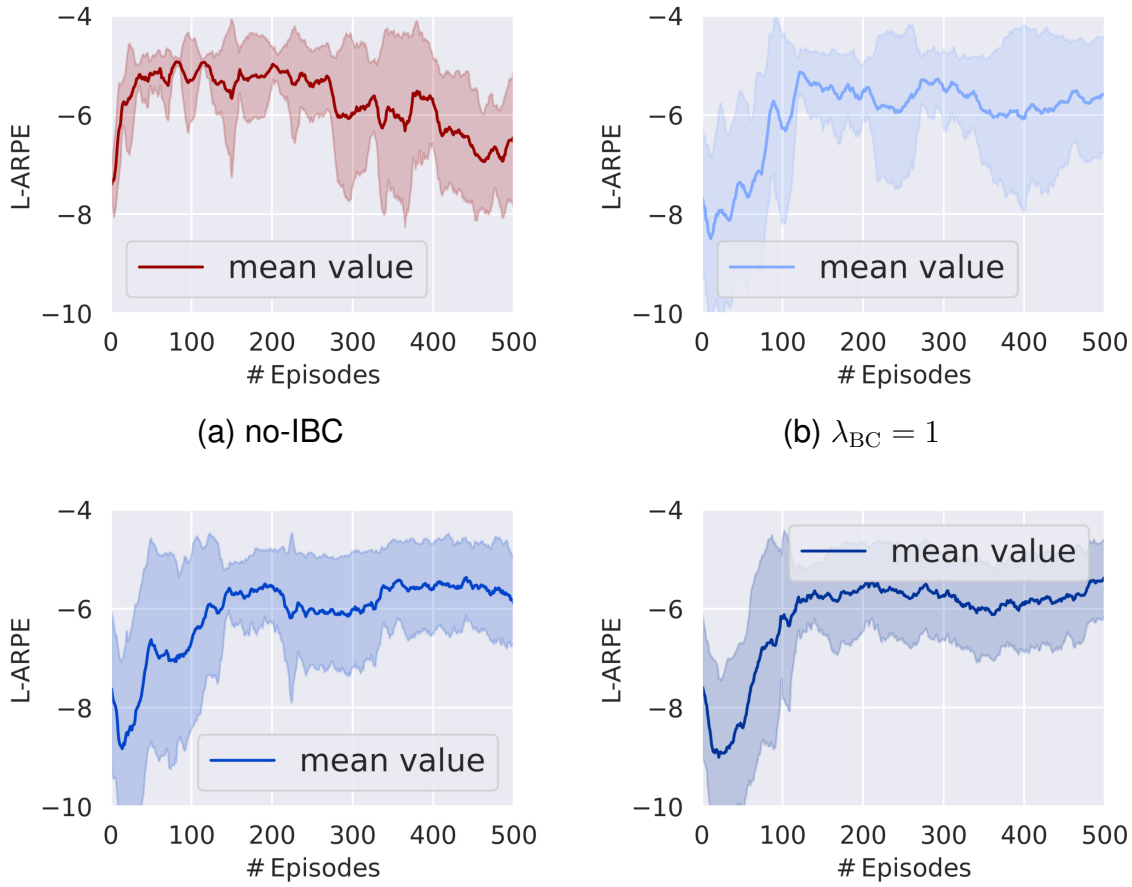
are repeatedly trained with the same initial condition but with different random seeds. The evaluation of the training performance of the agents will be conducted on all M policies to balance the effects of randomness.

Convergence Performance

At first, the convergence performance of the agents during the training process is evaluated. For each training episode $i = 1, 2, \cdots, N$ of each randomization $j = 1, 2, \cdots, M$, the accumulated reward per training episode (ARPE) score defined as $R_{ij} = \sum_{t=0}^{T} r_t^{ij}$ is used to value the training performance of an episode, where r_t^{ij} is the instant reward at step $t = 0, 1, \cdots, T$ of episode i for random seed j. Then, the performance of an RL agent can be evaluated by whether the ARPE converges to a high value. Nevertheless, the ARPE values among different training episodes may vary greatly. On the other hand, all ARPE values are non-positive. Therefore, the logarithm of ARPE (L-ARPE) defined as $L_{ij} = -\ln(1-R_{ij})$ is used to denote the training performance, for better visualization. The lines indicating the change of the L-ARPEs of all agents as the episode increases are illustrated in Fig. 5.3. In each subfigure, the solid line denotes the mean values $\mu(L_i) = \frac{1}{M}\sum_{j=1}^{M} L_{ij}$ of the L-ARPE lines over the M random seeds, and the shadow region represents the standard deviation $\sigma(L_i) = \frac{1}{M}\sum_{j=1}^{M}\left(L_{ij} - \mu(L_i)\right)^2$ of the L-ARPE lines. The lines in all subfigures are smoothed out with 10 episodes for clear presentation.

From Fig. 5.3, the L-ARPE lines of all the IBC-DMP RL agents ultimately converge to steady values before 500 episodes. Among these IBC-DMP agents, the one with $\lambda_{\mathrm{BC}} = 2$ has the highest ultimate average L-ARPE $\mu(L_N)$ and the smallest ultimate standard deviation $\sigma(L_N)$. It also shows a more clear reduction of the standard derivation $\sigma(L_i)$ as i increases, compared to other BC ratios. Nevertheless, the no-IBC agent failed to achieve convergence. Even though it achieves a high score $\mu(L_i)$ and low deviation $\sigma(L_i)$ in the early stage of the training, its performance becomes worse as the training proceeds. This comparison study indicates the advantages of the IBC-DMP agents with faster convergence

and higher stability to randomization than the conventional no-IBC agents.



(a) no-IBC (b) $\lambda_{\mathrm{BC}} = 1$

(c) $\lambda_{\mathrm{BC}} = 1.5$ (d) $\lambda_{\mathrm{BC}} = 2$

Figure 5.3: The L-ARPE lines of the agents in training.

Trained Scores

Apart from the L-ARPE lines, the trained performance of the agents is quantitatively evaluated by analyzing their training scores. Table 5.4 shows the ultimate L-ARPE scores L_{Nj} of all four agents for all random seeds $j = 1, 2, \cdots, M$. The averaged score $\mu(L_N)$ and the standard deviation $\sigma(L_N)$ are also displayed.

It is shown that all IBC-DMP agents have higher ultimate average L-ARPE scores and smaller standard deviations than the no-IBC agent. Specifically, all IBC-DMP agents have

training scores that are higher than or approximately equal to -6, while the score of the no-IBC agent is lower than -6. In this sense, the L-ARPE value can be selected to -6 as a standard to judge whether a trajectory is of good performance or not. The results indicate the superior training performance of the IBC-DMP agents over the conventional no-IBC agent. Table 5.4 also shows that $\lambda_{\mathrm{BC}} = 2$ is the best BC ratio among all configurations. The overall results indicate that IBC-DMP can improve the training performance of an RL agent given a proper BC ratio.

