Leveraging Structured-Graph Correspondence in Imitation Learning

Die Nutzung von graphisch strukturierter Korrespondez im Imitationslernen
Master thesis by Alper Gece
Date of submission: September 15, 2023

1. Review: Prof. Dr.-Ing. Marius Pesavento
2. Review: Prof. Dr. Georgia Chalvatzaki
3. Review: M.Sc. Kay Hansel
4. Review: M.Sc. An Thai Le
Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Alper Gece, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 15. September 2023

Alper Gece

A. Gece

Acknowledgments

I express my deepest gratitude to Prof. Dr. Georgia Chalvatzaki for her invaluable mentorship and guidance throughout my thesis. I'm also sincerely grateful to Prof. Dr.-Ing. Marius Pesavento from the ETIT faculty, for graciously accepting my thesis and enabling me to research in the Computer Science faculty.

I extend my heartfelt appreciation to my dedicated supervisors, M.Sc. Kay Hansel and M.Sc. An Thai Le. Their expertise and support were instrumental in the successful completion of my research goals.

Special thanks go to the cluster team of the Intelligent Autonomous Systems Group. Your computational resources and expertise were invaluable to my work.

I would also like to extend my thanks to every member of the Technical University of Darmstadt for contributing to my academic journey and growth in my field.

This endeavor wouldn't have been possible without the unconditional love and support of my parents, Selahattin and Kerime. I cannot thank you enough.

Last but not least, I'm deeply grateful to close friends Cemil Emre Ardıç and Arda Civan for their invaluable assistance and steadfast support throughout this journey.

Abstract

In the domain of robotics and artificial intelligence, imitation learning has garnered attention for enabling agents to acquire skills by mimicking experts. A central challenge is the "correspondence problem," which arises when learner and expert agents have different physical properties. This challenge is particularly salient when the learner and expert possess varying morphologies and dynamic constraints, making it difficult for the learner to imitate the expert effectively. To address this, the thesis proposes a new approach where embodiments are represented as structured graphs. The correspondence problem is then tackled as an inexact Graph Matching (GM) problem, providing a mathematical framework for solving it. Through experiments, this research demonstrates the approach's effectiveness. By providing a structured solution to the correspondence problem, the research contributes to enhancing the efficiency and adaptability of robotic and artificial agents, facilitating more effective learning from human expertise.

Zusammenfassung

Im Bereich der Robotik und künstlichen Intelligenz hat das Imitationslernen Aufmerksamkeit erregt, indem es Agenten ermöglicht, Fähigkeiten durch das Nachahmen von Experten zu erwerben. Eine zentrale Herausforderung besteht im "Korrespondenzproblem,"das auftritt, wenn Lernende und Experten unterschiedliche physische Eigenschaften haben. Diese Herausforderung ist besonders ausgeprägt, wenn Lernende und Experten unterschiedliche Morphologien und dynamische Einschränkungen haben, was es dem Lernenden erschwert, den Experten effektiv nachzuahmen. Um dies zu lösen, schlägt die Arbeit einen neuen Ansatz vor, bei dem Verkörperungen als strukturierte Graphen dargestellt werden. Das Korrespondenzproblem wird dann als inexaktes Graphenabgleichsproblem (GM) behandelt, das einen mathematischen Rahmen für seine Lösung bietet. Durch Experimente wird die Effektivität dieses Ansatzes demonstriert. Durch die Bereitstellung einer strukturierten Lösung für das Korrespondenzproblem trägt die Forschung zur Steigerung der Effizienz und Anpassungsfähigkeit von robotischen und künstlichen Agenten bei und erleichtert ein effektiveres Lernen von menschlicher Expertise.

Contents

Figures

List of Figures

1 Introduction

In the vast field of robotics, there's an underlying drive to make machines mimic human behavior to learn from us, and in some instances, perform even better. Over the years, imitation learning has emerged as a leading paradigm, striving to bridge the gap between robotic capabilities and human expertise [1]. From the breathtaking intricacies of helicopter acrobatics [2] to the subtle nuances of haptic control [3], imitation learning has showcased a wide array of accomplishments.

Behavioral Cloning (BC) and Inverse Reinforcement Learning (IRL) have been two primary algorithmic approaches in this domain, with each bringing its assumptions about the correspondence between the learner (robot) and the expert (human) [4]. For instance, BC-based approaches often require similarity in the embodiments of the learner and the expert for successful imitation, which can be a restrictive condition [5, 6]. In contrast to methods that require environmental interaction, BC enables immediate imitation of the demonstrator without such a requirement. It uses observed states and actions from an expert to train a classifier or regressor, directly mapping these inputs to replicate the expert's policy[7]. On the other hand, IRL methods are computationally expensive and involve complex reinforcement learning loops [8, 9]. IRL focuses on learning a cost function from the expert's behavior and subsequently applying reinforcement learning to achieve similar tasks [7].

The correspondence problem is central to imitation learning. It seeks to establish corresponding states between the expert and learner, especially when their embodiments differ in terms of morphology, degrees of freedom, dynamics, and more[10]. Many existing approaches in imitation learning sidestep the correspondence problem by using methods like kinesthetic teaching or teleoperation[11]. However, these methods often require robot-specific proprioceptions, which might not always be available. Despite the pivotal role it plays, the correspondence problem in imitation learning remains an open question, drawing sporadic attention but lacking comprehensive solutions. Addressing this gap forms the main contribution of this research.

While existing methods have explored the correspondence problem to some extent, there remains a gap in effectively adapting the solutions for scenarios involving vastly different morphologies. This issue delves into the complexities of aligning the varying physical characteristics and capabilities between humans and robots [12]. Distinct state spaces, body morphology, degrees of freedom, and even system dynamics create a complex challenge in morphology correspondence that is yet to be fully addressed. Teleoperation techniques have also been employed to address the correspondence issue by allowing a human operator to control a robot remotely. Yet, traditional teleoperation often bypasses the correspondence problem by using interfaces such as joysticks, haptic controls, or even Virtual Reality [13, 14, 15, 16].

The study delves into the correspondence problem by representing the embodiment as a structured graph. It aims to derive a similarity measure on these graphs using some isometry heuristics. The correspondence problem, in its simplest form, can be defined by finding actions so that the learner's state is similar to the expert's state for all timesteps.

The main objective is to investigate the correspondence problem in detail, propose a proper distance metric between embodiments, and then train an imitation policy using the proposed distance as reward signals. Motivated by this, the primary objective of this research is to provide an innovative solution to the correspondence problem. By interpreting the embodiment as a structured graph and employing similarity metrics grounded in isometric heuristics, we aim to present a nuanced understanding of the intricate alignment needed between human and robotic systems.

One significant contribution of this work is introducing a representation of embodiment as a structural graph. By pairing this with feature functions tailored for graph comparison, we create a potent tool for addressing the correspondence problem. Our approach does not demand exact matching of morphological features but instead seeks a functionally equivalent correspondence that could similarly perform the task, thus expanding the scope and applicability of imitation learning.

In the subsequent chapters, this thesis provides a comprehensive analysis of imitation learning, detailing the experiments and methodologies used in this research. The objective is to present a systematic solution to the correspondence problem, thereby advancing the capabilities of robots in the realm of imitation learning.

Overview

The structure of this thesis is segmented into six primary chapters, each dedicated to elucidating various facets of leveraging structured-graph correspondence in imitation

learning.

In Chapter 2, a comprehensive dive into the fundamental concepts is undertaken. The discourse begins with the broader robotics domain, followed by an in-depth exploration of reinforcement learning (RL). A notable sub-section of this chapter is dedicated to inverse reinforcement learning, a key concept in imitation learning. The chapter concludes with a summary of pivotal literature, contextualizing the current state of the art against the backdrop of this research.

Chapter 3 delineates the methodology and approach. The chapter sets out to formulate the problem statement and hypothesis. This is complemented by a detailed exposition of the data sources and tools used, ensuring the replicability of the experiments. The chapter culminates in outlining the procedure that forms the bedrock of the ensuing experiments.

The subsequent Chapter 4 is dedicated to the experiments and their consequent results. A meticulous breakdown of the experimental setup precedes the main findings, providing readers with both the context and the outcomes of the research efforts.

Chapter 5 delves into a discussion phase. The results, while empirically obtained in the previous chapter, are theoretically examined here. The interpretations of the findings, their broader significance, any lurking limitations, and prospective avenues for future research are dissected in this chapter. It serves as a bridge between the empirical and the theoretical, weaving them into a cohesive narrative.

Finally, Chapter 6 distills all the insights, findings, and discussions into a succinct conclusion. It not only reflects on the journey traversed in the thesis but also casts a gaze into the future, speculating on the potential advancements in the domain of imitation learning leveraging structured-graph correspondence.

2 Fundamentals and Literature Review

The foundation of any scientific inquiry lies in a profound understanding of the theoretical principles that govern the subject of study. Therefore, this chapter seeks to immerse the reader in the foundational knowledge necessary to appreciate the intricacies of the work presented in this thesis. We embark on a detailed exploration of the interwoven fields of robotics, reinforcement learning, and inverse reinforcement learning as applied to the correspondence problem in imitation learning.

In the following sections, the reader will be introduced to the vast landscape of robotics, beginning with a foundational understanding of what constitutes robotics and delving into the essential aspects of robot kinematics.

As we transition to the subject of Reinforcement Learning (RL), the discussion will focus on the core principles that define this groundbreaking approach to machine learning. By explicating various policy gradient methods and diving into specific applications of Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO), we aim to create a thorough comprehension of the mechanics of RL.

To complete the theoretical framework, we will take a discerning look at the existing literature, drawing from seminal works and recent developments that have shaped the field. Special attention will be paid to correspondence methodologies that form the core of the research problem tackled in this thesis.

2.1 Robotics

Robotics, as an academic and practical discipline, is a fusion of science, engineering, and technology that focuses on the design, construction, operation, and application of robots. These robots are autonomous or semi-autonomous machines that can perform tasks in the real world, often replicating human actions. The field of robotics is vast, encompassing

various sub-disciplines and specializations, each with its unique challenges and innovations. As technology advances, robotics plays an increasingly pivotal role in various sectors, from manufacturing and healthcare to entertainment and space exploration[17].

2.1.1 Introduction to Robotics

Robotics, an interdisciplinary domain, encompasses the design, construction, and operation of robots. These are autonomous or semi-autonomous machines adept at performing tasks in both real-world and virtual environments. The field has evolved from ancient attempts to mimic human motions in machines to the modern-day marvels that augment, automate, and innovate various human activities[17].

Central to robotics are the concepts of kinematics and dynamics. Kinematics focuses on the motion of bodies without delving into the forces causing such motion. It provides insights into how robots move based on their mechanical structure. Dynamics, on the other hand, delves deeper into the forces and torques that influence motion. A fundamental equation in dynamics is Newton's second law of motion, which can be represented as:

$$F = ma \tag{2.1}$$

where F is the force applied, m is the mass of the body, and a is its acceleration. This equation underscores the relationship between force and motion, a cornerstone in understanding robotic movements. The law is particularly vital for designing robots that can interact effectively with their environment, as it aids in predicting how a robot will respond to various forces and torques[18].

The degree of freedom (DOF) of a robot is pivotal, indicating the number of independent movements it can execute. It is influenced by the robot's joints and links. Mathematically, the DOF can be represented as:

$$\text{DOF} = 6n - m \tag{2.2}$$

where n is the number of links and m is the number of constraints. This concept is crucial as it directly influences the robot's workspace and potential configurations, determining its versatility and adaptability in various environments[18].

Control architecture ensures robots operate safely and efficiently. The relationship between the control input u and the system output y can be represented as:

$$\boldsymbol{y}(t) = \mathbf{G}(s) \times \boldsymbol{u}(t), \tag{2.3}$$

where $G(s)$ is the transfer function in the Laplace domain. Furthermore, the control of a robot can be described by:

$$\boldsymbol{u} = K_p\boldsymbol{e} + K_d\dot{\boldsymbol{e}} + K_i \int \boldsymbol{e}\, dt, \tag{2.4}$$

where $\boldsymbol{u}$ is the control input, $\boldsymbol{e}$ is the error, $\dot{e}$ is the rate of change of error, and K_p, K_d, and K_i are the proportional, derivative, and integral gains, respectively[17].

A robot's position and orientation in space are vital for its control. This can be mathematically described using vectors, transformation matrices, and quaternion representations. For instance, a vector's position from point A to point B is[18]:

$$\vec{\boldsymbol{x}}_{\text{AB}} = (B_x - A_x)\boldsymbol{i} + (B_y - A_y)\boldsymbol{j} + (B_z - A_z)\boldsymbol{k}. \tag{2.5}$$

Robotics stands at the forefront of technological innovation, with applications spanning manufacturing, healthcare, space exploration, and entertainment. Their precision and adaptability render them invaluable for tasks that are dangerous, repetitive, or intricate for humans. As technology advances, the capabilities of robots expand, heralding a future of seamless collaboration between robots and humans. In this thesis context, the fusion of robotics with machine learning techniques, such as reinforcement learning and imitation learning, is of paramount significance. Integrating these fields promises to unlock new potentials, pushing the boundaries of what robots can achieve in various domains[17].

