TUM School of Computation, Information and Technology
Technical University of Munich

Master's Thesis in Robotics, Cognition, Intelligence

Active Assistance for Pathological Gait using Inverse Reinforcement Learning

Aktive Assistenz bei pathologischen Gangstörungen mittels Inverse Reinforcement Learning

Supervisor Prof. Jan Peters Ph.D.
Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Michael Drolet, M.Sc.
Firas Al-Hafez, M.Sc.
Sebastian Hirt, M.Sc.
Dr. Oleg Arenz
Dr. Zhenshan Bing

Author Zongwei Zhang

Date September 16, 2025 in Munich

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Munich, September 16, 2025

(Zongwei Zhang)

Abstract

Although many studies on exoskeleton control have been conducted, several challenges remain. For one, some approaches focus solely on the exoskeleton itself and neglect the interaction and cooperation between humans and machines, resulting in the human having to adapt their motion to the exoskeleton. Moreover, controls are often generated using traditional techniques that don't leverage user data. Although these methods can yield satisfactory results, one of the disadvantages is that they are often limited to providing support passively through predefined parameters without individualization, thus leading to poor adaptability across scenarios and latency in the motion. These disadvantages can reduce the effectiveness of the wearable or even cause injury during gait correction in medical applications. This thesis presents a control strategy for the exoskeleton that optimizes the generation of healthy gait by correcting pathological gait based on the user's dynamic properties. The proposed solution employs inverse reinforcement learning to eliminate the need for manual reward engineering. Experiments are conducted in different environments to evaluate the feasibility of the approach, followed by training a unified policy capable of handling various impairments, including zero-shot cases. The final results demonstrate the potential of inverse reinforcement learning based methods for effective and adaptive gait rehabilitation.

Kurzfassung

Obwohl umfangreiche Forschung zur Regelung von Exoskeletten betrieben wurde, mehrere Herausforderungen bestehen weiterhin. Zuerst, dass viele Ansätze sich ausschließlich auf das Exoskelett alleine konzentrieren und die Interaktion sowie Kooperation zwischen dem Menschen und dem Gerät vernachlässigen. Infolgedessen muss sich der Nutzer häufig an die Bewegungen des Exoskeletts anpassen, anstatt rückkehrend. Darüber hinaus basieren Regelungsstrategien oft auf traditionellen Techniken, die keine nutzerspezifischen Daten einbeziehen. Obwohl solche Methoden zufriedenstellende Ergebnisse liefern können, bieten sie in der Regel nur passive Unterstützung durch vordefinierte Parameter ohne Individualisierung. Dieser Mangel an Anpassungsfähigkeit kann zu Verzögerungen in der Bewegungsreaktion führen und sogar das Risiko von Verletzungen während der Gangkorrektur in medizinischen Anwendungen erhöhen. Diese Arbeit schlägt eine Regelungsstrategie für Exoskelette vor, die die Erzeugung eines gesunden Gangs verbessert, indem sie pathologische Gangmuster basierend auf den dynamischen Eigenschaften des Nutzers korrigiert. Die vorgeschlagene Lösung nutzt Inverse Reinforcement Learning, um den Aufwand für die manuelle Definition von Belohnungsfunktionen zu vermeiden. Es werden Experimente in verschiedenen Umgebungen durchgeführt, um die Machbarkeit des Ansatzes zu evaluieren. Die abschließenden Ergebnisse zeigen das Potenzial von Inverse Reinforcement Learning basierten Methoden für eine effektive und adaptive Gangrehabilitation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Exoskeletons represent a compelling research direction in robotics due to their close interaction between humans and machines. Applications of exoskeletons can be found in various domains such as strength augmentation, motion assistance and gait correction for medical purposes [MFS23; RLF21; Lee+23]. With exoskeletons in different formats, researches have been conducted across multiple disciplines such as mechanics, electronics and computer science [Zho+19; San+19; Naz+23; BHF23]. These studies not only advance the development of exoskeleton technologies but also stimulate progress in related fields. One such field is sports and motion science, as exoskeletons are ultimately worn by humans. This necessitates a deeper understanding of how humans support their own movement in everyday life as well as how they interact with complex environments to different extent [Li+22; Fir+25].

To facilitate the effective use of exoskeletons, the control strategies should be capable of adapting to human motion patterns. This raises demands on exoskeleton control, especially given that humans are inherently able to accomplish tasks under diverse conditions and environments. Although many studies have explored control methods for exoskeletons, several challenges remain. For one, some approaches focus primarily on the exoskeleton itself and neglect the interaction/cooperation between humans and machines. As a result, users are often required to adapt their movements to the exoskeleton, rather than the other way around [Luo+21]. Additionally, controls are often generated using traditional techniques that don't leverage user data. Although such approaches may yield acceptable performance, they are often limited to providing support passively through predefined parameters without individualization, thus leading to poor adaptability across scenarios and latency in the motion [Che+22; Xue+19]. Another drawback is that some control models are based on simulations that fail to accurately represent human biomechanics. In theory, the modeled systems should be capable of executing motions without external support, just as humans can often perform movements independently of assistive devices. However, these models often ignore the fact that human motion is not driven by joint mounted motors, but by complex neuromuscular dynamics. As such, the individual's ego motion should also be taken into account. These disadvantages can reduce the effectiveness of the wearable or even cause injury during gait correction in medical applications or even lead to some unreal conclusions [Meh+23].

In recent years, reinforcement learning (RL) techniques have been developed and applied across various domains. Given that the core principle of RL is to take actions based on real-time observations, it presents a promising approach for exoskeleton control. Among the different RL methods utilized in control applications, one of the most critical factors for successfully training a policy is the design of a reward function tailored to the specific problem. However, reward engineering remains one of the most challenging and time consuming aspects of RL. Designing an effective reward function is rarely straightforward and often requires extensive domain knowledge and iterative refinement.

Inverse reinforcement learning (IRL), by contrast, tries to eliminate the need for manual

reward engineering. Instead of explicitly designing a reward function, IRL aims to infer the reward function from observed expert behavior. In doing so, the task specific problem of reward design is transferred to a more general learning problem, which is how to enable the agent to learn the underlying reward structure autonomously. Solving this problem has the potential to improve efficiency, as the learned reward models can be adapted and transferred across multiple scenarios.

In light of these observations, this thesis proposes an IRL control strategy for exoskeletons aiming at optimizing healthy gait by correcting pathological gait based on the user's dynamic properties. To evaluate the effectiveness of this approach, a musculoskeletal human model is first modified to simulate various medical conditions. Then IRL control method is adapted and applied to each condition independently to demonstrate its feasibility in gait correction. Following this, the study explores the development of one unified policy capable of addressing multiple gait disorders simultaneously. Several zero-shot cases are presented to further validate the robustness and adaptability of the proposed method as well.

The remainder of this thesis is organized as follows. Chapter 2 provides a brief review of related work, analyzing previous approaches and techniques for exoskeleton control including RL and IRL. Chapter 3 introduces the key concepts and mathematical foundations underlying the methods used in this thesis. Chapter 4 details the methodology for gait rehabilitation, while Chapter 5 presents the experimental setup and results. Finally, Chapter 6 offers a conclusion and outlines potential directions for future research.

Chapter 2

Related work

In this chapter, previous work on exoskeleton control is first reviewed in section 2.1 to provide an overview of the current state of control strategies in the field. To gain a deeper understanding of how reinforcement learning techniques can be applied to control models, we review relevant literature and studies on inverse reinforcement learning in section 2.2.