Table 5.4: The Quantitative Scores of Agent Training

Scores	no-IBC	$\lambda_{\mathrm{BC}}=1$	$\lambda_{\mathrm{BC}}=1.5$	$\lambda_{\mathrm{BC}}=2$
L_{N1}	-7.9646	-8.0932	-4.8913	-5.0798
L_{N2}	-9.2790	-5.1169	-8.2311	-4.8871
L_{N3}	-7.2851	-5.1896	-7.1078	-5.1206
L_{N4}	-6.9950	-4.9594	-5.1420	-5.1225
L_{N5}	-4.7116	-5.0356	-4.9359	-4.8457
L_{N6}	-5.0261	-4.7340	-4.7847	-5.4356
L_{N7}	-6.4798	-8.1941	-6.7794	-4.8920
L_{N8}	-8.3686	-5.4488	-5.0069	-5.3338
L_{N9}	-7.9009	-5.0293	-5.5441	-4.7767
L_{N10}	-4.8554	-5.2022	-7.5939	-7.8335
$\mu(L_N)$	**-6.8866**	**-5.7003**	**-6.0017**	**-5.3328**
$\sigma(L_N)$	**1.5088**	**1.2342**	**1.2297**	**0.8577**

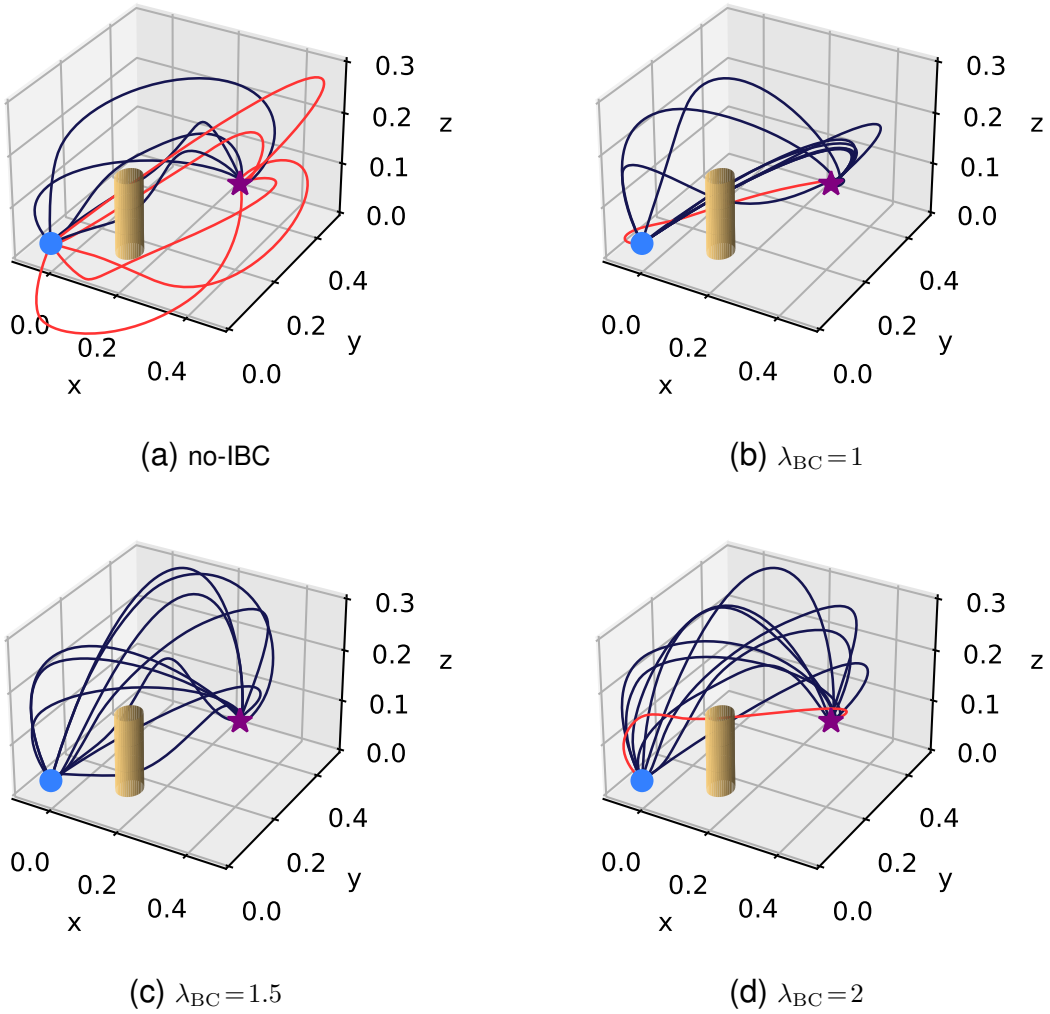
Agent Test

In this subsection, the performance of the four trained agents is evaluated in a test study. The Multi-DoF DMP model is required to start from a fixed initial position $\mathbf{x}_{\mathrm{i}} = (0\ 0\ 0.05)$ m to a random goal position $\mathbf{x}_{\mathrm{g}}$ while avoiding a cylinder obstacle in a random position $\mathbf{x}_{\mathrm{b}}$. For each trained policy $j = 1, 2, \cdots, M$ of each agent, $R = 400$ test runs are performed with different goal positions $\mathbf{x}_{\mathrm{g}}$ and obstacle positions $\mathbf{x}_{\mathrm{b}}$. The goal positions $\mathbf{x}_{\mathrm{g}}$ are sampled from a polyhedral region $\tilde{\mathcal{P}}_{\mathrm{g}} = \{\tilde{P}_{\mathrm{g}}, \Delta\tilde{P}_{\mathrm{g}}\}$, where $\tilde{P}_{\mathrm{g}} = (0.32\ 0.34\ 0.09)$ m and $\Delta\tilde{P}_{\mathrm{g}} = (\pm 0.3\ \pm 0.25\ \pm 0.05)$ m. Note that the sampling space for test $\tilde{P}_{\mathrm{g}}$ is larger

than the one for training P_{g} as introduced in Section 5.5, since the intention is to test the generalizability of the IBC-DMP RL agents to the goal positions that are not used for training. During the sampling of the goal positions, those goal positions that are too close to the initial positions are eliminated to ensure the feasibility of trajectory generation. For each sampled goal position $\mathbf{x}_{\mathrm{g}}$, the obstacle position $\mathbf{x}_{\mathrm{b}}$ is also randomly sampled from a polyhedral region $\tilde{\mathcal{P}}_{\mathrm{b}} = \{\tilde{P}_{\mathrm{b}}, \Delta\tilde{P}_{\mathrm{b}}\}$ of which the center $\tilde{P}_{\mathrm{b}} = \dfrac{\mathbf{x}_{\mathrm{i}} + \tilde{P}_{\mathrm{g}}}{2}$ is the mid-point between the initial position and the goal position, and the vertexes are $\dfrac{\mathbf{x}_{\mathrm{i}} + \tilde{P}_{\mathrm{g}}}{2} + \Delta\tilde{P}_{\mathrm{b}}$, where $\Delta\tilde{P}_{\mathrm{b}} = (\pm 0.05,\ \pm 0.05,\ \pm 0.02)$ m.