2.1.2 Kinematics in Robotics

Kinematics in robotics is a fundamental study that focuses on understanding the motion of robots without considering the forces that produce such motion. It provides insights into how the mechanical structure of a robot translates into its movement in space.

Configuration Space

The configuration space, often denoted as C, represents all possible positions and orientations a robot can assume. For a robot with n joints, its configuration space is n-dimensional. Each point in this space corresponds to a unique pose of the robot. For instance, a robot arm with three rotating joints would have a configuration space represented by three angles. The configuration space is crucial as it aids in understanding the robot's movement capabilities and is essential for tasks like path planning and collision avoidance [17].

Rigid-Body Motions

Rigid-body motion describes the movement of a solid object in space. This motion can be decomposed into rotational and translational components. The rotation in 3D space can be represented using rotation matrices R, which transform coordinates from one frame to another. A rotation matrix property is that its transpose is its inverse:

$$R^T = R^{-1} \tag{2.6}$$

Angular velocities, denoted by ω, describe how fast a body rotates and can be represented in a skew-symmetric form. Twists combine angular and linear velocities, offering a unified representation of rigid-body motion [19].

Forward and Inverse Kinematics

Forward kinematics determines the position and orientation of the robot's end-effector based on its joint parameters. Given joint angles $\vec{\boldsymbol{\theta}}$, the position $\vec{\boldsymbol{x}}$ of the end-effector can be determined using:

$$\vec{\boldsymbol{x}} = f(\vec{\boldsymbol{\theta}}) \tag{2.7}$$

Inverse kinematics, on the other hand, finds the joint parameters required for the end-effector to achieve a specific position and orientation. For a desired position $\vec{\boldsymbol{x}_d}$, the joint angles can be found as:

$$\vec{\boldsymbol{\theta}} = f^{-1}(\vec{\boldsymbol{x}_d}) \tag{2.8}$$

While forward kinematics generally has a unique solution, inverse kinematics can have multiple, one, or no solutions, making it computationally more challenging [17].

Forward Kinematics Calculations

Forward kinematics refers to the computational process of determining the position and orientation of a robot's end-effector given its joint parameters. For a multi-link robotic manipulator, the position and orientation of the end-effector are functions of the joint angles and link lengths.

In its most basic form, for a planar robot with n links, the position $\boldsymbol{x} = [x, y]$ of the end-effector in the Cartesian plane is given by:

$$x = \sum_{i=1}^{n} L_i \cos\left(\sum_{j=1}^{i} \theta_j\right) \tag{2.9}$$

$$y = \sum_{i=1}^{n} L_i \sin\left(\sum_{j=1}^{i} \theta_j\right) \tag{2.10}$$

where L_i represents the length of the i^{th} link and θ_i is the i^{th} element of the joint angle vector $\boldsymbol{\theta}$. The orientation of the end-effector, represented by ϕ, is the cumulative sum of the joint angles[17]:

$$\phi = \sum_{i=1}^{n} \theta_i \tag{2.11}$$

In my thesis work, I employed forward kinematics calculations to obtain correspondences and achieve their matching. The forward kinematics is computed statelessly for all links. Given joint configurations represented by the tensor q, the function processes this input to determine and store the pose of each link within a dictionary. This design ensures modularity, enabling scalable computations that accommodate robotic manipulators with a diverse range of degrees of freedom and intricacies.

2.2 Reinforcement Learning (RL)

Reinforcement Learning, commonly called RL, has emerged as a powerful paradigm for optimizing decision-making tasks in uncertain environments. At its core, RL provides mechanisms for agents to learn from interactions and subsequently improve their actions over time. This section delves into the foundational concepts, methodologies, and significance of RL, providing a holistic view of its intricacies and applications. The following diagram depicts the feedback loop between actions and rewards in a standard reinforcement learning model.

Figure 2.1: Standard RL model loop

2.2.1 Introduction to RL

Reinforcement Learning (RL) is a pivotal domain within machine learning, emphasizing the art and science of decision-making in sequential scenarios. At its core, RL is characterized by an agent's continuous interaction with an environment, striving to make decisions that maximize a cumulative reward over an extended period. This dynamic interplay encapsulates fundamental tenets of artificial intelligence, weaving in concepts of cause and effect, uncertainty, exploration versus exploitation, and the challenges of operating in nondeterministic settings[20].

The RL paradigm is distinct from other machine learning approaches. Unlike supervised learning, where the correct answers are provided, or unsupervised learning, which seeks patterns in data, RL is about learning from interaction. The agent learns from the consequences of its actions rather than from being explicitly taught, making it a powerful tool for tasks where the correct decision or action isn't known in advance. This trial-and-error search, combined with delayed reward, makes RL a robust framework for a myriad of applications, from game playing to robotics and from healthcare to finance[20].

The essence of RL lies in the balance of exploration, where the agent tries new actions to discover their effects, and exploitation, where the agent chooses actions that it knows have favorable outcomes. This duality is a recurring theme in RL and poses both challenges and opportunities for designing algorithms that can learn efficiently and effectively from interaction[20].

In the broader context, RL has been instrumental in various applications, from robotics to finance. Its ability to model decision-making under uncertainty makes it a powerful tool for numerous complex tasks. However, the field also faces challenges, especially

when dealing with large-scale problems, which often require sophisticated algorithms and computational techniques[21].

Markov Decision Processes

Markov Decision Processes (MDPs) are foundational tools for modeling sequential decision-making scenarios where an agent interacts with a system over time. At its core, an MDP is characterized by states, actions, and transitions, with the agent making decisions at each state based on a given policy. The outcome of each decision is probabilistic, leading to transitions between states and associated rewards.

An MDP is a mathematical framework that captures the essence of decision-making in environments with stochastic outcomes and can be formally defined by a tuple $M := \{S, A, r, P, \gamma\}$:

- S: Represents the finite state space.
- A: Denotes the finite action space of the protagonist.
- r: The reward function, mapping state and action pairs to rewards.
- P: The state transition probability function.
- γ: The discount factor.

In the context of MDPs, the agent's objective is to maximize the cumulative reward over time. This is achieved by following a policy, which is a mapping from states to actions. The quality of a policy is often measured by the expected total reward when following that policy from a given state. The optimal policy is the one that yields the highest expected reward[22].

A key property of MDPs is the Markov property, which asserts that the future state of the system depends only on the current state and action and not on the sequence of states and actions that preceded it. This property simplifies the analysis and computation of optimal policies.

The primary goal of an MDP is to identify a policy, denoted as π, that chooses the best action based on the current state in order to optimize the accumulated rewards over time. This can be formally expressed as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} r_t \mid \pi\right] \tag{2.12}$$

where π^* represents the optimal policy and r_t denotes the reward at time t.

MDPs have found widespread applications in various domains, from robotics to finance, due to their ability to model complex decision-making problems under uncertainty[23]. The intrinsic architecture of Markov Decision Processes (MDPs) lends itself well to utilizing dynamic programming methods. These approaches leverage the Bellman equation to solve the underlying optimization problem iteratively. Specifically, they compute the value function and then extract the optimal policies based on this function[22].

In practice, while the theoretical foundations of MDPs are well-established, solving large-scale MDPs remains a challenge due to the curse of dimensionality. However, advancements in algorithms and computational techniques continue to push the boundaries of what is achievable with MDPs[23].

2.2.2 Policy Gradient Methods

Policy Gradient Methods are a subset of reinforcement learning techniques that focus on directly optimizing a parameterized control policy using gradient descent. This approach is distinct from traditional value function approximation methods, which derive policies from a value function [24].

A policy gradient method is defined as a reinforcement learning strategy that directly optimizes a parametrized control policy through gradient descent. It falls under the category of policy search techniques that aim to maximize the expected return of a policy within a fixed policy class [24].

Policy gradient is a technique used in reinforcement learning, aiming to model and optimize the agent's policy directly. Reinforcement learning seeks an optimal behavior strategy for the agent to achieve the highest cumulative reward. Instead of focusing on value estimation as many other approaches do, policy gradient methods optimize the policy, a function often parameterized by θ [25].

The reward (objective) function, in the context of the policy gradient, is defined by:

$$J(\mathbf{Q}) = \sum_{s \in \mathbf{S}} d^{\boldsymbol{\pi}}(s) V^{\boldsymbol{\pi}}(s) = \sum_{s \in \mathbf{S}} d^{\boldsymbol{\pi}}(s) \sum_{a \in \mathbf{A}} \boldsymbol{\pi_\theta}(a|s) \mathbf{Q}^{\boldsymbol{\pi}}(s,a) \tag{2.13}$$

where $d^{\pi}(s)$ represents the stationary distribution of Markov chain for $\boldsymbol{\pi_\theta}$. The primary objective becomes the adjustment of θ such that the policy $\boldsymbol{\pi_\theta}$ maximizes this reward function[25].

Policy Gradient Methods allow for directly incorporating domain knowledge into policy parameterization. Often, fewer parameters are required to represent the optimal policy than the corresponding value function. They are guaranteed to converge to at least a locally optimal policy and can handle continuous states and actions, and often even imperfect state information [24]. However, they are challenging to use in off-policy settings, exhibit slow convergence in discrete problems, and global optima are not always attained [24].

In continuous spaces, policy-based methods often outshine value-based approaches due to the infinite number of actions or states, which can make value estimation computationally intensive [25]. By utilizing gradient ascent, θ can be adjusted in the direction suggested by the gradient to identify the optimal θ for π_θ that yields the highest reward.

Over the past few years, the landscape of policy gradient algorithms has expanded considerably, with many methods being introduced. However, amidst this vast array of options, this thesis delves explicitly into PPO and TRPO. These methods were selected not only for their prominence but also because they formed the cornerstone of my research investigations.

Trust Region Policy Optimization (TRPO)

The stability of training in reinforcement learning can be enhanced by ensuring incremental changes to the policy. TRPO adopts this strategy, and it implements a constraint on policy updates using the KL divergence, ensuring that significant alterations don't occur suddenly.

In off-policy reinforcement learning scenarios, there's a distinction between the policy for gathering trajectories and the one under optimization. Here, the objective function gauges the total advantage over state visitation distribution and actions. The disparity between the distribution of training data and the actual policy state distribution is balanced using an importance sampling estimator. In this context, θ_{old} refers to the policy parameters prior to updating, and the estimated advantage, $A_{\theta_{\text{old}}}$, is utilized because true rewards are typically elusive[25].

Even in on-policy training, asynchronous operations can cause a lag between the policy for data collection and the desired optimized policy. TRPO addresses this nuance by

designating the behavior policy as $\pi_{\theta_{old}}$. The resultant objective function is given by:

$$J(\theta) = E_{s\sim\rho_{\theta_{old}},a\sim\pi_{\theta_{old}}}\left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} \times A_{\theta_{old}}(s,a)\right] \tag{2.14}$$

The goal of TRPO is to amplify this objective function while ensuring the trust region constraint is satisfied. This constraint demands that the divergence between the old and updated policies, measured by KL-divergence, remains limited, typically within a defined parameter δ. Thus, TRPO maintains policy coherence while assuring a consistent enhancement over iterations. For a comprehensive understanding and proof, readers are directed to the original paper.

Trust Region Policy Optimization (TRPO) is an iterative procedure designed for optimizing policies with guaranteed monotonic improvement. The core idea behind TRPO is to make several approximations to a theoretically justified procedure, resulting in a practical algorithm that is effective for optimizing large nonlinear policies, such as neural networks [26].

TRPO operates by optimizing a local approximation to the expected return of the policy with a Kullback-Leibler (KL) divergence penalty. This approach ensures that the updated policy does not deviate too much from the old policy, thereby maintaining stability in the optimization process. The algorithm is scalable and can optimize policies with many parameters, which has been a challenge for model-free policy search in the past [26].

The foundation of TRPO lies in optimizing a certain surrogate objective function. This optimization guarantees policy improvement with non-trivial step sizes. The algorithm employs trust region methods designed to ensure that each update to the policy does not change the policy too drastically. This is achieved by constraining the KL divergence between the new and old policies to be below a certain threshold [26].

TRPO has demonstrated robust performance across a variety of tasks. For instance, it has been used to learn complex policies for robotic locomotion tasks such as swimming, hopping, and walking. Additionally, TRPO has been applied to play Atari games using raw image inputs, showcasing its versatility and capability to handle high-dimensional input spaces [26].

Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a novel family of policy gradient methods designed for reinforcement learning. These methods alternate between sampling data through

interaction with the environment and optimizing a "surrogate" objective function using stochastic gradient ascent. Unlike standard policy gradient methods that perform one gradient update per data sample, PPO introduces a unique objective function that allows for multiple epochs of minibatch updates. This results in an algorithm that is simpler to implement than trust region policy optimization (TRPO), yet it retains the benefits of TRPO in terms of performance and sample complexity [27]. Proximal Policy Optimization (PPO) was introduced as a more streamlined counterpart to the Trust Region Policy Optimization (TRPO) approach. While PPO simplifies implementation through its clipped surrogate objective, it maintains impressive performance metrics akin to TRPO[25].