2.1 Exoskeleton control

Since their invention, exoskeletons have been designed to provide external support or assistance to humans at different stages of mobility or rehabilitation. One of the most common applications of exoskeletons is gait rehabilitation. Zhu et al. investigated the effects of exoskeleton assistance on human gait and demonstrated that such support can improve the muscle coordination from the sense of muscle synergy [Zhu+21]. Similarly, De Luca et al. reported that exoskeleton devices could affect walking patterns, which further supports their feasibility for gait rehabilitation [De +19]. In terms of control strategies, Qian et al. simulated a post-stroke case including gait asymmetry. They developed a system that provides support exclusively at the hip joints using a series elastic actuator (SEA) as the driving mechanism. Their method employed adaptive oscillators to detect gait asymmetry, followed by PI controllers to generate the control signals [QYF22]. Likewise, Aguirre-Ollinger et al. adopted a similar approach using SEAs and adaptive frequency oscillators to extract phase information. However, they proposed a different control strategy involving disturbance rejection and formulated the control pipeline using the Riccati equation [AY20]. Beyond purely control based on gait history, Unluhisarcikli et al. developed a lower extremity exoskeleton using impedance control, which accounts for motion and contact forces simultaneously to produce more accurate control signals [Unl+11]. There are also structural innovations in exoskeleton design. For instance, Wang et al. introduced additional degrees of freedom in the frontal plane. These were controlled using a PID controller together with the sagittal plane actuators, thereby improving flexibility [Wan+13]. To address variability in patient data, Zhang et al. trained a XGBoost model to generate usable reference trajectories. These were then paired with impedance control to ensure stability and safety of the output signals. Their method also integrated neural networks to further enhance data utilization and control performance [Zha+22].

2.2 Control using reinforcement learning

One of the advantages of reinforcement learning (RL) is its ability to interact dynamically with the environment. Thus, it is a promising approach for training policies that output actions based on real-time observations. Consequently, RL has been applied to robotic control systems. Peng et al. developed a physics-based motion animation framework that combines prior expert data via RL. Their work showed that a single RL agent could generate appropriate actions across different environments to enable the model to follow expert behavior [Pen+18]. Shan et al. proposed a method for generating locomotion commands for quadruped robots. They extracted critical information from weak sensor inputs and detected the environments via Fourier analysis. They also designed a multifunctional reward to accommodate diverse scenarios [Sha+24]. Zhang et al. introduced an RL approach for adaptive motion control on complex terrains. They used a time-cluster mechanism to store sensor data and analyze it afterwards. Their reward function included an energy term to promote relatively efficient movement as well [Zha+24]. To incorporate biological insights into RL, several studies have been conducted on musculoskeletal models. For instance, Schumacher et al. proposed an algorithm to improve the policy exploration efficiency in RL, facilitating fast convergence in overactuated environments with a high-dimensional action space. Their approach integrated differential extrinsic plasticity (DEP) into RL and demonstrated successful results in various muscle-driven simulations such as humanreacher, ostrich run, and human run [Sch+23].

To further simulate the functionality of exoskeletons based on a musculoskeletal model, Rose et al. proposed an approach of deep deterministic policy gradient (DDPG) that is able to adapt actuator control torques and accounts for post-stroke individuals with different desired gait patterns. Their reward function incorporated terms such as angle errors and boundary penalties to assist the learning [RBN22]. Yuan et al. combined RL with dynamic movement primitives to generate control signals for a walking exoskeleton robot, which effectively integrates rhythmic motion generation with adaptive control [Yua+19]. Addressing exoskeleton complexity, Luo et al. developed a model featuring degrees of freedom in both the sagittal and frontal planes. They introduced the center of pressure as part of the control algorithm to enhance balance maintenance. The control robustness was also improved by exposing the model to different perturbations during RL training [Luo+21]. Building on this work, the team decoupled exoskeleton dynamics from human motion and trained three parallel networks to simulate the behaviors of humans equipped with exoskeletons. Their observation history was also incorporated, and the multiple action steps are predicted for the future, further improving control accuracy and robustness [Luo+23].

As inverse reinforcement learning (IRL) eliminates the time-consuming effort of manual reward engineering, researchers employing IRL have also gained traction. For instance, Bing et al. developed a biomechanical snake-like robot and implemented an RL control algorithm to replicate serpentine motion. Their work further demonstrated the feasibility of using adversarial inverse reinforcement learning (AIRL) to autonomously learn control policies [Bin+20]. Similarly, Geiss et al. investigated the influence of various techniques on policy exploration. Their methods were validated on a musculoskeletal model, confirming the availability of IRL for training control strategies in muscle driven biomechanical systems [Gei+24].

Chapter 3

Preliminary

This chapter introduces the concepts and techniques employed in the following chapters. The concept of the musculoskeletal model is introduced first in section 3.1, including the muscle-tendon unit and the application of forward kinematics to the entire system. This is followed by a review of the fundamental principles of reinforcement learning and inverse reinforcement learning in sections 3.2 and 3.3. Section 3.4 introduces the concept of the long short-term memory and section 3.5 gives an overview of the dynamic time warping algorithm.

3.1 Musculoskeletal model

Numerous robotic models have been developed both in simulation and in the real world [FB21]. Most of these models are actuated by electric motors, meaning that the joints not only define the movement constraints of the system but also serve as the primary source of actuation. However, this approach does not accurately represent the true functional mechanisms observed in biological organisms. In living creatures, joints provide structural constraints, while actuation is performed by muscles rather than by the joints themselves [An02; PA10].

To more closely replicate human motor function, a specialized model known as the musculoskeletal model has been introduced [WFD23; SLK21; GMA19]. This type of model consists of two key components: a skeletal system providing the structural framework and a muscle model that simulates the contractile behavior of biological muscles to generate motion.

3.1.1 Muscle-tendon unit

The muscle–tendon unit (MTU) is a biomechanical component designed to simulate both the structure and function of muscle tissue [HZG90]. The core of MTU is the contractile element (CE) representing muscle fibers and a series elastic element (SE) representing tendons. To replicate the muscle's protective mechanisms, the model also includes a parallel elastic element (PE) and a buffer elastic element (BE). The complete MTU model is illustrated in Figure 3.1. The force produced by the MTU is given by $F_m = F_{se} = F_{ce} + F_{pe} - F_{be}$ and it is dependent on the muscle fiber length l_{ce}. The contractile force generated by the CE is computed using the formula:

$$F_{ce} = AF_{max} f_l(l_{ce}) f_v(v_{ce}) \tag{3.1}$$

with

$$f_l(l_{ce}) = exp\left(c \left| \frac{l_{ce} - l_{opt}}{\omega \cdot l_{opt}} \right|^3 \right) \tag{3.2}$$

and

$$f_v(v_{ce}) = \begin{cases} \dfrac{v_{max} - v_{ce}}{v_{max} + K v_{ce}}, & \text{if } v_{ce} < 0 \\ N + (N-1)\dfrac{v_{max} + v_{ce}}{7.56 K v_{ce} - v_{max}}, & \text{if } v_{ce} \geq 0 \end{cases} \tag{3.3}$$

where A is the activation, F_{max} is the maximum isometric force, $f_l(l_{ce})$ and $f_v(v_{ce})$ are functions describing the force–length and force–velocity relationships, l_{opt} is the optimal CE length for maximum force production, ω is the width of $f_l(l_{ce})$ curve, c is the residual force factor for $f_l(l_{ce})$, K is curvature constant and N accounts for eccentric force enhancement. F_{se}, F_{pe}, F_{be} correspond to forces from SE, PE and BE respectively.