Test Trajectories

Fig. 5.4 shows the test trajectories of the trained agents for given goal and obstacle positions $\mathbf{x}_{\mathrm{g}} = (0.32\ 0.34\ 0.09)$ m and $\mathbf{x}_{\mathrm{b}} = (0.13\ 0.14\ 0.16)$ m. Each subfigure shows the trajectories of the corresponding agent generated by M trained policies for the fixed $\mathbf{x}_{\mathrm{g}}$ and $\mathbf{x}_{\mathrm{b}}$, where the red trajectories are those colliding with the obstacle or with the ground. Here, a trajectory is referred to as *with collisions* if it intersects with the obstacle or with the ground, i.e., if it has at least 1 discrete-time samples that are within the cylinder obstacle domain or under the ground. It is noticed that 5 out of 10 test trajectories of the conventional no-IBC agent are with collisions. On the contrary, each IBC-DMP agent only has at most 1 colliding trajectory. This indicates the superior test performance of IBC-DMP agents in terms of collision avoidance, compared to the no-IBC agent.



(a) no-IBC (b) $\lambda_{\mathrm{BC}}=1$

(c) $\lambda_{\mathrm{BC}}=1.5$ (d) $\lambda_{\mathrm{BC}}=2$

Figure 5.4: The test trajectories of the agents for fixed goal and obstacle positions, where the blue dot is the initial position, the purple star denotes the goal position, and the yellow cylinder represents the obstacle. The trajectories that collide with the obstacle or with the ground are marked as red, while those not are in blue.

Test Scores

The test performance of the agents is quantified using test scores. For each agent, the L-ARPE score L_{rj} defined in Section 5.5 is used to describe the test performance of a trajectory generated by policy $j = 1, 2, \cdots, M$ and position sample $r = 1, 2, \cdots, R$. Then, the averaged score $L_{\bar{r}j} = \frac{1}{R}\sum_{r=1}^{R} L_{rj}$ can be used to evaluate the general test performance of policy j. The test scores $L_{\bar{r}j}$ for all policies j and all trained agents are listed

in Table 5.5, where the mean values and standard deviations of the test scores, defined as $\mu(L_{\bar{r}}) = \frac{1}{M}\sum_{j=1}^{M} L_{\bar{r}j}$ and $\sigma(L_{\bar{r}}) = \frac{1}{M}\sum_{j=1}^{M}(L_{\bar{r}j} - \mu(L_{\bar{r}}))^2$ are also presented.

Table 5.5 clearly shows that all IBC-DMP agents have much higher average score $\mu(L_{\bar{r}})$ and smaller standard deviation $\sigma(L_{\bar{r}})$ than the no-IBC agent. Specifically, all IBC-DMP agents are higher than or approximately equal to the performance standard -6, while the no-IBC agent has a lower test score. This indicates the advantage of the IBC-DMP agents in test performance. Since there are significantly larger sampling spaces for goal and obstacle positions in the test than that in training, the superior test performance of the IBC-DMP agents also reflects their better generalizability to various goal and obstacle positions than the no-IBC agents. Also, among the IBC-DMP agents, the one with the largest BC ratio (λ_{BC}) performs the best with the highest average score $\mu(L_r)$ =-5.886 and the smallest standard deviation $\sigma(L_r)$ =0.241. This indicates that it does not only have the best overall test performance but also has the best stability to the changes of different environmental conditions. The overall results validate the efficacy of the IBC-DMP RL framework in improving the generalizability and stability of RL agents.

Table 5.5: The Quantitative Scores of Agent Test

Scores	**no-IBC**	$\lambda_{\mathrm{BC}}=1$	$\lambda_{\mathrm{BC}}=1.5$	$\lambda_{\mathrm{BC}}=2$
$L_{\bar{r}1}$	-6.1337	-6.6512	-6.1265	-6.0110
$L_{\bar{r}2}$	-9.7404	-5.9544	-5.1733	-5.6387
$L_{\bar{r}3}$	-6.1705	-5.9066	-6.0557	-5.8746
$L_{\bar{r}4}$	-5.6025	-5.9945	-5.9910	-5.9988
$L_{\bar{r}5}$	-5.7201	-5.9469	-6.0073	-5.5120
$L_{\bar{r}6}$	-5.5816	-5.6440	-6.6854	-6.1458
$L_{\bar{r}7}$	-7.1732	-6.3668	-5.6431	-5.5444
$L_{\bar{r}8}$	-8.8465	-6.2082	-5.7995	-6.2141
$L_{\bar{r}9}$	-6.3103	-5.8869	-6.2159	-5.8737
$L_{\bar{r}10}$	-5.6402	-5.7124	-6.1927	-6.2164
$\mu(L_{\bar{r}})$	**-6.6919**	**-6.0272**	**-5.9890**	**-5.9030**
$\sigma(L_{\bar{r}})$	**1.3916**	**0.2882**	**0.3772**	**0.2505**

Collision Rates

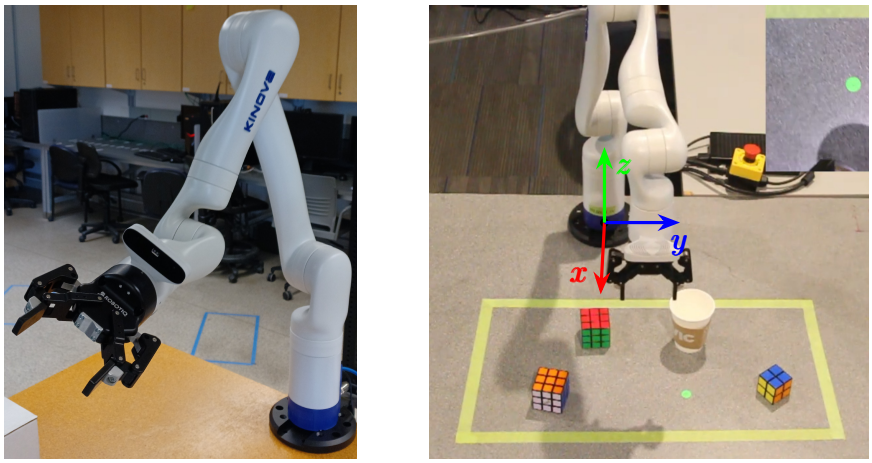
One of the main concerns of robot motion planning is the avoidance of collision with obstacles in the environment. To evaluate the performance of the agents with respect to collision avoidance in the test study, the collision rates $R_{cj} = N^j_{\text{cls}}/N^j_{\text{ttl}}$ is calculated for all policies $j = 1, 2, \cdots, 10$, where $N^j_{\text{ttl}} = 400$ is the total number of test trajectories of policy j and N^j_{cls} is the number of trajectories with collisions. Table 5.6 lists the collision rates of all 10 policies of the four agents. The overall average value and the standard deviation of the collision rates of each agent are also presented. It is clearly shown that the IBC-DMP agents have far smaller collision rates than the no-IBC agent. They also have smaller standard deviation values. The agent with $\lambda_{\text{BC}} = 2$ can be recognized as the safest and the most reliable one among the four agents with the lowest average collision rates and the smallest standard deviation. The overall results indicate that IBC-DMP agents have superior collision avoidance performance over the conventional no-IBC agent.