The core principle behind PPO is the optimization of a surrogate objective function. This function ensures that the updated policy does not deviate significantly from the previous policy, thereby maintaining stability in the optimization process. The algorithm employs a mechanism that clips probability ratios, forming a pessimistic estimate of the policy's performance. By alternating between sampling data and performing several epochs of optimization on this data, PPO achieves a balance between efficiency and robustness [27].

The probability ratio is defined as:

$$r_t(\theta) = \frac{\boldsymbol{\pi_\theta}(a_t|s_t)}{\boldsymbol{\pi_{\theta_{old}}}(a_t|s_t)} \tag{2.15}$$

where $r_t(\theta_{old}) = 1$.

The objective function of PPO is designed to minimize the discrepancy between its original and clipped values, thereby countering extreme policy updates even when higher rewards are at stake. Integrating PPO within a neural network framework, where both policy (actor) and value (critic) functionalities share parameters, leads to the objective function being governed by the clipped reward, complemented by an error term tied to value estimation and an entropy term that promotes exploration.

The surrogate objective for TRPO is defined as:

$$L^{CPI}(\theta) = \hat{E}_t \left[r_t(\theta) \hat{a}_t \right] \tag{2.16}$$

The objective function in TRPO (for on-policy scenarios) can be represented as $L_{TRPO}(\theta)$. However, the unconstrained maximization of $L_{TRPO}(\theta)$ can lead to extreme policy ratios, resulting in unstable conditions. PPO stabilizes this by constraining $r(\theta)$ to lie within a vicinity of 1, specifically in the range $[1 - \epsilon, 1 + \epsilon]$, where ϵ is a predefined hyperparameter[25].

The primary objective introduced for PPO is:

$$L^{CLIP}(\theta) = \hat{E}_t \left[\min \left(r_t(\theta)\hat{a}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{a}_t \right) \right] \tag{2.17}$$

where ϵ is a hyperparameter, commonly set to $\epsilon = 0.2$[27].

PPO operates based on a surrogate objective that maximizes the expected return of a policy while constraining the size of the policy update. This is achieved by using the Kullback-Leibler (KL) divergence to ensure that the new policy remains close to the old policy. The algorithm's foundation lies in optimizing this surrogate objective, which guarantees policy improvement with non-trivial step sizes. The use of trust-region methods ensures that each policy update remains within a certain threshold, preventing drastic changes to the policy [27].

Another approach is to use a penalty on KL divergence and adapt the penalty coefficient to achieve a target value of the KL divergence d_{targ} for each policy update. The KL-penalized objective is:

$$L^{KLPEN}(\theta) = \hat{E}_t \left[\frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_{old}}(\mathbf{a}_t|\mathbf{s}_t)} \hat{\mathbf{a}}_t - \beta \text{KL}[\pi_{\theta_{old}}(\cdot|\mathbf{s}_t), \pi_\theta(\cdot|\mathbf{s}_t)] \right] \tag{2.18}$$

There are two fundamental design aspects of PPO: firstly, the use of the clipped probability ratio for policy regularization, and secondly, the characterization of the policy action space through either continuous Gaussian or discrete softmax distributions. With this exploration, three potential pitfalls are inherent to standard PPO implementations.

1. In continuous action spaces, PPO might exhibit instability when rewards diminish outside established boundaries.
2. For discrete action spaces characterized by infrequent high rewards, PPO may stagnate at suboptimal action sequences.
3. Policy outcomes are prone to variations during initialization, especially if optimal actions are proximal to the initial state.

To circumvent these issues, one could either discretize the action space or employ the Beta distribution as an alternative to the Gaussian policy, addressing the first and third challenges. For the first and second challenges, KL regularization, reminiscent of the motivation behind TRPO, offers a viable solution [25].

PPO has demonstrated its effectiveness across various tasks. For instance, it has been applied to robotic locomotion tasks and has outperformed other online policy gradient

methods in environments like the Arcade Learning Environment. Its versatility and capability to handle high-dimensional input spaces make it a valuable tool in the realm of reinforcement learning [27].

Benchmark evaluations underscore the prowess of PPO, demonstrating its potent efficacy in tandem with its streamlined methodology. Consequently, I predominantly employed this technique to train the agents throughout my research.

2.2.3 Actor-Critic Methods

Actor-critic methods advance the conventional policy gradient techniques by seamlessly blending both policy and value function learning. Distinct from vanilla policy gradient methods that primarily target the policy model, Actor-Critic methods introduce a value function to streamline the policy update. By doing so, they harness the combined strengths of both policy-based and value-based approaches. These methods consist of two integral models: the Actor, which proposes actions based on the current policy, and the Critic, which assesses these actions through value estimation. With the Critic's feedback, the Actor refines its policy, promoting a more informed and efficient learning process [25, 28].

- **Actor:** Responsible for updating the policy parameters, θ, based on the direction provided by the critic.
- **Critic:** Focuses on updating the value function parameters, w. Depending on the specific algorithm, this could be either the action-value $Q(s, a)$ or the state-value $V(s)$.

A typical action-value Actor-Critic algorithm operates as follows:

1. Initialize policy parameters, θ, randomly and sample an initial state, s_0.
2. For each timestep t:
    - Sample reward r_t and the subsequent state s_{t+1}.
    - Sample the next action a_{t+1}.
    - Update the policy parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta \ln \pi_\theta(a_t|s_t) Q_w(s_t, a_t)$.
    - Compute the Temporal Difference (TD) error for the action-value at time t: $\delta_t = r_t + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t)$.
    - Update the action-value function parameters: $w \leftarrow w + \beta \delta_t \nabla_w Q_w(s_t, a_t)$.

- Update states and actions: $s_t \leftarrow s_{t+1}$ and $a_t \leftarrow a_{t+1}$.

Where α and β are predefined learning rates for the policy and value function parameter updates[25].

Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is a sophisticated version of the Actor-Critic approach, rooted in the maximum entropy reinforcement learning framework. In SAC, the Actor's objectives expand to not just maximizing the expected reward but also bolstering the entropy of the policy. Such a dual focus culminates in policies that balance exploration and exploitation, ensuring comprehensive exploration of the action space and mitigating premature convergence to less optimal strategies. A salient feature of SAC is its adeptness at efficiently navigating a myriad of tasks without the need for rigorous hyperparameter adjustments, distinguishing it from other reinforcement learning techniques. It finds its niche, especially in robotics and autonomous systems [25, 28].

Applications of SAC span across various domains, including robotics and autonomous systems, where the method's inherent capability to handle complex, high-dimensional environments is of paramount importance.

In conclusion, while the Actor-Critic methods, including SAC, provide a robust framework for reinforcement learning, the specific choice of method should always align with the complexities and requirements of the task at hand[28].

Imitation Learning (IL)

Imitation learning is a subset of methods that aim to reproduce desired behavior based on expert demonstrations. In many scenarios, the experts are human operators, and the learners are robotic systems. This technique facilitates the transfer of skills from humans to robotic systems. To implement imitation learning, a system is developed that records expert demonstrations and subsequently learns a policy to replicate the demonstrated behavior [4].

Imitation learning aims to emulate expert behavior through learned policies. Observations from either an expert or learner manifest as trajectories, denoted as $\tau = [\phi_0, ..., \phi_T]$, where ϕ represents features. These features, such as a robot's state or other measurements, can be as broad as raw image pixels. Demonstrations often occur under varying conditions,

identified by a context vector s, encapsulating task-specific information like initial robotic states or object positions. The context remains static during task execution, contrasting with the dynamic state features ϕ_t. Some scenarios also present an optimized reward signal r. Demonstrations accumulate into a dataset $D = \{(\tau_i, s_i, r_i)\}_{i=1}^{N}$, from which a policy π^* is derived through an optimization strategy:

$$\pi^* = \arg\min_D(q(\phi), p(\phi)) \tag{2.19}$$

where $q(\phi)$ is the expert-induced feature distribution, $p(\phi)$ the learner-induced one, and $D(q, p)$ a measure of similarity between q and p. The learning process can take place either offline or online, and when multiple tasks are demonstrated, it transitions into multitask learning. Furthermore, simulators or physical systems offer platforms for policy evaluation, interaction, and iterative enhancement[4].

The significance of imitation learning, especially in robotics, is profound. It is now considered a pivotal technology for manufacturing, elder care, and the service industry, where robots are expected to work with humans [4].

Behavioral Cloning (BC)

Behavioral Cloning (BC) serves as a potent approach in imitation learning, aiming at establishing a direct correlation between states or contexts and resultant actions or trajectories, circumventing the need to deduce the underlying reward function. The core advantage of BC lies in its ability to aptly replicate demonstrated behavior when such a direct association is the most succinct representation of the desired action sequence. Distinguishing between two main methodologies, model-free BC focuses on learning without relying on underlying system dynamics, whereas model-based BC capitalizes on this knowledge for enhanced learning.

Given an expert's demonstrations, BC aims to develop a policy that relates context to trajectories or links state (and context) to control input. Grounded in the assumption that an expert's demonstration dataset is accessible, BC models the learning challenge as a supervised regression problem, with the overarching objective being to approximate the expert's exhibited behavior[4].

2.3 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) focuses on recovering a reward function by observing a policy or demonstrating it. This technique, alternatively termed inverse optimal control, inverse planning, or structural estimation of MDPs, perceives the reward function as a concise way to specify the desired behavior. In the context of IRL, given measurements of an agent's behavior, sensory inputs, and a model of the physical environment, the challenge is to ascertain the reward function the agent is optimizing.

IRL operates under the assumption that the demonstrator's decision-making process follows a Markov decision process (MDP), which encompasses states, control inputs, state transition probabilities, a discount factor, an initial-state distribution, and a reward function.

The IRL objective is to determine the unknown reward function from expert trajectories. However, due to the potential for a policy to be optimal for several reward functions, identifying the exact reward function can be challenging. To address this ambiguity, multiple research efforts suggest optimizing additional objectives. Common methods of IRL typically follow an iterative learning routine that alternates between addressing an RL-based challenge and updating an estimate for the cost function. The goal of this process is to align the state-action frequency between demonstrated and learner-induced trajectories, thereby determining the policy and reward function parameters[4].

Data: Given expert trajectories $D = \{\tau_i\}_{i=1}^{N}$
Initialize parameters for reward w and policy θ;
while *not converged* **do**
    Compute the state-action visitation count μ for policy π_θ;
    Calculate the objective L and gradient $\nabla_w L$, contrasting μ against distribution from D;
    Refine reward parameter w;
    Adjust the policy parameter θ using RL techniques;
Result: Finalized policy θ and reward w parameters

Algorithm 1: Condensed Overview of Feature-Matching IRL

Comparison and differentiation from standard RL

Imitation learning is distinct from the more familiar supervised learning setting. In imitation learning, the features or states in a dataset of demonstrations are not necessarily drawn from the distribution of the features the learner will encounter using their own policy. This distinction means that the assumption of independent and identically distributed (i.i.d.) data is often violated in imitation learning [4].

Furthermore, imitation learning is closely related to reinforcement learning (RL). While RL aims to obtain a policy that maximizes an expected reward, imitation learning often assumes optimal expert demonstrations, which are not available in basic RL. These demonstrations provide prior knowledge that allows for more efficient methods, potentially leading to a significant decrease in sample complexity in learning a task by imitation rather than by trial-and-error reinforcement learning [4].

Discussion of seminal works

The field has seen significant advancements in addressing the correspondence problem. Modern approaches aim to find a balance between capturing the essence of the demonstrated task and adapting it to the robot's capabilities. This involves understanding both the spatial and temporal aspects of the demonstration and the robot's motion. Techniques have been developed to represent and compare the motions of different entities in a unified framework, allowing for more effective imitation learning.

The correspondence problem in imitation learning is a central challenge, especially when the expert and the learner have different embodiments. This issue arises when establishing corresponding states between two entities that differ in morphology, degrees of freedom, dynamics, and more. The problem revolves around how one can map the actions or states of one agent (typically a human expert) to another agent (like a robot) that might have a completely different structure or set of capabilities.

Historically, numerous approaches in imitation learning have often sidestepped the correspondence problem, either by focusing on scenarios where the expert and learner possess similar or identical embodiments or by employing methods that rely on robot-specific proprioceptions, such as kinesthetic teaching or teleoperation. However, with the advancement of robotics and artificial intelligence, there's been a heightened interest in confronting the correspondence problem directly. This is particularly evident when the demonstrator, a human, and the robot learner exhibit diverse morphologies or capabilities.

The contemporary literature has delved into various methodologies to address this challenge, striving to identify a correspondence that not only encapsulates the intrinsic intent of the demonstration but also remains adaptable to the robot's distinct characteristics.[29].

Recent developments

Imitation learning has witnessed significant advancements in recent years, particularly in addressing the correspondence problem. One innovative approach involves using structured representations, such as graphs, to encapsulate the embodiment of robotic systems. By deriving similarity measures between these representations, it's feasible to establish correspondence among robots with varied morphologies.

Another notable stride in the field is the introduction of adaptive policies and frameworks. These can determine correspondences based on the robot's interactions with its environment. Leveraging mathematical constructs like Riemannian metrics, these policies can prioritize different correspondences depending on the robot's current state and objectives[30].

A paramount challenge in this domain is ensuring that robots not only mimic tasks but also adhere to safety protocols, navigate around obstacles, and respect their physical boundaries. Contemporary research has introduced frameworks that amalgamate these considerations, guaranteeing both precise and safe imitation by the robot.