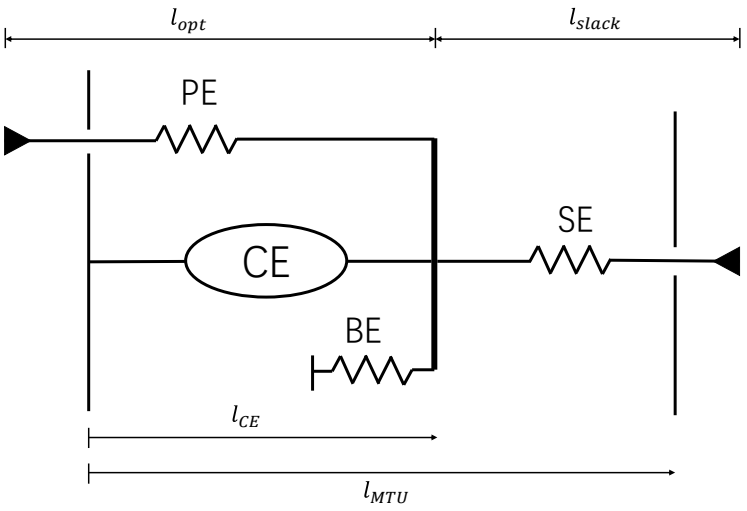

Figure 3.1: Structure of MTU. CE is the contractile element (muscle fiber) and SE is series elastic element (tendon). PE (parallel element) and BE (buffer elastic) only engage when l_{CE} falls outside the normal operating range to prevent the structure from collapse.

Aligning with CE, PE and BE help stabilize the structure of muscle unit. These elements only engage when CE is stretched or compressed beyond its normal operating range. PE and BE forces are calculated based on their respective deformation, using the definitions:

$$\begin{aligned} F_{be} &= F_{max}\left(\frac{l_{min} - l_{ce}}{l_{opt}\epsilon_{be}} \right)^2 \\ F_{pe} &= F_{max}\left(\frac{l_{ce} - l_{opt}}{l_{opt}\epsilon_{pe}} \right)^2 \end{aligned} \tag{3.4}$$

where $l_{min} = l_{opt} - \omega$ is the rest length of BE, the reference compression $\epsilon_{be} = \omega/2$ and $\epsilon_{pe} = \omega$ are the compression strains. The tendon part SE is modeled by a nonlinear function between the generated force and the deformation:

$$F_{se}(\epsilon) = \begin{cases} (\dfrac{\epsilon}{\epsilon_{ref}})^2, & \text{if } \epsilon > 0 \\ 0, & \text{if } \epsilon \leq 0 \end{cases} \tag{3.5}$$

where $\epsilon = (l_{se} - l_{slack})/l_{slack}$ is the tendon strain, l_{slack} is the tendon's slack length and ϵ_{ref} is the reference strain such that $F_{se}(\epsilon_{ref}) = 1$ [GH10; GSB03].

In addition to these basic components, the MTU model may also have a pennation angle between the muscle fibers (CE) and the tendon (SE). For muscles with a non-zero pennation angle, as shown in Figure 3.2, the contractile force is generated at an angle relative to the tendon. When the pennation angle is significant, all force components must be projected onto a common axis to ensure proper force transmission along the isometric direction. Aside from this geometric consideration, the functional behavior of the MTU remains unchanged [HZG90; Buc+04].

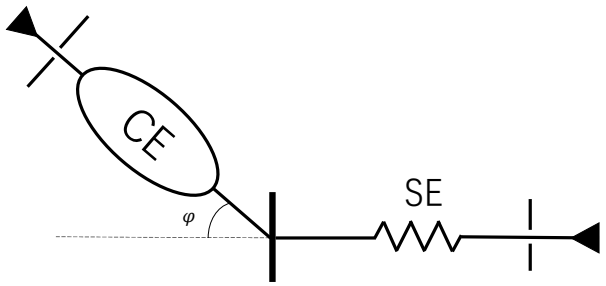

Figure 3.2: Structure of MTU with pennation angle. When non-negligible, the contractile force from CE should be projected onto the principle direction using φ.

3.1.2 Forward kinematics

With MTU working as actuators, the musculoskeletal model operates according to the following motion logic [Buc+04]. Upon receiving an excitation signal, the MTU undergoes a short delay corresponding to the muscle's reaction time, during which the excitation is transformed into activation. This activation initiates the contraction of the MTU, thereby generating force along the contractile direction. Given the geometry of the musculoskeletal model defined by its skeletal structure, the generated muscle forces produce joint torques based on their associated moment arms. These torques are then applied to the skeletal system and through the motion equations, the joint angular accelerations, angular velocities and angular degrees are computed. These dynamics result in the physical movement of the model. The entire process is illustrated in Figure 3.3.

There are different relationships between the excitation and activation, most of which involves a time delay, reflecting the physiological response of muscle tissue. This excitation contraction coupling can be described using a first-order differential equation:

$$\tau \frac{\mathrm{d}Act(t)}{\mathrm{d}t} = Exc(t) - Act(t) \tag{3.6}$$

where τ is the time constant representing the muscle's response delay. In this thesis, τ is set as 0.01 [GH10; GSB03].

In the simulated environment, the human walking gait is segmented into eight distinct phases: initial contact, loading response, mid stance, terminal stance, pre-swing, initial

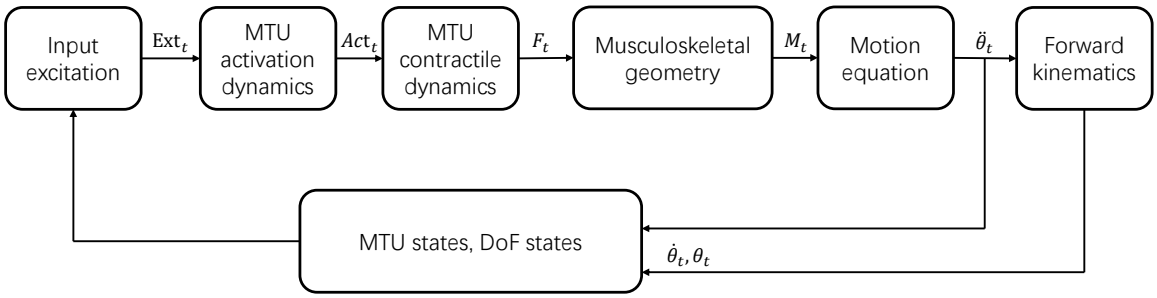

Figure 3.3: Forward kinematics of musculoskeletal model including biomechanical components.

swing, mid swing and terminal swing [PB24], as illustrated in Figure 3.4. The walking cycle is thus represented as a continuous loop traversing these phases sequentially from initiation to completion.

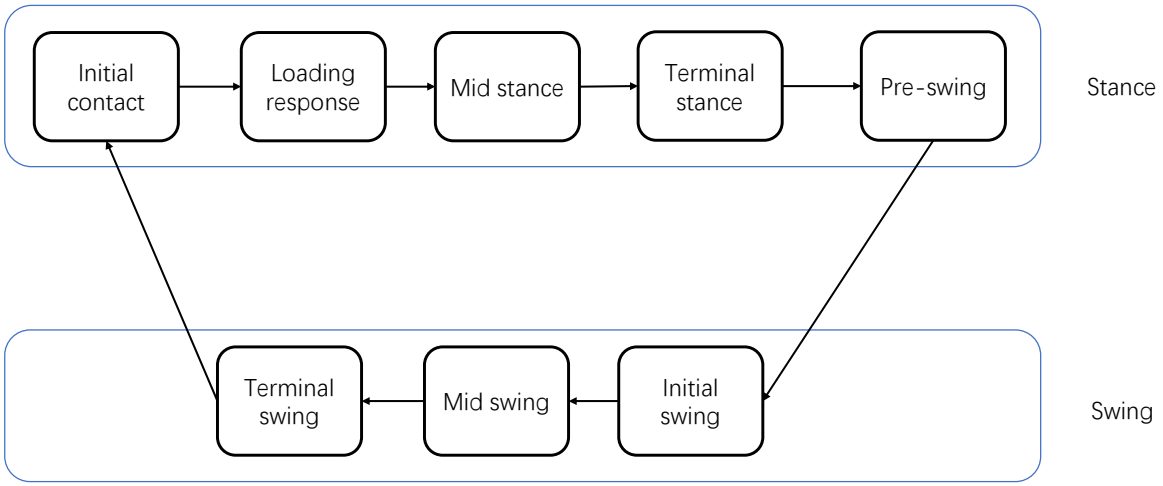

Figure 3.4: Eight phases of the walking gait cycle. Each stride begins with initial contact and ends with terminal swing.