Table 5.6: The Collision Rates of Agent Test

Scores	**no-IBC**	$\lambda_{\text{BC}} = 1$	$\lambda_{\text{BC}} = 1.5$	$\lambda_{\text{BC}} = 2$
R_{c1}	0.25%	29.75%	17.50%	0.25%
R_{c2}	100%	0	0.25%	0
R_{c3}	43.00%	0	0.25%	0
R_{c4}	0	0	0	0
R_{c5}	0	0	0.50%	0
R_{c6}	0	0	47.50%	0
R_{c7}	50.25%	16.75%	16.25%	0
R_{c8}	100%	0	0	0
R_{c9}	12.25%	0	0	0
R_{c10}	0	0	0.25%	0.50%
$\mu(R_c)$	**30.57%**	**4.65%**	**8.25%**	**0.08%**
$\sigma(R_c)$	**0.3896**	**0.0974**	**0.1465**	**0.0016**

5.6 Experimental Validation

In this section, an experimental case study is conducted to evaluate the performance of the proposed IBC-DMP RL method. A Rubik's cube-stacking case is used to demonstrate how to use an IBC-DMP agent to accomplish a general pick-and-place-based assembly task. The robot used in this study is a 6-DoF Kinova® Gen 3 manipulator with a two-finger Robotiq® 2F-85 gripper and an Omnivision OV5640 on-gripper camera as shown in Fig. 5.5a. The robot is connected to a desktop workstation which is equipped with an AMD® Ryzen9 3950X CPU and an Intel® RTX3090 GPU. The operating system of the desktop is Ubuntu 18.04. The robot is controlled by the Kinova® Kortex API written in Python [83]. Another RGB camera is deployed in the front of the robot to provide vision from the third-person point of view, as shown in Fig. 5.5b.



(a) (b)

Figure 5.5: The Kinova® Gen3 robot (b) and the assembly scenario (b).

Experiment Configuration

As shown in Fig. 5.5b, the workspace of the cube-stacking scenario is marked as a rectangular region on the table in front of the Gen3 robot arm. the origin of the task coordinate is set at the robot base. The axes of the coordinate are shown as colored arrows in Fig. 5.5b.

The coordinates of the vertexes of the workspace are $(0.25, 0.45, 0)$ m, $(0.25, -0.2, 0)$ m, $(0.6, 0.45, 0)$ m, and $(0.6, -0.2, 0)$ m. The size of the workspace is determined to cover the largest view of the on-gripper camera within the reach of the robot gripper. Three Rubik's cubes with different sizes (two cubes with a 57 mm edge and one cube with a 45 mm edge) and colors are randomly placed within the rectangular region manually. A paper cup with a radius of 90 mm and a height of 110 mm is also manually placed at a random position as an obstacle. The robot is commanded to start from the HOME position at (0.356, 0.106, 0.277) m, as shown in Fig. 5.5b, pick up the Rubik's cubes, and place them at the GOAL position (marked as a green light point) one by one from the largest to the smallest (red, orange, and then blue), until they are piled up in a column. The piling-up process can be recognized as a simple assembly task, during which the robot gripper should not collide with the paper cup. The vision of the Rubik's cubes and the paper cup obstacle is captured using the on-gripper camera and their positions are calculated using the pre-built object detection libraries in YOLOv8 [84] and OpenCV [85].

For each experimental trial, the initial positions of the three cubes with red, orange, and blue top surfaces is represented as P_{red}, P_{org}, and P_{blu}, respectively. Additionally, three via-points $\tilde{P}_{\text{red}}$, $\tilde{P}_{\text{org}}$, and $\tilde{P}_{\text{blu}}$ are set 5 cm right above them to ease the grasping of the cubes. In this sense, six trajectories need to be generated for the cube-stacking task, including HOME$-\tilde{P}_{\text{red}}-G_{\text{red}}-\tilde{P}_{\text{org}}-G_{\text{org}}-\tilde{P}_{\text{blu}}-G_{\text{blu}}$, where G_{red}, G_{org}, and G_{blu} are the stacking positions of the cubes in the ultimate stacking column. The procedure of the stacking task is described in Algorithm 3, where *DMP* is a function used to generate the desired trajectories using the multi-DoF DMP model (5.1) with given initial, goal, and obstacle positions $\mathbf{x}_{\text{i}}$, $\mathbf{x}_{\text{g}}$, and $\mathbf{x}_{\text{b}}$, respectively, and the trained policy $\mathbf{f}$. In this experiment, the policy #5 is selected. No additional task-level complexities are introduced in this task, as the main goal of this chapter is to evaluate the efficiency of an IBC-DMP agent in generating safe trajectories. Also, for brevity, the multi-DoF DMP model is only used to generate the translational

positions of the gripper. The orientations of the gripper are commanded in a very intuitive and simple manner with the following principles.

- The gripper always points down to the table.
- The orientations about x- and y-axis are fixed for all generated trajectories.
- The orientations about z-axis are linearly changed during the motion, in order to grasp the cube or place it with the proper orientation in the line 5 of Algorithm 3.

Such a design ensures the successful execution of the cube-stacking task without causing singular configurations. All trajectories are designed in the Cartesian space and are mapped to the joint space of the robot using the pre-built inverse kinematics (IK) library of the Kinova® Gen 3 robot.

Algorithm 3 Cube stacking task procedure

1: Initialize robot gripper at HOME position
2: Assign $\mathbf{x}_{\mathrm{b}}$ with the paper cup position
3: **for** j **in** $\{\mathrm{red}, \mathrm{org}, \mathrm{blu}\}$ **do**
4: $\quad \mathbf{x}_{\mathrm{i}} \leftarrow \mathbf{x}_t$, $\mathbf{x}_{\mathrm{g}} \leftarrow \tilde{P}_j$
5: $\quad$ Generate trajectory $(\mathbf{x}_t, \dot{\mathbf{x}}_t) = DMP(\mathbf{x}_{\mathrm{i}}, \mathbf{x}_{\mathrm{g}}, \mathbf{x}_{\mathrm{b}}, \mathbf{f})$
6: $\quad$ **while** $\|\mathbf{x}_t - \mathbf{x}_{\mathrm{g}}\| > \varepsilon_T$ **do**
7: $\quad\quad$ Follow trajectory $(\mathbf{x}_t, \dot{\mathbf{x}}_t)$
8: $\quad$ **end while**
9: $\quad$ Grasp cube at P_j
10: $\quad$ Determine stacking position $\mathbf{x}_{\mathrm{i}} \leftarrow \mathbf{x}_t$, $\mathbf{x}_{\mathrm{g}} \leftarrow G_j$
11: $\quad$ Repeat line 5 to line 8
12: $\quad$ Release cube
13: **end for**

Experiment Results

To evaluate the overall efficacy of the IBC-DMP RL framework, policy # 6 of $\lambda_{\mathrm{BC}} = 1.5$ is selected, which performs the worst in the test study as shown in Table 5.5, for the experimental validation. If the selected policy demonstrates satisfactory performance, it can be concluded that the IBC-DMP framework is generally applicable in practical applications.