The emergence of these new policies and frameworks has amplified the efficacy of imitation learning across diverse robotic systems. Such progress not only boosts the efficiency of the learning process but also extends its range of applications. This paves the way for versatile robotic systems capable of learning from various experts.

In the broader spectrum of imitation learning, solving the correspondence problem is vital. It empowers robots and other artificial agents to learn from various experts, be it humans, other robots, or even virtual entities. As the discipline matures, solutions to the correspondence problem are expected to be instrumental in steering the future trajectory of imitation learning and robotics[29].

3 Methodology and Approach

In this chapter, we'll unpack the methodology driving our exploration into robotics and reinforcement learning. We begin by articulating the correspondence problem central to this study, followed by a succinct exposition of our hypothesis. The tools and frameworks will be described, setting the stage for our research design. From coding to correspondence matching, embodiment modeling, and experimental setups, this chapter offers a clear roadmap for our research.

3.1 Problem Statement and Hypothesis

This section focuses on solving the correspondence problem in the imitation learning problem. We aim to define the problem, propose a hypothesis for its resolution, and present a structured approach for testing the hypothesis.

3.1.1 Defining the correspondence problem

The correspondence problem is a fundamental challenge in computer science, especially in pattern recognition, computer vision, graphics, and bioinformatics. It revolves around finding an optimal correspondence between the vertices of graphs to minimize or maximize their node and edge disagreements (affinities) [29]. This problem is essential because it relates to many real-world applications where correspondence and similarity between graphs are required. One of the main advantages of using graphs to describe relational information, as opposed to vectors, is that graphs offer a more powerful representation of structural relations. In most real-world scenarios, nodes and edges in these graphs are assigned with arbitrary attributes, making this modeling approach flexible and widely applicable.

In the context of imitation learning and reinforcement learning, addressing the correspondence problem is crucial. It enables robots and artificial agents to learn from various experts, including humans, other robots, or even virtual agents. By encoding geometrical cues in the graph representation and the matching process, graph-matching methods can generally find better correspondence than point-based registration methods, such as RANSAC or Iterative Closest Point (ICP) [29]. This is especially true when dealing with agents or robots with different morphologies or capabilities.

For this study, the correspondence problem was addressed by leveraging the Unified Robot Description Format (URDF). The URDF files were transformed into structured graphs, which encapsulated the essence of each robot's embodiment. This paved the way for correspondence-matching techniques, enabling the establishment of relationships between different robotic systems or agents via graph representations. By tackling the correspondence problem head-on, it becomes feasible to bridge the gap between diverse agents, ensuring that the underlying intent of demonstrations is captured and adapted to the unique characteristics of the learner [29].

3.1.2 Research Outline

This subsection outlines our methodology for tackling the correspondence problem, focusing initially on defining the graph structure that represents various robotic embodiments.

1. Graph Transformation:

The primary phase involves the transformation of Unified Robot Description Format (URDF) files into structured graphs. These graphs encapsulate the essence of each robot's embodiment, ensuring a comprehensive representation that can cater to various robotic architectures and embodiments.

- *Data Collection:* URDF files for various robotic agents were collected to ensure diversity in embodiments.
- *Graph Formation:* A handcrafted algorithm was developed to transform URDF files into structured graphs, with each node representing a joint and edges representing the distances between these joints.

2. Correspondence Matching:

Once the graphs are generated, the next step focuses on finding the optimal correspondence between them.

- *Feature Extraction:* For each graph, features that include geometric and kinematic cues are extracted. These features serve as the basis for matching.
- *Graph Matching Algorithm:* A graph matching algorithm is applied to find the best correspondence between the graphs. The objective is to minimize the distortion and maximize the affinity between graphs.

3. Imitation Learning Setup:

The established correspondences are then used to set up imitation learning scenarios where agents can learn from diverse experts.

- *Experiment Design:* Several experiments were designed where robotic agents, using the established correspondences, tried to imitate expert movements or tasks.
- *Evaluation Metrics:* Performance metrics were identified to evaluate how closely the agents could imitate the experts. This includes quantifying the difference between the movements, the accuracy of task completion, and the time taken.

4. Analysis and Refinement:

Post-experimentation, the results were analyzed to evaluate the effectiveness of our approach.

- *Results Compilation:* The outcomes of the experiments were compiled, quantifying the agents' performance.
- *Refinement:* Based on the results, iterative refinements were made to the graph transformation and matching algorithms to improve performance.

In conclusion, the research outline provides a structured pathway, from graph formation to imitation learning, ensuring that the correspondence problem is tackled holistically. Through iterative processes and robust evaluation, we aim to validate the proposed hypothesis and contribute to the field of robotics and reinforcement learning.

3.1.3 Hypothesis about the Optimal Correspondence

The correspondence problem in imitation learning, especially when considering different embodiments, remains a significant challenge. This problem arises when establishing corresponding states between an expert and a learner, especially when their embodiments differ in terms of morphology, degrees of freedom, and dynamics, among other factors.

The core of this problem is how an agent (the learner) can replicate a behavior observed in another agent (the expert) when they operate under different kinematic and dynamic constraints.

To address this, embodiments are represented as structured graphs, with the correspondence problem formulated as an inexact Graph Matching (GM) problem[29]. The correspondence can be represented as an injective function mapping nodes between embodiment graphs based on certain objectives. The inexact GM aims to find correspondence with minimum distortion, maximizing affinity between graphs. The embodiment metric is then defined as the sum of corresponding global feature distances, essentially measuring the distance sum of corresponding joint poses between embodiments.

The hypothesis is that the correspondence problem in imitation learning can be formulated as this inexact GM problem, where a bijective map between embodiment nodes might not always exist. This approach provides a structured and mathematical way to address the correspondence problem, offering a potential solution to one of the long-standing challenges in imitation learning[31, 32].

3.2 Data and Tools

This section details the data frameworks and tools foundational to this research. Beginning with an in-depth look into PyTorch and its application in reinforcement learning tasks, we segue into the nuances of neural network architectures, delving into their anatomy and operation. Further, we explore the essence of robot representation through the URDF and how it underpins the embodiment modeling. Finally, we introduce the specific robotic environments employed for experimentation, providing a setup on which our algorithms were tried and tested.

3.2.1 Overview

The landscape of machine learning has witnessed rapid advancements, particularly with the advent of robust libraries and hardware tailored for training and deployment pipelines. Among the prominent emerging frameworks is PyTorch, an imperative style, high-performance deep learning library [33]. PyTorch, coupled with Python and Numpy, offers a potent combination for robotics research, facilitating the development and deployment of sophisticated algorithms in a user-friendly environment.

TorchRL, a control library built on PyTorch, is a versatile tool for reinforcement learning and control tasks. It is designed to balance modularity and integration, making it especially suitable for complex, real-world data and environments. TorchRL introduces the TensorDict, a novel PyTorch primitive, which serves as a flexible data carrier. This design ensures seamless integration of various library components while preserving their individual modularity. Such a structure allows for replay buffers, datasets, distributed data collectors, environments, transforms, and objectives to be used either in isolation or in combination, providing researchers with unparalleled flexibility [34].

In the context of this research, TorchRL was particularly beneficial. Initially, the MushroomRL library was employed for experimenting with PPO and TRPO algorithms[35]. However, due to TorchRL's adaptability and the ease with which correspondence rewards could be defined, it became the library of choice. The ability to manipulate the PPO agent and define correspondence rewards was made significantly more convenient with TorchRL, underscoring its utility in reinforcement learning tasks.

Neural Network Architectures

Deep Reinforcement Learning (DRL) has emerged as a powerful approach for training agents to perform tasks by interacting with their environment. Central to the success of DRL is the use of deep neural networks, which provide the capability to process high-dimensional inputs, such as raw pixel data from images, and learn complex representations. These networks have achieved human-level performance in various tasks, notably in games like Atari 2600, where agents make decisions based solely on raw pixel values [36]. The rich representations offered by deep neural networks enhance the efficiency of reinforcement learning, allowing agents to generalize across different tasks and environments. The agent employed in this study is trained via the PPO algorithm. Within the PPO framework, neural networks are foundational for approximating the policy (actor) and value (critic) functions.

Shared Base Network: The primary layer of our agent's architecture is the Shared Base Network, constructed as a Multi-Layer Perceptron (MLP). It undertakes the role of processing state representations from the environment.

- **Architecture:** The MLP is constructed with two hidden layers, each containing 64 units.

- **Activation Function:** A Rectified Linear Unit (ReLU) is employed as the non-linear activation function.
- **Output:** Producing a shared feature vector, the output dimension is 256, serving as input for both actor and critic networks.

Actor-Network: The actor network uses the shared feature vector to produce a distribution over potential actions contingent on the nature of the action space.

- **Discrete Action Spaces:**
  - The network outputs logits corresponding to each potential action.
  - It utilizes an MLP, where the architecture's depth is externally defined. Typically, this comprises a single layer with units equivalent to the number of possible discrete actions.
- **Continuous Action Spaces:**
  - Output parameters fit for a Gaussian distribution, detailing the mean and variance.
  - An extra layer ensures these parameters fall within a predefined range.
  - The depth and unit count of each layer can be externally set. Generally, the MLP has one layer with units double the dimensionality of the action space to cater to both mean and variance values.

Critic Network: The critic network, integral for predicting the expected return from a particular state, derives its input from the shared feature vector. It maps this input to a singular scalar value, indicating the predicted value. The architecture of this network is modular, with layers whose number of units is determined by an external configuration. In the provided setup, the critic network often comprises a single layer, but its depth and width are adjustable, offering adaptability to diverse problem scenarios.

By adopting a shared base network, this architecture capitalizes on parameter efficiency. It enables the actor and critic networks to leverage common patterns in the data, streamlining their function approximations. The fully connected nature of these networks underscores a dense inter-neuronal connection, facilitating the complex function approximation requisite for reinforcement learning tasks.

3.2.2 Modeling of Embodiments

The Unified Robot Description Format (URDF) is an essential tool in robotics, offering a standardized method to represent a robot's geometry, kinematics, and dynamics. Developed as an XML-based format, URDF describes the kinematic structure, dynamic parameters, visual representation, and collision geometries of a robot [37]. This human-readable format delineates robot links (rigid bodies) and the joints connecting them using a tree structure.

A URDF model is a comprehensive XML file that captures the kinematic structure, dynamic parameters, visual representation, and collision geometries of a robot. The file may also reference other files containing 3D geometries of the robot's components[38]. The primary components of a URDF file are described as follows:

- **Robot Name:** Every URDF file begins by specifying the robot's name.
- **Links:** These represent the rigid bodies of the robot. Each link has a visual representation, which can be a simple geometric shape or a complex mesh.
- **Joints:** Joints define the connection between links. They specify the type of movement (e.g., continuous rotation) and the axis of rotation.

For instance, a simple 2 DoF planar robot's URDF file might define a base link, a visual representation using a box geometry, and a continuous joint connecting the base link to another link [38].

The URDF, or Unified Robot Description Format, constructs a robot's skeleton by hierarchically defining its links and joints. Each link represents a rigid body, and the joints define the spatial relationships and movements between these links. This hierarchical structure, often visualized as a tree, captures the robot's kinematic chain, starting from a base link and branching out to the various end-effectors and other components. The links and joints are described using XML tags, with attributes detailing their physical properties, visual representations, and collision geometries. This structured representation ensures that every aspect of the robot's physical structure, from its overall shape to the minutiae of its joint limits, is captured in a standardized format.

Regarding graph matching on the URDF-constructed skeleton, the tree structure inherent in the URDF can be transformed into a graph representation. Each node in the graph corresponds to a link in the robot, and the edges represent the joints connecting them. With this graph representation, graph-based algorithms can find correspondences between robot models. This is particularly useful in imitation learning, where the goal is to map

the movements of one robot (or human demonstrator) onto another robot with potentially different morphology. By finding the optimal node (link) correspondences between the two graphs, one can effectively determine how movements in the source robot should be translated to actions in the target robot, ensuring accurate and meaningful imitation.

In this research, URDF files were crafted for specific robotic environments: the inverted pendulum, double inverted pendulum, half cheetah, and ant. These URDF files not only provided a detailed model of each robot but also served as the foundation for subsequent graph-based representations. By transforming these URDF models into structured graphs, it became feasible to tackle the correspondence problem, a challenge central to this research. The graph representations encapsulated the essence of each robot's embodiment, paving the way for advanced correspondence-matching techniques[37].

In section 3.3.2, we utilize the URDF as the standard data format for constructing the embodiment graph, which lays the foundation for our imitation learning study.