3.2 Reinforcement learning

Driven by large volumes of data, machine learning typically focuses on tasks such as classification and regression by training neural networks on input datasets. However, reinforcement learning (RL), as a subfield of machine learning, distinguishes itself by involving interaction with the environment rather than relying solely on static data. Essentially, RL learns a mapping from the state space to the action space, guided by a reward signal that evaluates the quality of actions taken in different states. This framework allows an agent to improve its decision making policy through trial and error over time. The following statements are primarily based on the foundational concepts introduced in [SB+98].

3.2.1 Markov decision process

The RL process can be typically modeled as a markov decision process (MDP). In RL, the key components include the environment, the agent, states, actions and rewards. The objective

of RL is to train an optimal policy that enables the agent to perform actions that maximize its cumulative reward through interaction with the environment. As illustrated in Figure 3.5, at each time step t, the agent observes the current state $s_t \in S$ and selects an action $a_t \in A(s_t)$ to interact with the environment. Upon receiving the action a_t, the environment transitions to a new state $s_{t+1} \in S$ and provides a scalar reward through function $R : S \times A \times S \to \mathbb{R}$, which quantifies the desirability of the agent's action. The agent then uses this reward feedback to update its behavior in order to maximize the expected cumulative reward, commonly referred to as the return.

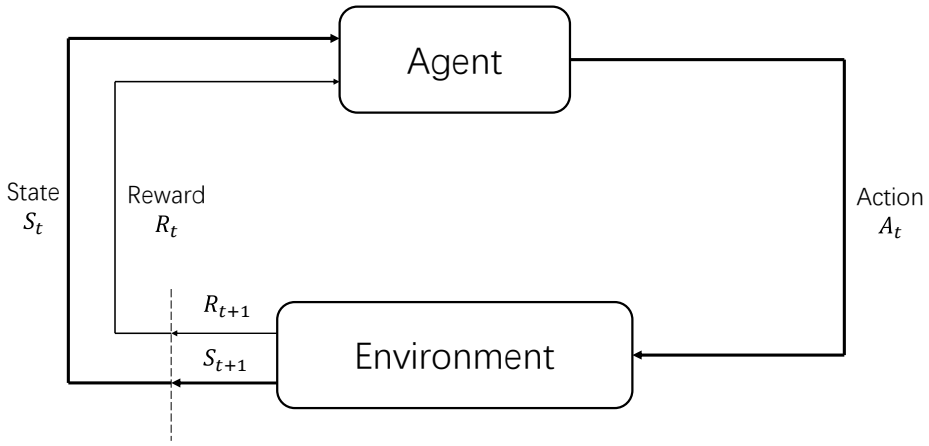

Figure 3.5: Framework of RL. The agent receives observations from the environment and output actions. The policy is updated based on reward signals provided by the environment.

The outcome of the RL is a policy $\pi_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ that defines the agent's decision making rule based on state observations. The policy could be deterministic, where actions are given by a function $a_t = \mu(s_t)$ or stochastic, where actions are sampled from a probability distribution $a_t \sim \pi_{\boldsymbol{\theta}}(\cdot|s_t)$. Through repeated interaction with the environment, a trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots)$ is collected as a sequence of states and actions. The initial state s_0 is drawn from a starting state distribution $\rho_0(\cdot)$. In stochastic environments, state transitions are modeled as $s_{t+1} \sim P(\cdot|s_t, a_t)$ where P denotes the state transition probability distribution.

The whole RL problems can be formally described as follows. Assuming both the environment transitions and the policy are stochastic, the probability of observing a trajectory τ over T time steps is given by:

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \tag{3.7}$$

The expected return of the policy $J(\pi)$ is then depicted as:

$$\begin{aligned} J(\pi) &= \int_{\tau} P(\tau|\pi)R(\tau) = \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)] \\ R(\tau) &= \sum_{t=0}^{\infty} \gamma^t r_t \end{aligned} \tag{3.8}$$

where $R(\tau)$ is the cumulative reward of the trajectory τ under the policy π and $\gamma \in (0, 1)$ is the discount that determines the relative importance of future rewards. Therefore, the central objective in RL is to find the optimal policy π^* that maximizes the expected return $J(\pi)$. The optimal policy is defined as:

$$\pi^* = \arg\max_{\pi} J(\pi) \tag{3.9}$$

3.2.2 Bellman equation

To identify the optimal policy, a common approach is to assign a value to each state indicating the expected cumulative reward an agent can obtain. This valuation guides the derivation of the optimal policy. Two key value functions are used to quantify the quality of states and actions within a policy framework. Assuming an infinite horizon setting, the V-value $V^\pi(s)$ under policy π represents the expected return starting from state s

$$V^\pi(s) = \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_0 = s] \tag{3.10}$$

and the Q-value $Q^\pi(s,a)$ measures the expected return of taking action a in state s according to the policy π:

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a] \tag{3.11}$$

Correspondingly, both $V^\pi(s)$ and $Q^\pi(s,a)$ achieve their maximum possible values when the policy is optimal:

$$\begin{aligned} V^*(s) &= \max_\pi \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_0 = s] \\ Q^*(s,a) &= \max_\pi \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a] \end{aligned} \tag{3.12}$$

Both V-value and Q-value can be updated iteratively using the Bellman equation to find the optimal values, thus extracting the optimal policy:

$$\begin{aligned} V^\pi(s) &= \mathop{\mathbb{E}}_{a\sim\pi, s'\sim P}[r(s,a) + \gamma V^\pi(s')] \\ Q^\pi(s,a) &= \mathop{\mathbb{E}}_{s'\sim P}[r(s,a) + \gamma \mathop{\mathbb{E}}_{a'\sim\pi}[Q^\pi(s',a')] \end{aligned} \tag{3.13}$$

where s' denotes the next state sampled from state transition and a' stands for the action for next state s' drawn from policy π. The optimal functions are therefore derived via:

$$\begin{aligned} V^*(s) &= \max_a \mathop{\mathbb{E}}_{s'\sim P}[r(s,a) + \gamma V^*(s')] \\ Q^*(s,a) &= \mathop{\mathbb{E}}_{s'\sim P}[r(s,a) + \gamma \max_{a'} Q^*(s',a')] \end{aligned} \tag{3.14}$$

3.2.3 REINFORCE

Instead of learning the policy indirectly through value functions, an alternative approach is to optimize the policy directly. Policy gradient tries to maximize the expected return by computing gradients of the performance objective with respect to the policy parameters. Starting from Equation 3.7, the log probability of a trajectory τ and its gradient with respect to the policy parameters $\boldsymbol{\theta}$ are given by:

$$\begin{aligned} \log P(\tau|\pi) &= \log \rho_0(s_0) + \sum_{t=0}^{T}(\log P(s_{t+1}|s_t, a_t) + \log \pi(a_t|s_t)) \\ \nabla_{\boldsymbol{\theta}} \log P(\tau|\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \log \rho_0(s_0) + \sum_{t=0}^{T}(\nabla_{\boldsymbol{\theta}} \log P(s_{t+1}|s_t, a_t) + \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t)) \\ &= \sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t) \end{aligned} \tag{3.15}$$

Combining with the trick from log derivative:

$$\nabla_{\boldsymbol{\theta}} P(\tau|\boldsymbol{\theta}) = P(\tau|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log P(\tau|\boldsymbol{\theta}) \tag{3.16}$$

The final update rules for policy parameters $\boldsymbol{\theta}$ are:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\tau\sim\pi_{\boldsymbol{\theta}}}[R(\tau)] \\
&= \nabla_{\boldsymbol{\theta}} \int_{\tau} P(\tau|\boldsymbol{\theta}) R(\tau) \\
&= \int_{\tau} \nabla_{\boldsymbol{\theta}} P(\tau|\boldsymbol{\theta}) R(\tau) \\
&= \int_{\tau} P(\tau|\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log P(\tau|\boldsymbol{\theta}) R(\tau) \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log P(\tau|\boldsymbol{\theta}) R(\tau)] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t) R(\tau)]
\end{aligned} \tag{3.17}$$

This forms the foundation of the REINFORCE algorithm, which is one of the earliest policy gradient methods. It enables direct optimization of the policy without requiring value function estimation. Due to its efficiency and simplicity, policy gradient methods are widely used in modern RL.