A total of 22 experimental trials are conducted to account for the influence of randomness. For each trial, the initial positions of the cubes and the cup are randomly placed by hand. Also, challenging situations are intentionally created where the robot gripper has a higher likelihood of maneuvering around the paper cup. One example trial is shown in Fig. 5.6, where the paper cup is in the mid-way between the red and the orange cubes, and the goal position. In this situation, the gripper must go around the cup to avoid collision with it, leading to nontrivial trajectories. Fig. 5.6 shows that the IBC-DMP policy can successfully generate collision-free trajectories for the robot gripper, which indicates the efficacy of the IBC-DMP motion planning framework.

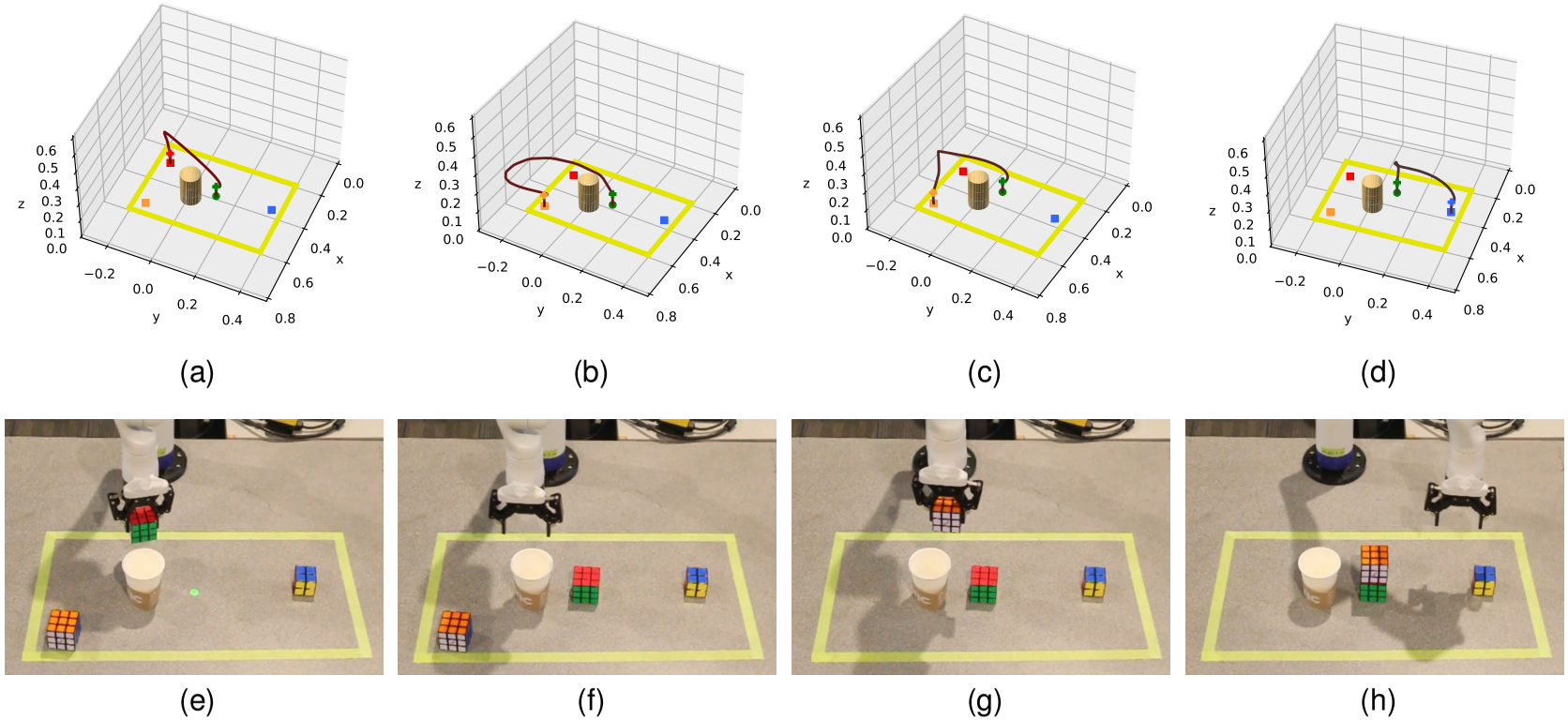


Figure 5.6: The training performance of the off-policy agent without BC and with BC subject to different λ_{BC} values.

Similar to Section 5.5, the L-ARPE scores is also used to quantitatively evaluate the performance of the IBC-DMP policy in the experimental study. The internal position sensor of the Gen 3 robot is used to record the actual executed paths of the gripper during the cube-stacking task. Fig. 5.7 is the box plot of the L-ARPE scores of these paths which are grouped according to trials. Each box denotes the distribution of the L-ARPE scores of the 6 trajectories of an experimental trial. The red bar in each box represents the median value of the L-ARPE scores which quantifies the overall performance of the policy in the corresponding trial. It is noticed that all red bars are higher than -7, which shows its advantage compared to the test scores of the no-IBC agent in Table 5.5. Also, the scores of 16 out of 22 trials are higher than the performance standard -6, which indicates the decent overall performance of the selected policy with respect to L-ARPE scores.

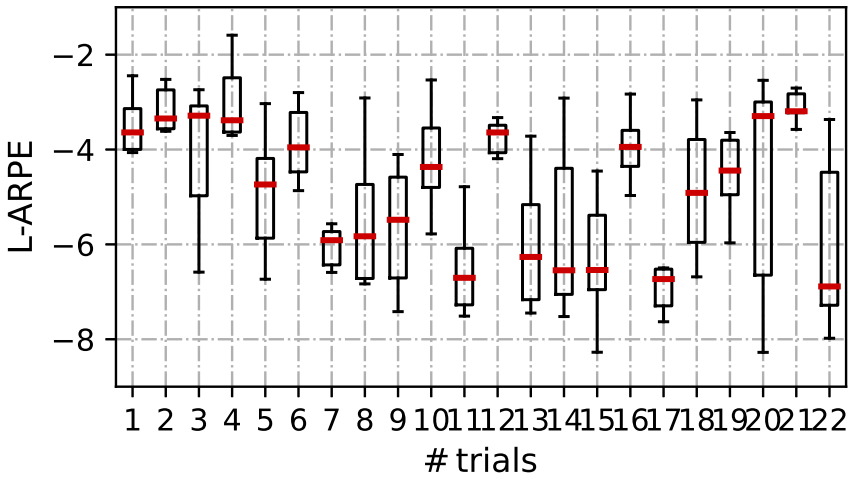


Figure 5.7: The training performance of the off-policy agent without BC and with BC subject to different λ_{BC} values.