3.2.3 Experimental Environments

In reinforcement learning, simulated environments play a key role. These environments, often governed by physics engines, provide a sandbox for agents to learn and refine their policies before deployment in real-world scenarios. The following environments were utilized in this study:

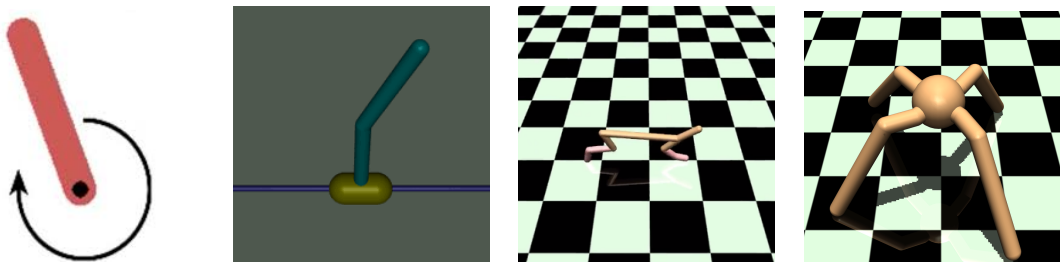
Inverted Pendulum: The Inverted Pendulum task is a fundamental control problem where the primary objective is to balance a pendulum upright by applying forces at its pivot point. This task is a fundamental benchmark in reinforcement learning due to its simplicity yet non-trivial dynamics. It is an initial testbed for many algorithms, offering insights into their stability and control capabilities[39].

Double Inverted Pendulum (IDP): An extension of the Inverted Pendulum, the Double Inverted Pendulum, or IDP, introduces an additional pendulum segment. This added complexity makes the task more challenging, requiring finer control to maintain balance. It's a testament to an algorithm's capability to handle increased system intricacies[39].

Half Cheetah: The Half Cheetah environment simulates a two-dimensional, planar, bipedal robot that mimics the motion of a cheetah. The primary goal in this environment is to make the cheetah run as fast as possible. It's a more complex task that tests an algorithm's proficiency in handling locomotion challenges, multi-joint coordination, and energy-efficient movement[39].

Ant: The Ant environment represents a quadrupedal robot navigating a two-dimensional plane. The objective is to guide the ant to move in various directions while maintaining stability. This environment is particularly challenging due to the multifaceted dynamics of a four-legged creature, demanding intricate coordination and balance[39].

These environments, depicted in Figure 3.1, serve as a diverse set of benchmarks that range from simple balancing tasks to complex locomotion challenges. Specifically, we utilize these settings to perform imitation learning between expert and learner pairs. For instance, the inverted pendulum and the double inverted pendulum serve as the expert-learner pair for one set of experiments. Another planned expert-agent pair for future work involves the half-cheetah and ant environments. These settings are instrumental in evaluating the robustness, adaptability, and efficiency of reinforcement learning algorithms in continuous control domains.



(a) Inverted Pendulum (b) Double Pendulum (c) Half Cheetah (d) Ant

Figure 3.1: Experimental gym environments

3.3 Approach and Procedure

To ensure the accurate and systematic translation of robotic motion into abstract skeletal structures and further into a trainable reinforcement learning model, a well-defined methodology was crucial. This chapter explains the step-by-step procedure we followed,

beginning with tool and library selection and culminating in the design of correspondence reward functions based on the correspondence hypothesis mentioned in the previous chapter. Each sub-phase of our approach was vital, establishing building blocks that collectively shaped the foundation and execution of our study.

3.3.1 Representation of Embodiment Using Structured Graphs

Robotic motion and interaction within an environment are fundamentally shaped by the spatial relationships and movements of their constituent parts. Abstracting these relationships into skeletal structures offers a concise representation of a robot's physical configuration and joint articulations. These skeletons serve as graphical representations of the robot's physical structure and joint relationships, enabling an intuitive understanding and manipulation of robotic motion.

Our choice of embodiment representation is motivated by the need to meet several requirements: (i) *generic* - the representation can be applied to diverse physical embodiments; (ii) *geometric-kinematic* - it encapsulates the shape and motion of the embodiment; (iii) *extrinsic* - features such as links or joint poses can be measured against a reference frame; and (iv) *metric-compatible* - it permits the introduction of a distance measure across varied embodiments with different DoFs.

Rigid Body State Representation

In robotics, a common approach to represent the kinematics of interconnected rigid links is through a Lie group representation using the special Euclidean group $SE(3)$[40]. A point on this manifold, represented as $\mathbf{T} \in SE(3)$, encapsulates the entire pose, both position and orientation, of a robot link frame or an object. Formally, this can be expressed as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^\intercal & 1 \end{bmatrix} \in SE(3)\,, \tag{3.1}$$

where $\mathbf{R} \in SO(3)$ is a rotation matrix with properties $\mathbf{R}\mathbf{R}^\intercal = \mathbf{R}^\intercal\mathbf{R} = \mathbf{I}$ and $\det \mathbf{R} = 1$. The column vector $\mathbf{p} \in \mathbb{R}^3$ describes the frame's translation with respect to a reference frame. Simplifying the notation, we often write $\mathbf{T} = [\mathbf{R}, \mathbf{p}]$.

Choice of metric in $SE(3)$. We define the distance measure between two points in $SE(3)$, $\mathbf{T}_1 = [\mathbf{R}_1, \mathbf{p}_1]$ and $\mathbf{T}_2 = [\mathbf{R}_2, \mathbf{p}_2]$, as a combination of a translational and a rotational component:

$$d_R(\mathbf{T}_1, \mathbf{T}_2) = \|\mathbf{p}_1 - \mathbf{p}_2\| + \|(\text{LogMap}(\mathbf{R}_1^\intercal \mathbf{R}_2))\|, \tag{3.2}$$

where LogMap$(\cdot)$ refers to the operator mapping an element from $SO(3)$ to its tangent space. In this context, while the first term uses the standard Euclidean norm between the two frame origins $\mathbf{p}, \hat{\mathbf{p}}$, the second term employs a bi-invariant Riemannian metric on $SO(3)$. Given that both terms in the equation are metrics, Equation 3.2 establishes itself as a metric on $SE(3)$[41].

Constructing Graphical Representations from Robot Models

Robot models, commonly provided in the Universal Robot Description Format (URDF), detail the links and joints of a robot. Using specialized classes, these models can be converted into a format suitable for further operations.

During the transformation process, each model link is examined to determine the degrees of freedom (DoFs) for movable joints while acknowledging fixed joints. This conversion process captures the key properties of individual rigid bodies, especially the parent-child relationships between links.

Further, we construct the embodiment graph by associating a node with each joint of the robot body. Physical links between joints dictate the edges. For an embodiment with M-DoFs and joint configuration $\mathbf{q} \in \mathbb{R}^M$, each node's joint pose is computed by $\mathbf{T}_i = l_v^i(\mathbf{q})$.

The embodiment can be represented as a structured graph $G = (V, E, l_v, l_s, l_e)$ comprising a node index set, edge set, and feature functions.

Each node $v \in V$ and edge $e \in E$ possesses distinct attributes and features:

Node Attributes:

- `pose`: The spatial position of the node.
- `var`: Uncertainty associated with the node's position.
- `name`: A unique identifier for the node.
- l_v: Maps a node to a joint pose with respect to a base frame.

- l_s: Computes local graph heuristics measuring the physical embodiment topology.

Edge Attributes:

- `length`: The Euclidean distance between two interconnected nodes.
- l_e: Represents the physical length between nodes, i.e., joint poses.

The metric space defined by $(SE(3), d_R)$ serves as the global feature space with a specific distance function for node comparison. The local feature space is identified as an Euclidean space $(\mathbb{R}, d_{\mathbb{R}})$.

Utilizing the 'networkx' library, the skeletal graph can be efficiently represented. Essential operations include loading the skeletal structure, calculating link lengths, examining adjacency matrices, and visualizing the skeletal construct.

For an N-joint robot, the overall embodiment state can be distinctly defined by an element of the direct product space comprising concatenated joint poses.

This methodological framework equips researchers with a thorough toolkit for creating, visualizing, and analyzing skeletal structures for embodiment comparison. By translating these entities into graphs, various mathematical and computational techniques become available, heralding deeper insights and advanced operations.

3.3.2 Graph Matching as a Representation of Embodiment Correspondence

In this section, we delve into expressing the connection between two embodiment representations, denoted as G_1 and G_2, which have a predetermined number of nodes and edges, captured as $n = |V|$ and $m = |E|$. This connection or correspondence is conceptualized as an injective function $c : V_1 \to V_2$ that establishes a link between nodes across embodiment graphs based on specific criteria. We theorize that within imitation learning, this correspondence challenge is best framed as an approximate graph matching problem [29]. Given that there's rarely a direct bijective relationship between nodes (considering n_1 doesn't always equate to n_2), the strategy is designed to pinpoint correspondence with minimal distortion or to amplify graph similarities.

First, we introduce the notion of correspondence through a binary assignment matrix, represented as $X \in \{0, 1\}^{n_1 \times n_2}$. The overarching structure of the graph G is depicted through the node-edge incident matrix, captured as $G \in \{0, 1\}^{n \times m}$, with the condition that $G_{i,e} = G_{j,e} = 1$ if and only if the i-th and j-th nodes are bridged by the e-th edge.

Drawing inspiration from Lawler's Quadratic Assignment Problem, the correspondence conundrum between G_1 and G_2 is framed as:

$$\begin{aligned} &\text{vec}(\mathbf{X})^\intercal K \text{vec}(\mathbf{X}) \\ &\text{s.t. } \mathbf{X} \in \{0,1\}^{n_1 \times n_2},\ X\mathbf{1}_{n_2} = \mathbf{1}_{n_1},\ \mathbf{X}^\intercal \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2} \end{aligned} \tag{3.3}$$

Within this formulation, K serves as the second-order affinity matrix. The inherent constraints are designed to ensure a two-way one-to-one node connection across graphs. As detailed in [42], the decomposition of K is presented as:

$$\begin{aligned} \mathbf{K} &= (H_2 \otimes H_1)\text{diag}(\text{vec}(\mathbf{L}))(H_2 \otimes H_1)^\intercal, \\ \text{s.t. } \mathbf{H_1} &= [\mathbf{G_1}, \mathbf{I_{n_1}}] \in \{0,1\}^{n_1 \times (m_1+n_1)}, \\ \mathbf{H_2} &= [\mathbf{G_2}, \mathbf{I_{n_2}}] \in \{0,1\}^{n_2 \times (m_2+n_2)}, \\ \mathbf{L} &= \begin{bmatrix} \mathbf{K}^e & -\mathbf{K}^e\mathbf{G}_2^\intercal \\ -\mathbf{G}_1\mathbf{K}^e & \mathbf{G}_1\mathbf{K}^e\mathbf{G}_2^\intercal + \mathbf{K}^v \end{bmatrix} \in \mathbb{R}^{(m_1+n_1)\times(m_2+n_2)} \end{aligned} \tag{3.4}$$

Local graph-oriented heuristics in the aforementioned factorized expression are derived from both edge-affinity and node-affinity matrices, as illustrated:

$$\begin{aligned} \mathbf{K}^e_{i,j} &= \exp\left(-\frac{(l^1_e(i) - l^2_e(j))^2}{\sigma_e}\right),\ \sigma_e > 0, \\ \mathbf{K}^v_{k,l} &= \exp\left(-\frac{(l^1_v(k) - l^2_v(l))^2}{\sigma_v}\right),\ \sigma_v > 0, \end{aligned} \tag{3.5}$$

where hyperparameters are represented by σ_e and σ_v.

There exists an array of solution approaches to address the problem within the realm of graph-matching [29, 43]. But given that the approximate Graph Matching problems in this study involve relatively smaller graphs (less than 30 nodes as an example), our strategy harnesses the straightforward yet effective Spectral Matching technique, also recognized as the Power Iteration method. The methodology behind Spectral Matching streamlines the matching output into a vector unit, bypassing the binary constraints. Following this,

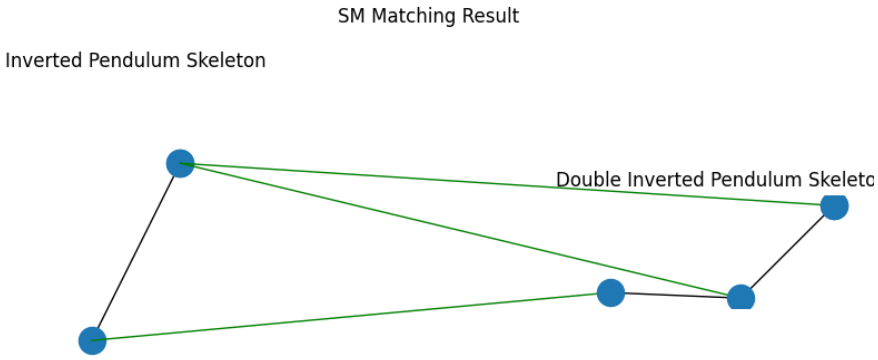


Figure 3.2: Correspondence match between inverted pendulum and double inverted pendulum

the relaxed solution draws from the power iteration rule: $x_{i+1} = Kx_i/\|Kx_i\|$. As a final step to restore the binary constraints, the solution derived from Spectral Matching is subjected to the Hungarian algorithm [44].

Example: Correspondence Matching in Kinematic Trees

To elucidate the concept of correspondence matching between kinematic trees, let's consider two different systems: an inverted pendulum and a double inverted pendulum. In this case, we have three key points of correspondence: the base joints, the pivot points, and the endpoints of the pendulums.