3.2.4 Policy gradient methods

To prevent the policy from converging to a suboptimal solution, Peters et al. proposed bounding the information loss during policy updates using a relative entropy term, while still enabling the policy to discover the optimal solution. This approach is known as relative entropy policy search (REPS) [PMA10]. The relative entropy boundary constraint is defined as

$$D(p^{\pi}\|q) = \sum_{s,a} \rho^{\pi}(s)\pi(a|s) \log \frac{\rho^{\pi}(s)\pi(a|s)}{q(s,a)} \leq \epsilon \tag{3.18}$$

where ρ^{π} is the state distribution of states following policy π, $\rho^{\pi}(s)\pi(a|s)$ is the new data distribution induced by the new policy π, $q(s,a)$ is the current observed data distribution and ϵ is a hyperparameter.

Inspired by REPS, trust region policy optimization (TRPO) is a widely adopted algorithm for solving RL problems. The fundamental idea behind TRPO is to maximize the performance improvement of the policy during each update, while ensuring that the updated policy does not deviate too far from the current policy. This constraint helps prevent instability or performance collapse due to overly aggressive updates. Let $\pi_{\boldsymbol{\theta}}$ denote the policy parameterized by $\boldsymbol{\theta}$. The goal is to find the optimal parameter $\boldsymbol{\theta}$ that maximizes a surrogate objective, often referred to as

$$\mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\theta}) = \mathop{\mathbb{E}}_{s,a\sim\pi_{\boldsymbol{\theta}_k}}[\frac{\pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}_k}(s,a)} A^{\pi_{\boldsymbol{\theta}_k}}(s,a)] \tag{3.19}$$

where A represents the advantage function under the old policy $\pi_{\boldsymbol{\theta}_k}$. This formulation forms the basis of the TRPO update mechanism, which seeks to improve the policy while staying within a trusted region of parameter space.

The TRPO policy update step is defined in Equation 3.20 where $\overline{D}_{KL}(\boldsymbol{\theta}_k, \boldsymbol{\theta})$ denotes the average KL-divergence between the current policy and the previous policy over the states sampled from the previous policy and δ is the upper boundary. In TRPO, the term of condi-

tional distribution $\pi_{\boldsymbol{\theta}_k}(a|s)$ is replaced with joint distribution $q(s,a)$

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_k, \boldsymbol{\theta}) \\ s.t. \overline{D}(\boldsymbol{\theta}_k, \boldsymbol{\theta}) &= \mathop{\mathbb{E}}_{s\sim\pi_{\boldsymbol{\theta}_k}} [\mathbb{KL}(\pi_{\boldsymbol{\theta}}(a|s)\|\pi_{\boldsymbol{\theta}_k}(a|s))] \leq \delta \end{aligned} \tag{3.20}$$

However, there are two main limitations of TRPO. First, the new policy may deviate too much from the old one when the probability ratio is either too large or too small, thus destablizing the training. Furthermore, due to the reliance on second-order optimization, the algorithm is computationally expensive and complex to implement. To address these issues, proximal policy optimization (PPO) is introduced. Derived from TRPO, both TRPO and PPO use the actor critic framework, in which a policy network and a value function are updated simultaneously. Although the two methods are conceptually similar, they differ in how the optimization problem is solved. PPO reformulates the problem using simpler, first-order optimization techniques and it introduces a clipping mechanism that approximates the trust region constraint without explicitly computing the KL-divergence. The goal is to maximize this objective with respect to the policy parameters $\boldsymbol{\theta}$, ensuring that the probability ratio between the new and old policies does not deviate too far from 1. This maintains the relative entropy in a stable range while also reduces computational overhead during policy updates. The clipped surrogate objective is defined in Equation 3.21

$$\mathcal{L}(s,a,\boldsymbol{\theta}_k,\boldsymbol{\theta}) = \min[\frac{\pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}_k}(s,a)} A^{\pi_{\boldsymbol{\theta}_k}}(s,a),\ g(\boldsymbol{\theta},\boldsymbol{\theta}_k,\epsilon,A^{\pi_{\boldsymbol{\theta}_k}}(s,a))] \tag{3.21}$$

with

$$g(\boldsymbol{\theta},\boldsymbol{\theta}_k,\epsilon,A) = CLIP(\frac{\pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}_k}(s,a)}, 1-\epsilon, 1+\epsilon)A \tag{3.22}$$

This formulation ensures that policy updates are conservatively constrained, preventing excessively large policy changes that could degrade performance. The final update is performed by maximizing the expected clipped objective:

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{s,a\sim\pi_{\boldsymbol{\theta}}} [\mathcal{L}(s,a,\boldsymbol{\theta}_k,\boldsymbol{\theta})] \tag{3.23}$$

Unlike the traditional advantage function, which is typically computed as the difference between the Q-value and the value estimated by the critic, the advantage function used in PPO in this thesis is based on generalized advantage estimation (GAE) [Sch+16]. GAE provides a more robust and flexible way to estimate advantages by using multiple steps of temporal difference (TD) errors along a trajectory. Specifically, it computes a weighted average of these TD errors, using an exponentially decaying factor to balance bias and variance. This method helps reduce the variance associated with advantage estimation. It proves useful when the value function is imperfect or noisy, as it enables more stable and efficient policy updates. The detailed procedure for computing GAE is presented in Algorithm 1 and the complete PPO training process is summarized in Algorithm 2.

3.3 Inverse reinforcement learning

Although RL employs neural networks to train agents through interaction with their environments, one of the most challenging aspects of RL is the design of the reward function. The reward function directly influences the agent's learned behavior and ultimately determines the success of the resulting policy. Since reward functions are typically task specific, effort is

Algorithm 1 GAE

Input: V-values for current states $\boldsymbol{V}$, V-values for next states $\boldsymbol{V}_{next}$, Rewards $\boldsymbol{R}$, Done flags $\mathbb{I}_{\textbf{Done}}$, TD error factor γ and GAE factor λ
Calculate TD errors $\boldsymbol{\delta} = \boldsymbol{R} + (\mathbf{1} - \mathbb{I}_{\textbf{Done}})\gamma \boldsymbol{V}_{next} - \boldsymbol{V}$
Initialize $\boldsymbol{GAE}$ with the same shape as $\boldsymbol{R}$
$\boldsymbol{GAE}_{-1} = \boldsymbol{\delta}_{-1}$
for $k = \ldots, 2, 1, 0$ **do**
    $\boldsymbol{GAE}_k = \boldsymbol{\delta}_k + \gamma\lambda(\mathbf{1} - \mathbb{I}_{\textbf{Done}})_{k+1}\boldsymbol{GAE}_{k+1}$
end for
Return: Normalized $\boldsymbol{GAE}$