The distribution of the ultimate reaching errors is displayed in the box plot in Fig. 5.8. It is noticed that all reaching errors are strictly restricted under 0.01 m due to the selection of the error threshold ε_T. Meanwhile, the overall collision rate is 13.64% which is higher than the testing collision rates of the IBC-DMP agents due to the uncertainties in the real

experiment. However, it is far lower than the testing collision rate of the conventional no-IBC agent. Therefore, the experimental results indicate that the selected policy achieves the ideal ultimate reaching errors as prescribed by the error threshold and has advantages over the conventional no-IBC agent in terms of collision avoidance. Since the selected policy is of inferior test performance among other IBC-DMP policies, its decent performance implies the efficacy of the proposed IBC-DMP RL framework in resolving practical robot tasks.

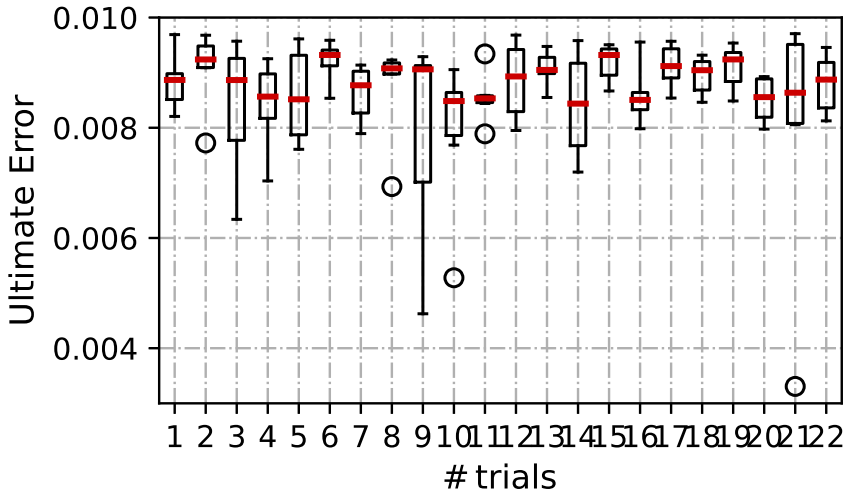


Figure 5.8: The training performance of the off-policy agent without BC and with BC subject to different λ_{BC} values.

Chapter 6

Concluding Remarks

This thesis contributes to the advancement of robot motion planning by showcasing the value of incorporating human demonstrations and expert knowledge for skill transfer. The proposed approaches offer practical solutions for enhancing the learning capabilities and adaptability of Reinforcement Learning (RL) agents. By effectively utilizing human demonstrations, RL agents become more capable of autonomously learning complex tasks and adapting to various scenarios. The insights gained from this research not only enhance our understanding of RL in robotics but also pave the way for the development of advanced and intelligent robotic systems capable of leveraging human expertise.

In this final chapter, we draw upon the findings and insights presented in the previous chapters to provide a comprehensive conclusion to this thesis. We reflect on the value of incorporating human demonstrations into RL for skill transfer and highlight the benefits of the proposed approaches: Dynamic Feature Extraction (DFE) and the integration of Multi-Degree of Freedom (DoF) Dynamic Movement Primitives (DMP) and Implicit Behaviour Cloning (IBC). Furthermore, we discuss potential areas for future research to extend and enhance the suggested framework.

6.1 Discussions

By exploring the synergies between RL and Learning from Demonstrations (LfD), our research aims to improve the learning capabilities of RL agents by leveraging the knowledge and expertise captured in human demonstrations. The two novel approaches presented in this thesis demonstrate the potential of effectively acquiring and utilizing abstract skills from human experts, leading to more efficient and adaptable robotic systems.

The first approach suggested in Chapter 4 focuses on the extraction of dynamic features from demonstrated trajectories within the DMP framework. By capturing the dynamic characteristics of human motion, a fresh perspective on transferable skills is provided, going beyond the reliance on explicit paths or trajectories commonly used in LfD. This approach eliminates the need for time-consuming manual parameter tuning and increases the robot's capability to learn motion autonomously. The extracted dynamic features can be integrated into the DMP framework, enabling RL agents to explore a wider range of potential trajectories and find optimal solutions, surpassing the constraints imposed by the demonstrated trajectory. The utilization of these human dynamic features enhances the robot's compliance with human intentions, bridging the gap between expert demonstrations and autonomous learning.

The second approach introduced in Chapter 5, the novel framework to facilitate off-policy RL for robot motion planning is presented. The framework addresses the integration of the two important technologies used to facilitate off-policy RL for robot motion planning: *Multi-DoF DMP* and *IBC*. They both can be recognized as the proper encoding of expert knowledge. The multi-DoF DMP is adapted from conventional motion primitives of which the effectiveness has been verified by many experts and peers. From a mathematical point of view, DMP models can be seen as a class of heuristic models dedicated to reducing the dimensionality of a planning problem. The dynamic structure of a DMP model reflects how expert knowledge is used to restrict the smoothness and inherent stabil-

ity of generated trajectories. Additionally, IBC, as an adapted version of BC for imitation learning, provides a different manner to encode expert knowledge by cloning the expert's behaviors in the task. Note that these two different sources of expert knowledge might not always be beneficial to the given robot task. They may even conflict with each other in certain circumstances. A very important technical point of this paper is to flexibly combine the two types of expert knowledge using an off-policy RL framework, such that the resulting motion planning policy is not overfitting to any source of expert knowledge. Furthermore, the advantages of both types of knowledge are fully exploited to improve the training speed, generalizability, and reliability of the RL agent. The decent results of the simulation and experimental studies provide evidence that the performance of RL can be improved by properly encoding expert knowledge.

Beyond the highlighted contributions, the creation of a human demonstration dataset stands as a significant enhancement to the field. This dataset encompasses 544 trajectories of the Point-to-Point Reaching (P2PR) task with obstacle avoidance, positioning itself as a valuable resource for future investigations. Its integration within the proposed framework of this thesis not only amplifies our current understanding but also introduces new avenues for researchers to explore the domain of transferable skills across diverse contexts. Moreover, the expansion of this dataset to encompass other motion primitives, such as circular motions or peg-in-hole tasks, has the potential to fortify the application of modular human skills in various robotic tasks. This inclusivity could even pave the way for combining existing skills to acquire entirely novel ones, offering a strong basis for incubating innovative research in the domain of skill transfer for robot motion planning.

6.2 Limitations and Future Work

In this section, the results of the suggested framework is discussed individually for each approach. Additionally, potential areas for further research to extend the proposed framework

will be investigated.

Human-Robot Dynamic Feature Transfer via DMP

The key idea of DFE is to maximize the human-likeness while guaranteeing the representability of its dynamic system across similar motions in the task space. The implementation of the extracted value into the real robot showed that the robots reproduce trajectories robustly and stably throughout the motion.

The extracted features generated a trajectory that performed as robustly and stably as the fine-tuned heuristic features, while the trajectories generated with the heuristic features before tuning resulted in sudden acceleration or failure to reach the goal. The RL with DMP framework was more sensitive to the right choice of dynamic features. The extracted features showed stable performance as well as LfD with DMP, delivering the dynamic characteristic extracted from human demonstration.