1. **Base Joints**: The distance between the base joints of the two systems is always zero since these points are fixed and serve as the origin for the other joints.
2. **Pivot Points**: These joints enable the pendulum's motion. In the case of the double-inverted pendulum, there will be an additional pivot compared to the simple inverted pendulum. Matching these pivots would involve computing their spatial distance in a specific configuration, contributing to the overall correspondence matching metric.
3. **End-points**: These are the points at the free ends of the pendulums. In the case of a double inverted pendulum, the second pendulum's end-point would be matched to an equivalent point derived from the simple inverted pendulum.

The visual representation of these correspondence matches can be illustrated with lines pointing from matched joint to matched joint, emphasizing the spatial relationship and how each joint in one structure corresponds to a joint in the other. The overall goal is to minimize the spatial discrepancy across these matched joints, which contributes to the reward function in optimizing the kinematic tree configuration.

Correspondence Distance Computation

The correspondence distance computation serves as an aggregate measure for the alignment between the two robotic embodiments. To incorporate the notion of correspondence between the joints, we make use of the optimal correspondence matrix $\mathcal{M}^*$, obtained through the approximate Graph Matching method, as described in the subsection 3.3.2.

The correspondence distance is a more nuanced quantity obtained as follows:

$$d_C = \text{tr}(\mathcal{M}^* \mathcal{D}^\intercal) \tag{3.6}$$

where $\mathcal{D}_{ij}$ represents the distance between joint i in the first model and joint j in the second model, and $\text{tr}(\cdot)$ is the trace of the resulting matrix. The trace essentially captures the sum of the diagonal elements, the aligned distances after correspondence.

The resulting correspondence distance serves as a critical metric in designing our correspondence reward function. It quantifies the deviation or alignment between the two robotic embodiments and plays a significant role in influencing the agent's actions and learning trajectory.

3.3.3 Utilization of PPO and TRPO Algorithms

In the beginning stages of this research, we explored the trust region by employing the TRPO and PPO algorithms to train the gym environments. The purpose was to analyze the effectiveness and nuances of each algorithm for the problem at hand.

Core Framework

Initially, an experiment setup was developed to test and evaluate the performance of the selected algorithm, whether it be PPO or TRPO. This setup included:

- Configuring the environment using OpenAI's Gym interface for the single inverted pendulum task.
- Specifying a Gaussian policy, which is parameterized by a neural network.
- Iteratively training the agent over specified epochs, each involving learning from the environment and evaluating the agent's performance.
- After training, the agent's performance was visualized over a few episodes.
- For reproducibility and future use, the learned weights of the agent's policy were saved.

Refinement and Modularization

To enhance the scalability of the training process across multiple environments and tasks, a more refined and modular framework was constructed. This framework used configuration-driven design thanks to the *hydra* utility. With this setup, training various environments became as simple as specifying the environment name and corresponding configuration files.

The core components of this refined framework are:

- A mechanism to construct PPO models based on whether the environment state was described using pixel data or traditional state vectors.
- Separate modules for policy and value function, each defined using neural networks. For environments with continuous actions, policies were formulated using a TanhNormal distribution, whereas discrete action spaces utilized the OneHotCategorical distribution.
- A loss computation module, which, in the context of PPO, employed a *clipping* mechanism to maintain the updated policy close to the original policy during optimization. This is a central feature of the PPO algorithm to ensure stable learning.

3.3.4 Design of Correspondence Reward Functions

In the domain of Inverse Reinforcement Learning, the design of the reward function holds paramount importance. It offers the agent crucial insights about the optimality of its actions within a specific state. For our study, which delves into correspondence matching in robotics, the reward function is indispensable. It not only serves as a guide but also ensures that the agent achieves the desired behavior: a close correspondence between different embodiments.

A challenge intrinsic to our study is the assessment of the proximity between the joints of two robot embodiments. With this challenge in mind, our reward function is tailored to reflect the distance between corresponding joints. Our primary goal was to encourage our Proximal Policy Optimization (PPO) agent to align the joint configurations of two diverse robotic embodiments closely. As such, our correspondence reward system was designed such that the closer the joint configurations between the two models, the higher the reward given to the agent, and conversely, more significant disparities resulted in lower rewards.

Embodiment Metric and Reward Function Construction

Consider $\mathcal{P}_1$ as the imitating embodiment (learner) with the configuration vector $\mathbf{p}$. This structure is characterized by its kinematics representation as $m_1^1(k) = K_{\mathbf{p}}(k)$. For every node k within $\mathcal{N}_1$, its Jacobian is given by $\mathcal{J}_k(\mathbf{p})$, derived from a known robotic model. We utilize the notation $\mathcal{P}_1(\mathbf{p})$ to emphasize how each configuration can alter the graph's properties via $K_{\mathbf{p}}(\cdot)$.

The mentor morphological structure, denoted as $\mathcal{P}_2$, can either be derived from a controlled robot or an unregulated human form using certain humanoid models. When the optimal correspondence matrix $\mathcal{M}^*$ is retrieved using the approximate Graph Matching method, the morphological metric is formulated as the aggregate of the paired global feature differences. Morphological metric can be explicitly described as the accumulated distance of matched joint positions across the forms:

$$\begin{aligned} &\delta_M(\mathcal{P}_1(\mathbf{p}), \mathcal{P}_2) = \text{tr}(\mathcal{M}^* \mathcal{B}^\intercal), \\ &\quad \text{where } \mathcal{B}_{k,l} = \delta_S(K_{\mathbf{p}}(k), m_1^2(l)) \, \forall k \in \mathcal{N}_1, l \in \mathcal{N}_2. \end{aligned} \tag{3.7}$$

The function $\delta_M(\mathcal{P}_1, \mathcal{P}_2)$ serves as a metric within the space of all morphological graph representations, symbolized by $\mathscr{P}$.

Having established how the morphological features of both the learner and mentor robots are quantified, we can now utilize this embodiment metric, δ_M, in formulating our reward function.

The correspondence reward function utilizes the tensor $\mathbf{X}_d$, which encapsulates the details of joint configurations and is obtained through a specialized correspondence method. The reward function is defined as:

$$r(\mathbf{q}_1, \mathbf{q}_2) = exp(-\delta_M(\mathcal{P}_1(\mathbf{q}_1), \mathcal{P}_2(\mathbf{q}_2))) \tag{3.8}$$

where $\mathbf{q}_1$ and $\mathbf{q}_2$ represent the joint configurations of the two models. The scalar $d(\mathbf{q}_1, \mathbf{q}_2, \mathbf{X}_d)$ signifies the total distance between the correspondences of these models, as elaborated in a preceding section. The design ensures that a smaller d yields a higher r, guiding the agent to find configurations that minimize the correspondence distance.

Integration with the PPO Agent

The correspondence reward function was encapsulated within a class. This class inherits from a superclass and primarily interfaces with the agent's observations, transforming them into correspondence rewards. The reward is then integrated with the data collection process and the testing environment. This ensures that during training and evaluation, the PPO agent's interactions are consistently guided by the designed correspondence reward mechanism.

To further illustrate, upon the agent's interaction with the environment, the observed joint configuration is passed through the reward function. The resultant reward value, grounded in the principles of correspondence, then influences the agent's learning, subtly nudging it towards achieving better alignment between the two robotic models.

In conclusion, the correspondence reward function serves as a bridge between the agent and the desired task of achieving optimal joint correspondences. It ensures that the PPO agent receives pertinent feedback regarding the quality of its actions, driving it towards achieving the intended goal of the study.

4 Experiments and Results

In this illuminating chapter, we delve into the practical facet of our research. With a foundational understanding of our methodologies, we transition to the empirical side, presenting the detailed experimental setups we employed, from the hardware configurations to the digital environments. We systematically analyze and present the outcomes, showcasing the performance metrics and trajectory analyses employed.

4.1 Experimental Setup

The experiments were conducted on the Lichtenberg High Performance Computer (HPC). This HPC system offers a robust infrastructure that caters to high computational demands. The Lichtenberg HPC comprises multiple login nodes, and these nodes are equipped with the "Cascade Lake" architecture, supporting AVX512, and boast 96 cores with a memory capacity of 768 GB RAM. Notably, some nodes are also further enhanced with NVidia Tesla T4 GPUs.

Furthermore, the HPC system is equipped with accelerator nodes that feature GPU accelerators from Nvidia, specifically the Volta 100 and Ampere 100 models. These GPUs are designed to significantly boost computational capabilities, especially beneficial for tasks that require parallel processing, such as deep learning models and simulations[45].

For experiment tracking and result visualization, we utilized Weights and Biases (wandb), which greatly facilitated the monitoring of real-time metrics and overall performance [46].

In the context of this study, leveraging MuJoCo's gym environments provided a robust platform for simulating and analyzing the dynamics of robotic systems, ensuring accurate and efficient results. MuJoCo, which stands for Multi-Joint Dynamics with Contact, is a physics engine tailored specifically for model-based control. MuJoCo represents multi-joint dynamics in generalized coordinates and computes them using recursive algorithms [47].

4.1.1 Description of Environments

In this study, various environments were used to assess the effectiveness of our correspondence-matching approach in reinforcement learning. These include the single-inverted pendulum, double-inverted pendulum, half-cheetah, and ant environments. Classical deep reinforcement learning techniques, specifically the Proximal Policy Optimization algorithm, were initially employed to train agents in each setting.

For the inverted pendulum and double inverted pendulum experiments, the focus was on achieving specific states, namely the swing-up state. After training an agent in the single pendulum environment, we used our correspondence matching method to transfer this learning to the double pendulum environment. The double pendulum agent was guided to reach the swing-up state of the single pendulum by minimizing the correspondence points' distance. Remarkably, the rewards achieved through this approach were better than those gained through conventional deep reinforcement learning. This improvement was also visually evident in the environment simulations.

In the case of the half-cheetah and ant environments, the objective is to teach locomotion. We plan to train the half-cheetah agent using classical PPO methods to learn how to walk efficiently. The collected trajectories, which will consist of states and actions, are intended to guide the ant agent through our correspondence matching method. While we have not yet completed this phase, our hope is that the ant agent will be able to leverage the 'lessons' learned by the half-cheetah to improve its own walking capabilities.

4.1.2 Performance Metrics

To rigorously evaluate the effectiveness and efficiency of our correspondence-matching approach, we rely on a combination of quantitative metrics and qualitative insights. We employ wandb for experiment tracking, and our analysis covers a range of metrics:

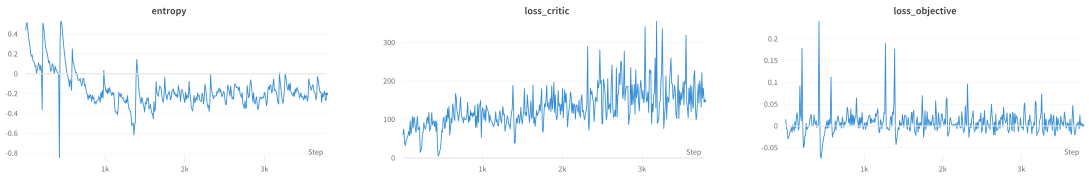
1. **Reward Metrics:** Our primary metric is the reward, used both for training and testing which depicted in the below Figure 4.1.

(a) Training reward (b) Testing reward

Figure 4.1: Example metrics obtained from the combined reward solution.

- *Environment Reward (Vanilla RL):* The basic reward given by the environment, serving as a baseline for performance comparison[48].
- *Correspondence Reward:* We also make use of a correspondence reward, designed based on our correspondence matching methodology.
- *Combined Reward:* A blend of the environment and correspondence reward, which yielded the best performance in our experiments.

2. **Success Rate over Seeds:** To rigorously evaluate the robustness and reliability of our approach, we employed various seed values in our experiments. While the seeds themselves are not critical for the learning process, using different seeds allows for a more comprehensive evaluation of the model's performance across different initial conditions.
3. **Entropy and Loss Metrics:** While not direct performance metrics, 'entropy', 'critic loss', 'entropy loss', and 'objective loss' provide valuable diagnostic information. These metrics help understand the agent's behavior, such as its exploration-exploitation balance and the convergence of the critic network. The relevant metrics are depicted in Figure 4.2, given below.



(a) Entropy (b) Critic Loss (c) Objective Loss

Figure 4.2: Additional statistics of the combined reward solution.

In our experiments on the pendulum environment, the best performance was achieved using the combined reward scheme, followed by the correspondence reward and then the classical RL reward. This suggests that augmenting the traditional environment reward with our custom correspondence-based reward can indeed enhance the learning process, providing both a robust and effective approach for robotic task imitation.

These metrics are designed to provide a multi-faceted evaluation, capturing both the quality and efficiency of the imitation learning process. Combined, they give a robust understanding of the system's performance, guiding future improvements and refinements.

4.1.3 Trajectory Analysis and Correspondence Matching

In this section, we aim to provide insights into the performance and efficacy of our approach through two key lenses: correspondence matching and trajectory analysis. First, we will showcase results from various correspondence-matching exercises that were carried out to validate the universality of our method. This will include extending our matches beyond the inverted pendulum and double-inverted pendulum examples discussed in the previous chapter. Following that, we will delve into the trajectory analysis to examine the behavioral patterns exhibited by our models during the learning process.

Correspondence Matching: After detailing correspondence matches between the inverted pendulum and the double inverted pendulum, here we broaden the scope to incorporate other intriguing matches. One such match places the inverted pendulum against the cart pole system, as depicted in Figure 4.3.