Algorithm 2 PPO

Input: initial policy parameters $\boldsymbol{\theta_0}$, initial value parameters $\boldsymbol{\phi_0}$
for $k = 0, 1, 2, \ldots$ **do**
    Collect set of trajectories $\boldsymbol{D}_k = \{\boldsymbol{\tau}_i\}$ by applying policy $\pi_k = \pi(\boldsymbol{\theta}_k, state)$
    Collect corresponding rewards $\hat{\boldsymbol{R}}_t$
    Compute GAE $\boldsymbol{A}^{\pi_{\theta_k}}$ based on current value function $\boldsymbol{V}_{\boldsymbol{\phi}_k}$
    Update $\boldsymbol{\theta}$ by maximizing the PPO objective

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \frac{1}{|\boldsymbol{D}_k|T} \sum_{\tau \in \boldsymbol{D}_k} \sum_{t=0}^{T} \min\left[\frac{\pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}_k}(s,a)} \boldsymbol{A}^{\pi_{\theta_k}}(s,a),\ g(\boldsymbol{\theta}, \boldsymbol{\theta}_k, \epsilon, \boldsymbol{A}^{\pi_{\theta_k}}(s,a))\right] \tag{3.24}$$

    Update $\boldsymbol{\phi}$ by minimizing the mean square error

$$\boldsymbol{\phi}_{k+1} = \arg\min_{\boldsymbol{\phi}} \frac{1}{|\boldsymbol{D}_k|T} \sum_{\tau \in \boldsymbol{D}_k} \sum_{t=0}^{T} (\boldsymbol{V}_{\boldsymbol{\phi}_k}(s_t) - \hat{\boldsymbol{R}}_t)^2 \tag{3.25}$$

end for

often required to craft a well shaped reward for each new environment or application. This process is not only time consuming but also prone to suboptimal or biased design.

To address this limitation and promote greater generalization, inverse reinforcement learning (IRL) has been proposed. Instead of manually defining a reward function for every new scenario, IRL infers the reward function autonomously from expert demonstrations. Once the reward function is learned, it can be used within a standard RL framework to train the agent's policy. One of the approaches within IRL is the maximum entropy IRL algorithm. The core idea of this method is to find a reward function within a predefined function class that maximizes the difference in expected cost between expert and non-expert trajectories. In other words, the algorithm aims to assign lower costs to expert behaviors while penalizing other behaviors, thereby maximizing the margin between them. The optimization objective of maximum entropy IRL is formulated as:

$$\max_{R}(\min_{\pi} -H(\pi) + \mathbb{E}_{\pi}[R(s,a)]) - \mathbb{E}_{\pi_E}[R(s,a)] \tag{3.26}$$

where $H(\pi) = \mathbb{E}_{\pi}[-\log \pi(a|s)]$ is the entropy of policy π. The goal policy to be found is:

$$RL(R) = \arg\min_{\pi} -H(\pi) + \mathbb{E}_{\pi}[R(s,a)] \tag{3.27}$$

which leads to the policy having high entropy while being able to minimize the cumulative reward [Zie+08].

3.3.1 Generative adversarial imitation learning

Generative adversarial imitation learning (GAIL) is a method used for IRL or imitation learning [HE16]. In GAIL, which is inspired by generative adversarial networks (GAN) [Goo+20], the generator or policy attempts to produce actions that mimic the expert's state-action pairs, while the discriminator distinguishes between the observation through generated actions and those of the expert. The reward is given by the discriminator and actions that appear more expert-like receive higher rewards, guiding the policy toward expert behavior. The overall structure of this approach is illustrated in Figure 3.6. A suitable policy is learned once the discriminator can no longer differentiate between the agent and the expert. The discriminator's loss is computed as

$$Loss(D,\tau) = \min_{D} \mathop{\mathbb{E}}_{\tau\sim\pi}[-\log(D(s,a))] + \mathop{\mathbb{E}}_{\tau_E}[-\log(1-D(s,a))] \tag{3.28}$$

where $\tau \sim \pi$ refers to the trajectory generated by the agent's policy, τ_E represents the expert's trajectory, and $D(s,a)$ denotes the discriminator's score. In this thesis, PPO is deployed for the policy update step within the GAIL framework and the procedure is summarized in Algorithm 3.

3.3.2 Variational adversarial imitation learning

A notable variant of GAIL is variational adversarial imitation learning (VAIL), which incorporates a variational discriminator bottleneck (VDB) for improved IRL based control. The following explanation is based on the methodology presented in [Pen+19].

To enhance the generalization capability of the learned policy, VAIL introduces an encoder $E(z|x)$, which maps the input observation features x into a latent distribution Z. This encoder is integrated into the discriminator and forms a bottleneck structure that encourages the

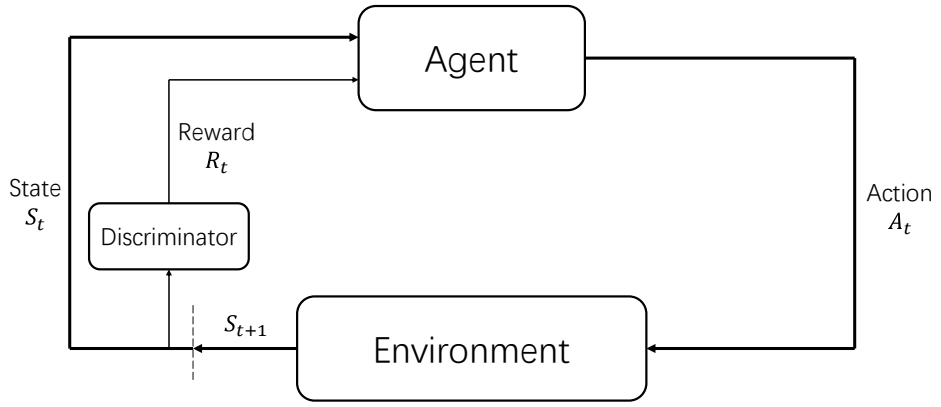

Figure 3.6: Framework of GAIL. The structure is similar to that of RL, but the reward signals are from a discriminator rather than from the environment.

Algorithm 3 GAIL

Input: initial policy parameters $\boldsymbol{\theta_0}$, initial value parameters $\boldsymbol{\phi_0}$, initial discriminator parameters $\boldsymbol{\omega_0}$
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\boldsymbol{D}_k = \{\boldsymbol{\tau}_i\}$ by applying policy $\pi_k = \pi(\boldsymbol{\theta}_k, state)$
  for $i = 0, 1, 2, \ldots$ **do**
    Update discriminator via

$$\boldsymbol{\omega}_{i+1} = \arg\min_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\tau \in \boldsymbol{D}_k} [-\log(1 - D_{\boldsymbol{\omega}}(s,a))] + \mathop{\mathbb{E}}_{\tau_E} [-\log(D_{\boldsymbol{\omega}}(s,a))] \tag{3.29}$$

  end for
  Compute the reward logits via discriminator

$$\hat{\boldsymbol{R}} = \log\big(D_{\boldsymbol{\omega}_k}(s,a)\big) \tag{3.30}$$

  Compute GAE $\boldsymbol{A}^{\pi_{\theta_k}}$ based on current value function $\boldsymbol{V}_{\boldsymbol{\phi}_k}$
  Update $\boldsymbol{\theta}$ by maximizing the PPO objective

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \frac{1}{|\boldsymbol{D}_k| T} \sum_{\tau \in \boldsymbol{D}_k} \sum_{t=0}^{T} \min\Big[\frac{\pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}_k}(s,a)} \boldsymbol{A}^{\pi_{\theta_k}}(s,a),\ g(\boldsymbol{\theta}, \boldsymbol{\theta}_k, \epsilon, \boldsymbol{A}^{\pi_{\theta_k}}(s,a))\Big] \tag{3.31}$$

  Update $\boldsymbol{\phi}$ by minimizing the mean square error

$$\boldsymbol{\phi}_{k+1} = \arg\min_{\boldsymbol{\phi}} \frac{1}{|\boldsymbol{D}_k| T} \sum_{\tau \in \boldsymbol{D}_k} \sum_{t=0}^{T} (\boldsymbol{V}_{\boldsymbol{\phi}_k}(s_t) - \hat{\boldsymbol{R}}_t)^2 \tag{3.32}$$

end for

network to focus on task relevant latent representations. Its goal is to first extract meaningful latent features via the encoder and then allow the discriminator to operate in the latent space to distinguish between the expert and the agent. This encourages smoother and more robust policy learning. The overall structure of this approach is illustrated in Figure 3.7.