There are several areas for further investigation. While DMP uses second-order dynamics, the proposed DFE can be examined for higher-order system dynamics (e.g., jerk in a third-order equation of motion). Also, further research may investigate the application of the suggested framework to other variations of DMP designed for different scenarios, such as learning obstacle avoidance or kicking motion.

Further investigations could explore different schemes to improve the objective function. For example, the time integral used for Euclidean distance and standard deviation may emphasize distance and similarity when motions are misaligned in time. The methods invariant to time shifts, such as phase correlation, could be used to overcome this limitation.

There are several potential areas for further exploration. Future research could explore the application of the proposed framework to other variants of DMP tailored for specific scenarios, such as obstacle avoidance or kicking motions involving additional terms in the formula. Additionally, investigations could focus on enhancing the objective function by exploring alternative schemes. For instance, integrating time-invariant methods like phase

correlation could mitigate the issue of emphasizing distance and similarity solely based on temporal misalignments, as observed with the current time integral-based approach for Euclidean distance and standard deviation.

RL-based Robot Motion Planning Enhanced with IBC and DMP

An off-policy RL agent serves as a bridge to flexibly combine the expert knowledge encoded by the two methods, resulting in an advantageous agent with improved training speed, generalizability, and reliability. Its efficacy and advantage over the conventional no-IBC agent are validated by simulation and experimental studies. In future work, we will focus on improving the stability of the Multi-DoF DMP model and the sensitivity of IBC-DMP agents to exploring action noise.

Although the general efficacy of the IBC-DMP framework is validated, it still has limitations. The multi-DoF DMP model has a higher flexibility than the conventional DMP since its actuation function is dependent on the state of the model. However, this may sacrifice the stability of the DMP model, possibly leading to odd-shaped trajectories. Besides, an IBC-DMP may be over-trained if the number of training episodes is too large, where the training score of the policy decreases or becomes unsteady as the training proceeds longer. This may also be due to the sacrifice of the inherent stability of the Multi-DoF DMP. Another limitation of IBC-DMP is that its performance tends to be sensitive to the exploring noise added to actions during the training stage. Further investigations on the correlation between the performance of IBC-DMP and the action noise are needed in future work.

Bibliography

[1] Z. Zhang, K. Qian, B. W. Schuller, and D. Wollherr, "An online robot collision detection and identification scheme by supervised learning and bayesian decision theory," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1144–1156, 2020.

[2] A. M. S. Enayati, Z. Zhang, and H. Najjaran, "A methodical interpretation of adaptive robotics: Study and reformulation," *Neurocomputing*, vol. 512, pp. 381–397, 2022.

[3] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 1461–1466, IEEE, 2020.

[4] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6244–6251, IEEE, 2018.

[5] B. Nemec, L. Žlajpah, and A. Ude, "Door opening by joining reinforcement learning and intelligent control," in *2017 18th International Conference on Advanced Robotics (ICAR)*, pp. 222–228, IEEE, 2017.

[6] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[7] Y. Yang, J. Pan, and W. Wan, "Survey of optimal motion planning," *IET Cyber-Systems and Robotics*, vol. 1, no. 1, pp. 13–19, 2019.

[8] J.-J. Kim and J.-J. Lee, "Trajectory optimization with particle swarm optimization for manipulator motion planning," *IEEE transactions on industrial informatics*, vol. 11, no. 3, pp. 620–631, 2015.

[9] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[10] I. Pérez-Hurtado, M. Á. Martínez-del Amor, G. Zhang, F. Neri, and M. J. Pérez-Jiménez, "A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning," *Integrated Computer-Aided Engineering*, vol. 27, no. 2, pp. 121–138, 2020.

[11] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9535–9541, IEEE, 2020.

[12] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[13] Y. Wang and X. Guo, "Memory-based stochastic trajectory optimization for manipulator obstacle avoiding motion planning," in *2022 7th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pp. 188–194, IEEE, 2022.

[14] A. Billard and D. Grollman, "Robot learning by demonstration," *Scholarpedia*, vol. 8, no. 12, p. 3824, 2013. revision #138061.

[15] J. N. Pires, A. Loureiro, T. Godinho, P. Ferreira, B. Fernando, and J. Morgado, "Welding robots," *IEEE robotics & automation magazine*, vol. 10, no. 2, pp. 45–55, 2003.

[16] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[17] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[18] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 362–369, 2019.

[19] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[20] A. O. Ly and M. Akhloufi, "Learning to drive by imitation: An overview of deep behavior cloning methods," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 195–209, 2020.

[21] R. D. Crowninshield and R. A. Brand, "A physiologically based criterion of muscle force prediction in locomotion," *Journal of Biomechanics*, vol. 14, no. 11, pp. 793–801, 1981.

[22] B. I. Prilutsky and V. M. Zatsiorsky, "Optimization-based models of muscle coordination," *Exercise and sport sciences reviews*, vol. 30, no. 1, p. 32, 2002.

[23] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, and M. Q.-H. Meng, "A survey of learning-based robot motion planning," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, pp. 302–314, 2021.

[24] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[25] J. Bao, G. Zhang, Y. Peng, Z. Shao, and A. Song, "Learn multi-step object sorting tasks through deep reinforcement learning," *Robotica*, pp. 1–17, 2022.

[26] J. Kulhánek, E. Derner, and R. Babuška, "Visual navigation in real-world indoor environments using end-to-end deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4345–4352, 2021.

[27] Z. Zhang, R. Dershan, A. M. S. Enayati, M. Yaghoubi, D. Richert, and H. Najjaran, "A high-fidelity simulation platform for industrial manufacturing by incorporating robotic dynamics into an industrial simulation tool," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9123–9128, 2022.

[28] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 737–744, IEEE, 2020.

[29] Y. Tsurumine, Y. Cui, E. Uchibe, and T. Matsubara, "Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation," *Robotics and Autonomous Systems*, vol. 112, pp. 72–83, 2019.

[30] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7973–7980, 2021.

[31] X. Yang, Z. Ji, J. Wu, Y.-K. Lai, C. Wei, G. Liu, and R. Setchi, "Hierarchical reinforcement learning with universal policies for multistep robotic manipulation," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[32] H. Xiong, T. Ma, L. Zhang, and X. Diao, "Comparison of end-to-end and hybrid deep reinforcement learning strategies for controlling cable-driven parallel robots," *Neurocomputing*, vol. 377, pp. 73–84, 2020.

[33] F. Voigt, L. Johannsmeier, and S. Haddadin, "Multi-level structure vs. end-to-end-learning in high-performance tactile robotic manipulation.," in *CoRL*, pp. 2306–2316, 2020.

[34] F. L. Da Silva and A. H. R. Costa, "A survey on transfer learning for multiagent reinforcement learning systems," *Journal of Artificial Intelligence Research*, vol. 64, pp. 645–703, 2019.

[35] F. Stulp, E. Theodorou, M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal, "Learning motion primitive goals for robust manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 325–331, IEEE, 2011.

[36] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.