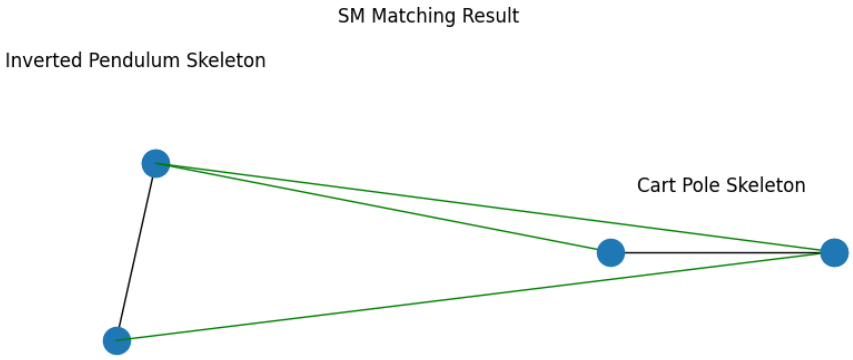


Figure 4.3: Correspondence match between inverted pendulum and cart pole

An additional example considers the matching between a human skeleton and a dolphin skeleton, illustrated in Figure 4.4.

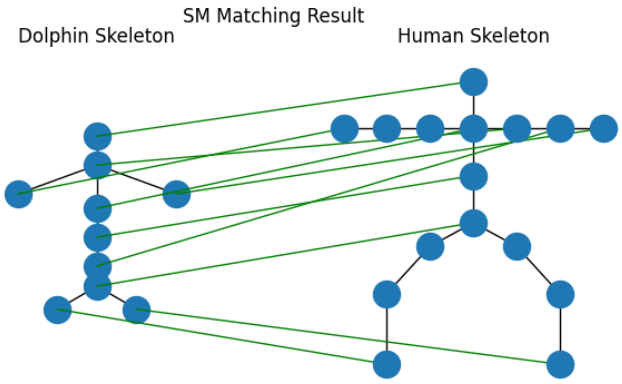


Figure 4.4: Correspondence match between human skeleton and dolphin skeleton

It is worth mentioning that our automated code system can carry out these calculations for any kind of embodiment, provided the URDF files are accessible.

Trajectory Analysis: An essential component of our methodology involves the collection and examination of trajectories. These consist of state-action pairs along with associated

weights. Once training is complete, the actor's state dictionary, which essentially represents the trained policy, is saved for future use. This allows us to reload the state dictionary as needed for analysis.

Through this mechanism, we identify successful state-action pairs that serve as effective guiding signals for the reinforcement learning agents during their learning phase. These pairs are especially crucial for our overarching goal of matching corresponding state-action pairs across different environments. For instance, successful walking state-action pairs observed in the expert agent operating in the 'half cheetah' environment can provide invaluable guidance to a learner agent in the 'ant' environment.

However, it's worth mentioning that this aspect of trajectory analysis was only partially implemented in our research due to time constraints.

4.1.4 Configurations and Parameters

In our experimental setup, we utilized Hydra for configuring the pipeline. Hydra facilitates the organization and management of complex application configurations, providing a structured way to declare and override parameters via YAML files.

Base Configuration File

```
defaults:
  - experiment: ppo_double_pendulum
  - _self_

# wandb configs
project: c4il
entity: ias-tudarmstadt
wandb_path: /home/alper/c4il/
save_path: /home/alper/c4il/data/weights/weights.pt
```

Experiment Configuration File

```
defaults:
  - task: double_pendulum
```

```
# collector
collector:
  frames_per_batch: 64
  total_frames: 1_600
  collector_device: cuda

# logger
logger:
  backend: wandb
  exp_name: double_pendulum
  log_interval: 64
  record_video: True
  wandb_kwargs:
    entity: ias-tudarmstadt
    project: c4il

# Optim
optim:
  device: cuda
  lr: 3e-4
  weight_decay: 1e-4
  lr_scheduler: True

# loss
loss:
  gamma: 0.99
  mini_batch_size: 64
  ppo_epochs: 10
  gae_lambda: 0.95
  clip_epsilon: 0.20
  critic_coef: 0.5
  entropy_coef: 0.0
  loss_critic_type: l2
  normalize_advantage: True

# network architecture
nets:
```

```
  policy_num_cells: [128]
  value_num_cells: [128]
```

Task Configuration File

```
env_name: InvertedDoublePendulum-v4
env_task: ""
env_library: gym
frame_skip: 1
num_envs: 1
noop: 1
reward_scaling: 1.0
from_pixels: True
use_pixel_state: False
pixels_only: False
n_samples_stats: 3
device: cuda
seed: 0
```

The hyperparameters for alternative environments, such as the half cheetah, were configured closely aligned with those of the primary experiments.

4.2 Main Findings

In this section, we present a thorough analysis of the correspondence-matching results compared to classical Reinforcement Learning approaches. We then delve into the successes achieved and challenges faced during this study.

4.2.1 Results and Analysis of Correspondence Matching

We conducted experiments to compare the efficacy of our correspondence-matching method against classical Reinforcement Learning (RL). The vanilla RL approach was run with six different seed values to ensure a comprehensive evaluation, and the mean results are depicted in Figure 4.5.

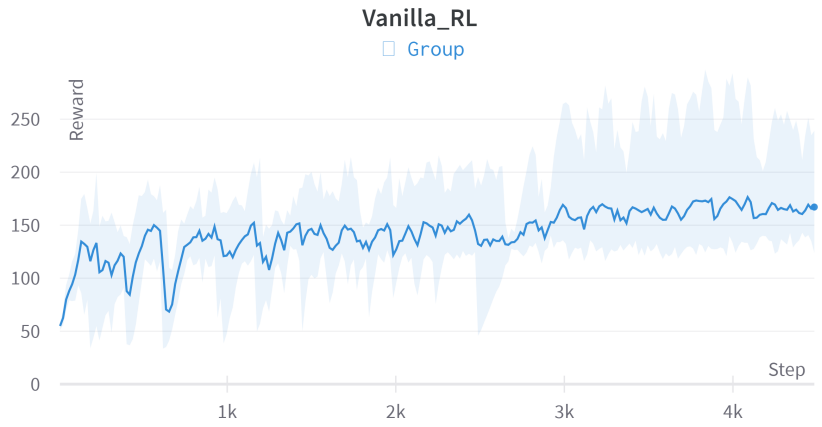


Figure 4.5: Mean training rewards of Vanilla RL

The rescaled mean curve version of the Vanilla RL approach is illustrated in Figure 4.6.

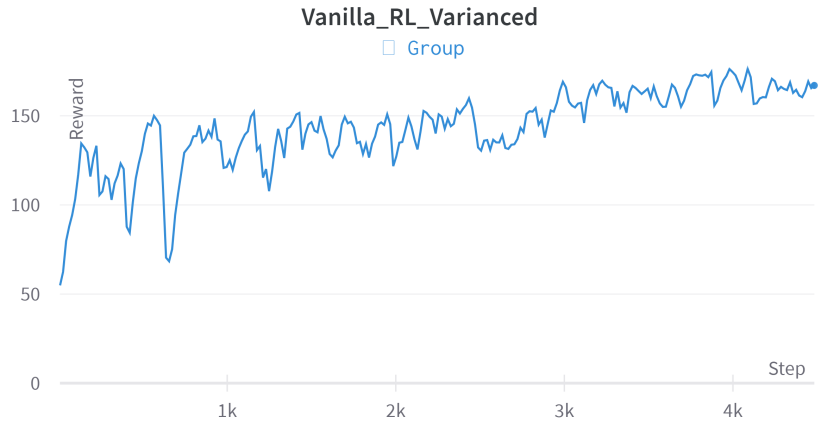


Figure 4.6: Rescaled mean training rewards curve obtained using Vanilla RL

Following this, we employed a correspondence reward function that leverages our correspondence-matching technique and applied this function to a Proximal Policy Optimization (PPO) agent. The results are presented in Figure 4.7.

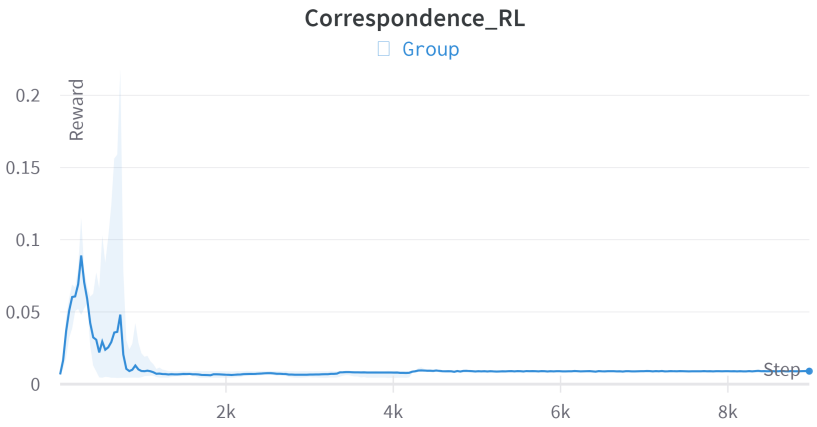


Figure 4.7: Mean training rewards using correspondence-matching integrated PPO agent

The rescaled mean curve version of the results obtained with the correspondence reward function is shown below in Figure 4.8.

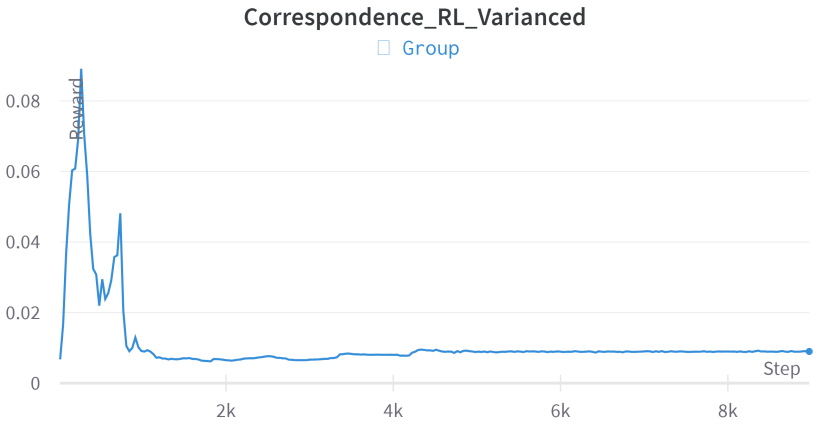


Figure 4.8: Rescaled mean training rewards curve using correspondence-matching integrated PPO agent

Finally, we also experimented with a combined reward system that merges the correspondence reward from the correspondence matching and the traditional rewards given by the environment, as shown below in Figure 4.9.

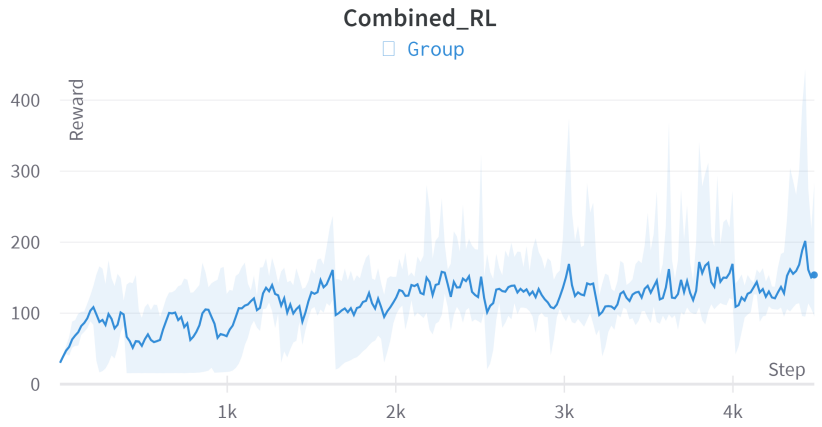


Figure 4.9: Mean training rewards using both correspondence and environmental rewards

The rescaled mean curve of the combined reward graph is also calculated and presented in Figure 4.10 below.

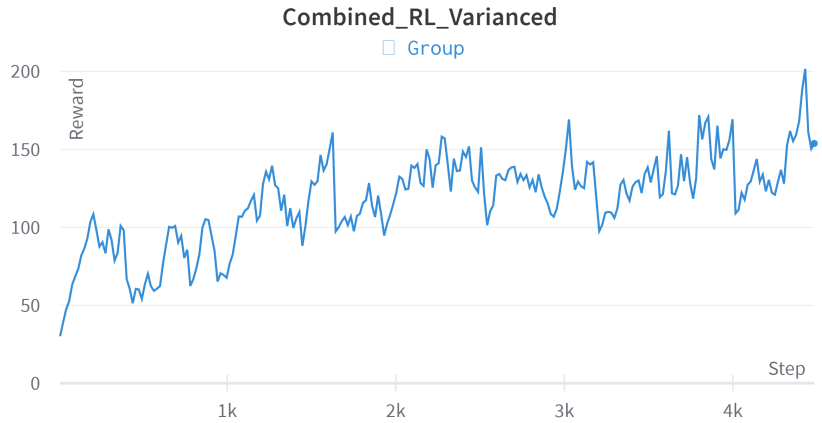


Figure 4.10: Rescaled mean training rewards curve using both correspondence and environmental rewards

Analysis: Upon evaluating the learning curves and the observed behavior of the systems, we found that the combined reward system delivered the best performance. With this approach, the system was able to stabilize the pendulum indefinitely. When we applied only

the correspondence reward based on correspondence matching, the pendulum stabilized well for a period, but then the reward levels decreased dramatically. This can be attributed to the diminishing returns on the correspondence reward once the distances are minimized. In this phase, however, the pendulum stabilized indefinitely and, in most cases, for over 20 seconds. Conversely, the classical RL approach managed to stabilize the pendulum for a maximum of approximately 8 seconds at peak performance.