To prevent the encoder from being influenced by irrelevant or noisy features, Alemi et al. [Ale+17] introduced an information bottleneck regularizer that constrains the mutual information $I(x,z)$. The purpose of this regularization is to encourage the encoder to retain only the discriminative features necessary for the task. Such mutual information $I(x,z)$ could be estimated by the KL-divergence [Pen+19; Csi75]. The objective becomes a constrained optimization problem, in which both the discriminator and the encoder are jointly optimized to minimize a loss composed of the standard GAIL objective with an added VDB constraint:

$$\begin{aligned} Loss(D,E) = \min_{D,E} \mathop{\mathbb{E}}_{s,a\sim\pi^*(s)}[\mathop{\mathbb{E}}_{z\sim E(z|s)}[-\log(D(z,a))]] + \mathop{\mathbb{E}}_{s,a\sim\pi(s)}[\mathop{\mathbb{E}}_{z\sim E(z|s)}[-\log(1-D(z,a))]] \\ s.t. \mathop{\mathbb{E}}_{s\sim\widetilde{\pi}(s)}[\mathbb{KL}[E(z|x)\|r(z)]] \leq I_c \end{aligned} \tag{3.33}$$

where $\widetilde{\pi} = \frac{1}{2}\pi^* + \frac{1}{2}\pi$ represents an even mixture of the agent data and expert data while I_c is the upper bound of the mutual information defined manually.

For the implementation of VDB, the constraint is integrated into the loss function to formulate a Lagrangian function with the Lagrange multiplier β

$$\begin{aligned} Loss(D,E,\beta) = \min_{D,E}\max_{\beta\geq 0} \mathop{\mathbb{E}}_{s,a\sim\pi^*(s)}[\mathop{\mathbb{E}}_{z\sim E(z|s)}[-\log(D(z,a))]] + \mathop{\mathbb{E}}_{s,a\sim\pi(s)}[\mathop{\mathbb{E}}_{z\sim E(z|s)}[-\log(1-D(z,a))]] \\ + \beta(\mathop{\mathbb{E}}_{s\sim\widetilde{\pi}(s)}[\mathbb{KL}[E(z|x)\|r(z)]] - I_c) \end{aligned} \tag{3.34}$$

For each updating step, the encoder is updated together with the discriminator and the multiplier β is updated adaptively based on the current mutual information constraint:

$$\begin{aligned} D,E &= \arg\min_{D,E} Loss(D,E,\beta) \\ \beta &= \max(0, \beta + \alpha_\beta(\mathop{\mathbb{E}}_{s\sim\widetilde{\pi}(s)}[\mathbb{KL}[E(z|x)\|r(z)]] - I_c)) \end{aligned} \tag{3.35}$$

where α_β is the step size for dual variable [Pen+19; BV04]. The reward signal for the agent is then derived from the discriminator using the latent representation of the state $r_t = -\log(1-D(\mu_E(s)))$ where μ is the mean of encoder's output distribution $E(z|x)$.

3.4 Long short-term memory

Long short-term memory (LSTM) consists of several recurrent units to process sequential data like text and speech, etc [Elm90]. As most vanilla RNNs suffer from vanishing gradient or cannot store the history information for too long [BSF94], the LSTM is designed to solve such a problem [HS97]. As depicted in Figure 3.8, the main idea of LSTM is to use a cell state and three different gates, each of which is a MLP followed by a sigmoid activation, to process the information. The LSTM maintains a cell state to function as a memory mechanism. The input gate determines how much of the current input should be stored in the cell state, the forget gate regulates the extent to which past information is discarded and the output gate controls what information is passed to the next time step.

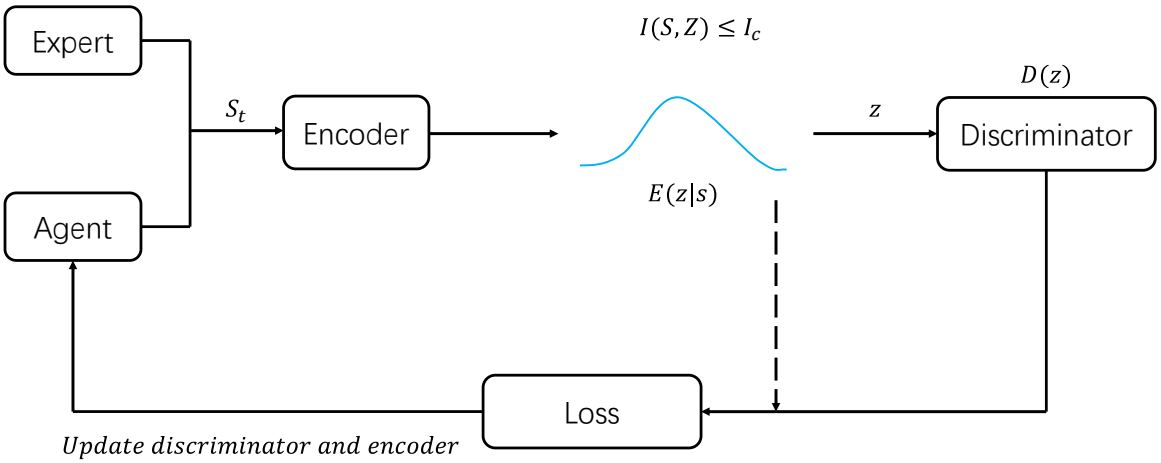

Figure 3.7: Scheme of VAIL. The basic structure is similar to that of GAIL, the discriminator is trained by the latent variables of encoder. The loss metric is minimized by discriminator $D(s,a)$ and encoder $E(z|s)$ jointly.

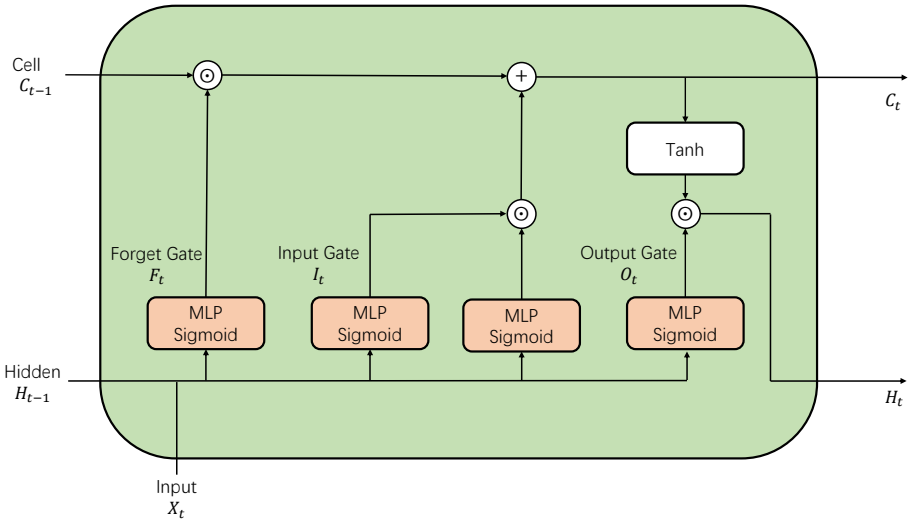

Figure 3.8: Framework of LSTM. It consists of a single cell and three gates to simulate the memory mechanism.