[37] F. Stulp, J. Buchli, A. Ellmer, M. Mistry, E. A. Theodorou, and S. Schaal, "Model-free reinforcement learning of impedance control in stochastic environments," *IEEE Transactions on Autonomous Mental Development*, vol. 4, no. 4, pp. 330–341, 2012.

[38] A. Li, Z. Liu, W. Wang, M. Zhu, Y. Li, Q. Huo, and M. Dai, "Reinforcement learning with dynamic movement primitives for obstacle avoidance," *Applied Sciences*, vol. 11, no. 23, 2021.

[39] Y. Yuan, Z. Li, T. Zhao, and D. Gan, "Dmp-based motion generation for a walking exoskeleton robot using reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 5, pp. 3830–3839, 2020.

[40] H. Sun, W. Zhang, R. Yu, and Y. Zhang, "Motion planning for mobile robots—focusing on deep reinforcement learning: A systematic review," *IEEE Access*, vol. 9, pp. 69061–69081, 2021.

[41] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," *Journal of Intelligent Manufacturing*, vol. 33, no. 2, pp. 387–424, 2022.

[42] L. Yu, J. Luo, and K. Zhou, "An intelligent robot motion planning method and application via lppo in unknown environment," in *2022 12th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 265–270, IEEE, 2022.

[43] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[44] F. Ying, H. Liu, R. Jiang, and X. Yin, "Trajectory generation for multiprocess robotic tasks based on nested dual-memory deep deterministic policy gradient," *IEEE/ASME Transactions on Mechatronics*, 2022.

[45] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, "Surreal: Open-source reinforcement learning framework and robot manipulation benchmark," in *Conference on Robot Learning*, pp. 767–782, PMLR, 2018.

[46] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3389–3396, 2021.

[47] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel ddpg method with prioritized experience replay," in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, pp. 316–321, IEEE, 2017.

[48] J. Luo and H. Li, "Dynamic experience replay," in *Conference on robot learning*, pp. 1191–1200, PMLR, 2020.

[49] Y. Liu, Z. Li, H. Liu, and Z. Kan, "Skill transfer learning for autonomous robots and human–robot cooperation: A survey," *Robotics and Autonomous Systems*, vol. 128, p. 103515, 2020.

[50] M. E. Cansev, H. Xue, N. Rottmann, A. Bliek, L. E. Miller, E. Rueckert, and P. Beckerle, "Interactive human–robot skill transfer: A review of learning methods and user experience," *Advanced Intelligent Systems*, vol. 3, no. 7, p. 2000247, 2021.

[51] Z. Lu, N. Wang, and C. Yang, "A constrained dmps framework for robot skills learning and generalization from human demonstrations," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 6, pp. 3265–3275, 2021.

[52] X. Yu, P. Liu, W. He, Y. Liu, Q. Chen, and L. Ding, "Human-robot variable impedance skills transfer learning based on dynamic movement primitives," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6463–6470, 2022.

[53] F. Bian, D. Ren, R. Li, P. Liang, K. Wang, and L. Zhao, "An extended dmp framework for robot learning and improving variable stiffness manipulation," *Assembly Automation*, 2019.

[54] J. Choi, H. Kim, Y. Son, C.-W. Park, and J. H. Park, "Robotic behavioral cloning through task building," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1279–1281, IEEE, 2020.

[55] W. Farag and Z. Saleh, "Behavior cloning for autonomous driving using convolutional neural networks," in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1–7, IEEE, 2018.

[56] C. Chen, C. Yang, C. Zeng, N. Wang, and Z. Li, "Robot learning from multiple demonstrations with dynamic movement primitive," in *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 523–528, IEEE, 2017.

[57] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, "Prodmp: A unified perspective on dynamic and probabilistic movement primitives," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2325–2332, 2023.

[58] Y. Zhou and T. Asfour, "Task-oriented generalization of dynamic movement primitive," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3202–3209, IEEE, 2017.

[59] A. Kumar, J. Hong, A. Singh, and S. Levine, "Should i run offline reinforcement learning or behavioral cloning?," in *International Conference on Learning Representations*, 2021.

[60] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[61] Y. Tian, X. Cao, K. Huang, C. Fei, Z. Zheng, and X. Ji, "Learning to drive like human beings: A method based on deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[62] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6292–6299, IEEE, 2018.

[63] K. Gupta, *Reinforcement learning in complex environments with locally trained naïve agents*. PhD thesis, University of British Columbia, 2021.

[64] K. Gupta and H. Najjaran, "Exploiting abstract symmetries in reinforcement learning for complex environments," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3631–3637, IEEE, 2022.

[65] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*, pp. 158–168, PMLR, 2022.

[66] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*, pp. 261–280, Springer, 2006.

[67] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2011.

[68] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pp. 231–238, IEEE, 2011.

[69] Y. Cohen, O. Bar-Shira, and S. Berman, "Motion adaptation based on learning the manifold of task and dynamic movement primitive parameters," *Robotica*, vol. 39, no. 7, pp. 1299–1315, 2021.

[70] N. Hogan and D. Sternad, "Dynamic primitives of motor behavior," *Biological cybernetics*, vol. 106, no. 11, pp. 727–739, 2012.

[71] Y. Liang, W. Li, Y. Wang, R. Xiong, Y. Mao, and J. Zhang, "Dynamic movement primitive based motion retargeting for dual-arm sign language motions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8195–8201, IEEE, 2021.

[72] Y. Yuan, Z. L. Yu, L. Hua, Y. Cheng, J. Li, and X. Sang, "Hierarchical dynamic movement primitive for the smooth movement of robots based on deep reinforcement learning," *Applied Intelligence*, pp. 1–18, 2022.

[73] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *arXiv preprint arXiv:2102.03861*, 2021.

[74] V. Holzwarth, J. Gisler, C. Hirt, and A. Kunz, "Comparing the accuracy and precision of steamvr tracking 2.0 and oculus quest 2 in a room scale setup," in *2021 the 5th International conference on virtual and augmented reality simulations*, pp. 42–46, 2021.

[75] Z. Zhang and J. Hong, "Dataset of human hand motion planning," Apr. 2023.

[76] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud, "Learning compact parameterized skills with a single regression," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 417–422, IEEE, 2013.

[77] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," *Robotics: Science and Systems, MIT Press Journal*, vol. 6, pp. 33–40, 2011.

[78] E. Ugur and H. Girgin, "Compliant parametric dynamic movement primitives," *Robotica*, vol. 38, no. 3, pp. 457–474, 2020.

[79] Y. Zhou and T. Asfour, "Task-oriented generalization of dynamic movement primitive," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3202–3209, 2017.

[80] H. Li, Y. Wu, M. Chen, and R. Lu, "Adaptive multigradient recursive reinforcement learning event-triggered tracking control for multiagent systems," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[81] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[82] Z. Zhang, "Reinforcement learning for robot motion planning facilitated by implicit behavior cloning and dynamic movement primitive," July 2023.

[83] K. inc., "Kinova® kortex™," 2022.

[84] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023.

[85] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.