4.2.2 Successes and Challenges

Our work has successfully developed a correspondence-matching method, which, when combined with a Proximal Policy Optimization (PPO) agent, shows promising results compared to standard reward designs in Reinforcement Learning. The combined reward system showed particular promise, stabilizing the pendulum indefinitely in tests. Despite these achievements, challenges arose, notably due to time constraints that impacted the full implementation of state-action pair matching for complex environments like the half cheetah-ant case. The long-term effectiveness of correspondence rewards from correspondence matching also presented a dilemma, necessitating additional methods to maintain stability.

5 Discussion

Here, we'll dissect our results, and contemplate the broader implications, ensuring the reader can discern the full scope of our research endeavors.

5.1 Interpretation of Results

The primary focus of this thesis is to leverage correspondence metrics into the reward function design for reinforcement learning, thereby introducing an element of imitation learning into the learning process. Specifically, we used a correspondence metric along with a policy from a different embodiment to train another embodiment using Reinforcement Learning.

Our contribution lies in integrating bi-directional correspondence matrices into the RL framework as an auxiliary reward. This approach was employed in two distinct manners: solely using the correspondence reward and combining it with the environment's nominal reward. The experimental results indicate the viability of incorporating correspondence metrics into the reward function. This hybrid approach of using RL with imitation learning elements demonstrates promising performance.

In summary, the outcomes from our empirical study validate the utility of integrating imitation learning concepts into RL-based policy training, either as an exclusive reward function or as a supplementary one combined with traditional rewards. This strategy could potentially address the challenges in training robots with different embodiments for complex tasks.

5.2 Significance of Findings

The implications of this research are significant for robotics and reinforcement learning. By successfully automating the process of establishing correspondence between dissimilar robotic structures, the study enhances the adaptability and versatility of robotic systems. This advancement allows these systems to learn from a diverse set of expert models, thereby expanding the scope and effectiveness of imitation learning techniques.

Moreover, the generic and flexible nature of the metric-based imitation method introduced in this research suggests its potential applicability across a wide range of robotic systems and scenarios. While the current study focused primarily on rigid kinematic chains, the foundational principles could be extended to other topologies, including tree-like structures like humanoids or even free topologies such as swarms of flying objects. However, it's worth noting the limitations of the current approach, particularly the absence of a probabilistic description and its current restriction to rigid kinematic chains. Future research could delve deeper into these areas, further refining the method and expanding its scope.

5.3 Possible Limitations

In the ever-evolving domain of robotics, our approach presents to bridge the data gap, especially concerning intricate embodiments. Historically, collecting data for complex robotic forms was laborious and resource-intensive. Our methodology sidesteps this by reducing the need for data collection, relying on a strategy of corresponding matching. It facilitates the transfer of learned behaviors from one embodiment to another without the arduous task of collecting new datasets for every complex robotic environment. However, this methodology is not without its limitations.

One significant drawback to our method lies in the challenge posed by robot embodiments that differ considerably in structure and functionality. For instance, matching data and transferring learnings between a bipedal (two-legged) robot and a quadruped (four-legged) robot might prove problematic. The physical differences between these two types of robots are vast, and the underlying dynamics governing their movements and interactions with the environment differ significantly. Thus, our corresponding matching method might falter in scenarios where two embodiments are too different or when the nuances of one robot's functioning cannot be adequately captured or translated for the other.

5.4 Future Work

Building on our findings and the challenges faced, an intriguing avenue for future research is addressing the dichotomy between robots with similar structural skeletons but divergent behaviors. The pressing question becomes: How can one effectively train and formulate policies that cater to such intricacies?

One unexplored aspect is the effect of time-dependent behaviors in simple embodiments. How would correspondence mapping adapt when the behavior of the more straightforward entity changes over time? Addressing this issue could provide further insights into the adaptability and robustness of our method.

Another interesting avenue for research involves investigating the limitations when embodiments are too different. While the current study has made strides in matching correspondence between different embodiments, a deeper understanding is needed to explore the constraints and potential workarounds when the embodiments diverge significantly.

There's a vast potential in exploring hybrid training techniques that combine traditional data collection with advanced transfer learning algorithms. By doing so, one might unlock the capability to quickly adapt policies crafted for one robot to another with minor tweaks in its structure or behavior. Furthermore, diving deeper into meta-learning could provide solutions that allow robots to learn how to learn, thereby becoming more adaptive and less reliant on large datasets. The end goal would be to create a universal reinforcement learning paradigm for robotics, where diverse robots could swiftly benefit from the experiences and learnings of their counterparts. For instance, leveraging policies from both a single pendulum and a double pendulum could inform a more robust approach to stabilizing a cart-pole system.

Lastly, a question remains: When is behavior "close" when we have similar but still different skeletons or graphs? Developing and defining metrics for behavioral divergence could be a key part of future work in this area.

Overall, moving forward, further research will be essential to extend the current work, addressing existing limitations and investigating new approaches for more comprehensive solutions.

6 Conclusion

This thesis explores the intricacies of the correspondence problem within the realm of robotics and reinforcement learning. Through rigorous research and experimentation, we have delved deep into the challenges and potential solutions associated with this problem, particularly in the context of imitation learning. This problem arises when the learner and expert agents possess different physical properties, making it challenging for the learner to effectively imitate the expert.

Our research began with a foundational understanding of reinforcement learning, emphasizing its significance in sequential decision-making. The exploration of Markov Decision Processes (MDPs) further solidified the theoretical underpinnings, providing a robust framework for modeling sequential decision-making scenarios. The foundational concepts of robotics, reinforcement learning, and inverse reinforcement learning were explored in depth, providing the necessary theoretical backdrop for the research. The methodology adopted was rigorous, ensuring that the investigation was both robust and replicable. Through a series of experiments, the effectiveness of the proposed approach was demonstrated, highlighting its potential to enhance the efficiency and adaptability of robotic and artificial agents.

The core of our investigation revolved around the correspondence problem, where we introduced a novel approach to address the challenges of imitation learning across dissimilar embodiments. By transforming Unified Robot Description Format (URDF) files into structured graphs, we were able to encapsulate the essence of each robot's embodiment. This transformation helped for our correspondence-matching phase, where features were extracted and a graph-matching algorithm was applied to find the best correspondence between the graphs.

Our primary contribution lies in the development and validation of a novel correspondence-matching method. Unlike traditional approaches, our method leverages a bi-directional correspondence matrix, allowing for a more dynamic and automated establishment of correspondences between links. Also, the introduction of a distance measure, which

utilizes the correspondence matrix, has provided a robust metric to gauge the similarity between different embodiments.

Furthermore, the experiments and results section of this thesis has provided valuable insights into the practical implications of our approach. The performance metrics used have given us the effectiveness of our methodology, guiding future refinements and iterations. Our experiments, conducted on the IAS Cluster, provided empirical evidence of the efficacy of our approach.

While this research shows potential in specific applications, it has its limitations that warrant further investigation. One area of concern is the scope of its applicability across diverse robotic embodiments and diverging behaviors. These complexities require additional study. Furthermore, the issue of time-dependent policies, as we initially considered for the cheetah and ant case, poses an additional layer of complexity that has not yet been addressed. Lastly, challenges arising from the curse of dimensionality, a common obstacle in machine learning problems, also require further examination in future work.

In conclusion, this thesis has made strides in addressing the correspondence problem in imitation learning. The structured-graph correspondence approach has added a valuable tool to the repertoire of solutions available to tackle this challenge. The contributions of this work have the potential to inform future research in the fields of robotics and artificial intelligence.

Bibliography

[1] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[3] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[4] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 1811.06711, 2018.

[5] P. Englert, A. Paraschos, M. P. Deisenroth, and J. Peters, "Probabilistic model-based imitation learning," *Adaptive Behavior*, vol. 21, no. 5, 2013.

[6] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *arXiv preprint arXiv:1011.0686*, 2011.

[7] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, (Stockholm, Sweden), July 2018. arXiv:1805.01954v2 [cs.AI] 11 May 2018.

[8] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the Twenty-first International Conference on Machine Learning*, Stanford University, Stanford, CA 94305, USA, 2004.

[9] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," *arXiv preprint arXiv:1603.00448*, 2016.

[10] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[11] G. Niemeyer, C. Preusche, S. Stramigioli, and D. Lee, *Telerobotics*, pp. 1085–1108. Springer, 2016.

[12] A. Billard, S. Calinon, and R. Dillmann, *Learning from humans*, pp. 1995–2014. Springer, 01 2016.

[13] J. Vertut and P. Coiffet, *Teleoperation and Robotics: Applications and Technology*. NSRDS Bibliographic Series, Dordrecht: Springer Dordrecht, 1 ed., 2013. Published: 13 March 2013.

[14] N. E. Sian, K. Yokoi, S. Kajita, F. Kanehiro, and K. Tanie, "Whole body teleoperation of a humanoid robot using simple joysticks," *Journal of the Robotics Society of Japan*, vol. 22, no. 4, 2004.

[15] J. Park and O. Khatib, *Robust Haptic Teleoperation of a Mobile Manipulation Platform*. January 2006.

[16] L. Fritsche, F. Unverzagt, J. Peters, and R. Calandra, "First-person tele-operation of a humanoid robot," *Unknown Journal*, 2015.

[17] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[18] S. B. Niku, *Introduction to Robotics: Analysis, Control, Applications*. John Wiley & Sons, Inc., 2001.

[19] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, 2016.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2020.

[21] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, *An Introduction to Deep Reinforcement Learning*. Foundations and Trends in Machine Learning, 2018.

[22] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust reinforcement learning: A review of foundations and recent advances," *Machine Learning and Knowledge Extraction*, vol. 4, 2022.

[23] C. Szepesvári, *Algorithms for Reinforcement Learning*. University of Alberta, 2010.

[24] J. Peters and J. A. Bagnell, "Policy gradient methods," *Carnegie Mellon University. Journal contribution*, 2018.

[25] L. Weng, "Policy gradient algorithms," *lilianweng.github.io*, 2018.

[26] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," *arXiv preprint*, vol. 1502.05477, 2015. Accessed from arXiv.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, vol. 1707.06347, 2017.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[29] J. Yan, X.-C. Yin, W. Lin, C. Deng, H. Zha, and X. Yang, "A short survey of recent advances in graph matching," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pp. 167–174, 2016.

[30] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmpflow: A computational graph for automatic motion policy generation," in *International Workshop on the Algorithmic Foundations of Robotics*, pp. 441–457, Springer, 2018.

[31] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377–388, 1996.

[32] M. Zaslavskiy, F. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2227–2242, 2009.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024–8035, 2019.

[34] A. Bou, M. Bettini, S. Dittert, V. Kumar, S. Sodhani, X. Yang, G. De Fabritiis, and V. Moens, "Torchrl: A data-driven decision-making library for pytorch," *arXiv preprint arXiv:2306.00577*, 2023.

[35] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Mushroomrl: Simplifying reinforcement learning research," *Journal of Machine Learning Research*, vol. 22, no. 131, pp. 1–5, 2021.

[36] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara, "Transparency by design: Closing the gap between performance and interpretability in visual reasoning," *arXiv preprint arXiv:1809.06061*, 2018.

[37] D. Tola and P. Corke, "Understanding urdf: A survey based on user experience," *arXiv preprint arXiv:2302.13442*, 2023.

[38] D. Rosen, J. Smith, and J. Doe, "Understanding urdf," *Journal of Robotic Systems*, 2023.

[39] A. Parvez Mohammed and M. Valdenegro-Toro, "Can reinforcement learning for continuous control generalize across physics engines?," *arXiv preprint arXiv:2010.14444*, 2020.

[40] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018.

[41] F. C. Park and B. Ravani, "Smooth invariant interpolation of rotations," *ACM Transactions on Graphics (TOG)*, vol. 16, no. 3, pp. 277–295, 1997.

[42] F. Zhou and F. De la Torre, "Factorized graph matching," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 127–134, IEEE, 2012.

[43] J. Yan, S. Yang, and E. R. Hancock, "Learning for graph matching and related combinatorial optimization problems," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4988–4996, International Joint Conferences on Artificial Intelligence Organization, 2020.

[44] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[45] B. Juhl, "Introduction to the lichtenberg high performance computer." `https://www.hhlr.tu-darmstadt.de/`, 2022.

[46] L. Biewald, "Experiment tracking with weights and biases," 2020. Software available from wandb.com.

[47] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[48] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.