3.5 Dynamic time warping

Dynamic time warping (DTW) is a technique used widely in the area of speech recognition. It is an algorithm used for comparing the difference between two different time series with different lengths. The goal of DTW is to find the best matching pairs for the joints on two time series so that the overall distance metric of these two time series is minimized. Suppose there are two time sequences $\boldsymbol{\tau}_{\boldsymbol{x}} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_M\}$ and $\boldsymbol{\tau}_{\boldsymbol{y}} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots, \boldsymbol{y}_N\}$, a warping function is defined via:

$$\begin{aligned} \varphi(t) &= (\varphi_x(t), \varphi_y(t)), t = 1, 2, \cdots, T, \\ \varphi_x(t) &\in \{0, 1, \cdots, M\}, \\ \varphi_y(t) &\in \{0, 1, \cdots, N\}, \end{aligned} \tag{3.36}$$

where $\varphi_x(t)$ and $\varphi_y(t)$ represent the time steps of each time series respectively. The warping curve then aligns the time steps of points on trajectories along time series and the final matching result from DTW is to minimize the metric via:

$$Loss(\boldsymbol{\tau}_{\boldsymbol{x}}, \boldsymbol{\tau}_{\boldsymbol{y}}) = \min_{\varphi} \frac{\sum_{t=1}^{T} d(\boldsymbol{x}_{\varphi_x(t)}, \boldsymbol{y}_{\varphi_y(t)}) w(t)}{\sum_{t=1}^{T} w(t)} \tag{3.37}$$

where $d(x, y)$ is the objective distance of two single data points on each time series (for example, the squared cost or Manhattan distance). The weight $w(t)$ in the formula is a coefficient that assigns different weights to alignments. There are mainly two kinds of weighting strategies supported in DTW. The symmetric form assumes both trajectories are mapped onto a new time axis alone, and the integration of the loss term is made along the new axis. The asymmetric form, on the other hand, integrates the loss term only on one of the given trajectories. Figure 3.9 displays a simple instance of how the weighting coefficient is defined in both symmetric and asymmetric cases. Suppose the weights are independent from the trajectories. Then in the symmetric case, the $w(t)$ and the denominator in Equation 3.37 is defined by:

$$\begin{aligned} w(t) &= \varphi_x(t) - \varphi_x(t-1) + \varphi_y(t) - \varphi_y(t-1), \\ \sum_{t=1}^{T} w(t) &= \text{length of } \varphi_x + \text{length of } \varphi_y \end{aligned} \tag{3.38}$$

while the asymmetric form defines the same term as:

$$\begin{aligned} w(t) &= \varphi_x(t) - \varphi_x(t-1) \textbf{ or } \varphi_y(t) - \varphi_y(t-1), \\ \sum_{t=1}^{T} w(t) &= \text{length of } \varphi_x \textbf{ or } \text{length of } \varphi_y \end{aligned} \tag{3.39}$$

To ensure that the alignment process between two sequences remains meaningful and consistent with the temporal structure of the data, two fundamental constraints are imposed on the warping path. The monotonicity ensures that the alignment does not move backward in time

$$\begin{aligned} \varphi_x(t-1) &\leq \varphi_x(t), \\ \varphi_y(t-1) &\leq \varphi_y(t) \end{aligned} \tag{3.40}$$

and the continuity imposes the restriction that the alignment advances at most one time step in either sequence at each step

$$\begin{aligned} \varphi_x(t) - \varphi_x(t-1) &\leq 1, \\ \varphi_y(t) - \varphi_y(t-1) &\leq 1 \end{aligned} \tag{3.41}$$

In addition, the constraints on the boundary are introduced so as to make sure the starting points and the end points are always matched.

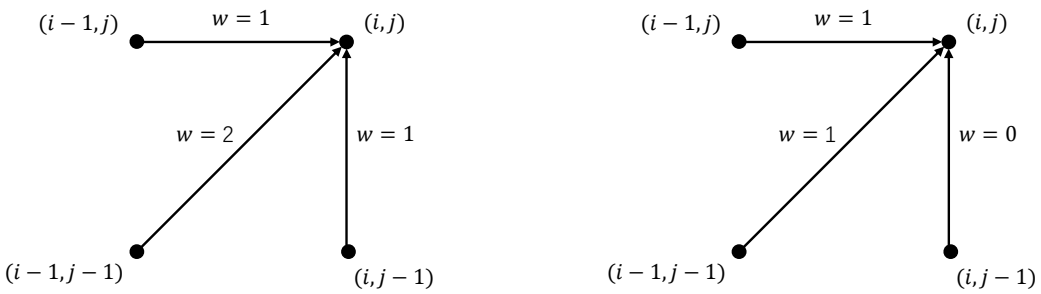

Figure 3.9: Two DTW weighting styles. The left one is symmetric style, with which equal weighting is applied to transitions from both trajectories. The right one is the asymmetric style, with which transitions are considered from only one trajectory.

Chapter 4

Methodology

In this section, the mechanism of the controllers is introduced first in section 4.1, followed by the experiments design for single environment in section 4.2 and for multiple environments in section 4.3.

4.1 Overview of model control strategy

The model is controlled by a combinatorial control strategy. The control strategy consists of two primary components: a low-level reflex controller for the muscles and a high-level joint motor controller.

The low-level controller operates as a finite state machine that can detect different phases of the gait cycle and applies phase-specific control parameters. To simplify the gait phases described in section 3.1, this thesis adopts a simplified version consisting of five phases: early stance, late stance, lift off, swing and landing. Transitions between phases are determined by the relative sagittal foot position and the normalized load on the leg. A phase transition is triggered when both variables reach predefined thresholds, details of which can be found in [Gei19].

For each phase of the gait, the low-level controller incorporates both a muscle reflex controller and a degrees of freedom (DoF) reflex controller, as proposed by Geyer and Herr [GH10]. The control law for the muscle reflex controller is defined as

$$U = C_0 + K_F[(F - F_0)]_+ + K_L[(L - L_0)]_+ + K_V[(V - V_0)]_+ \tag{4.1}$$

where K_F, F_0 represent the force feedback gain and offset; K_L, L_0 are the length feedback gain and offset; and K_V, V_0 correspond to the velocity feedback gain and offset. Here, F, L and V represent the real-time muscle force, muscle length and muscle contractile velocity respectively. The $[]_+$ operator ensures non-negative entries and the output U represents the excitation.

The control law for the DoF reflex controller is defined as

$$U = C_0 + K_P(P - P_0) + K_V(V - V_0) \tag{4.2}$$

where K_P, P_0 represent position feedback gain and target position, and K_V, V_0 represent velocity feedback gain and target velocity. The muscle is activated by the combined output of these reflex controllers, generating force to drive the model forward. Further details can be found in [GSB03; Buc+04].

While the low-level controller simulates the physiological behavior of muscles in response to muscle and joint states, the high-level controller governs motor actions based on a policy

trained using IRL algorithm GAIL. Extending IRL to an observation-only setting is straightforward by defining the reward function solely based on observations [Al-+23]. This approach is advantageous because intrinsic states, such as muscle activations or forces, may not be directly measurable [ABH05]. Even when these variables can be measured, differences in embodiment across cases typically require extensive task-specific parameter tuning [Ham+17; Qiu+20; GDG21]. Also, the process is simplified and consistent when using kinematic models to align the desired movement patterns. Therefore, IRL is a preferred method to match the feature space provided by the expert.

4.2 Single environment

To evaluate the IRL method in correcting pathological gait, the policy is first trained on each individual environment and later on all environments combined to obtain a unified policy. This thesis considers three distinct environments, each representing a different medical condition. During training, several techniques are employed to improve the performance gait correction. Section 4.2.1 describes the data normalization process used prior to generating outputs through the policy. Sections 4.2.2 and 4.2.3 present methods designed to mitigate instability during discriminator training.

4.2.1 Normalization

Although the policy takes observations from the model as input, results indicate that these observation values should be normalized before being passed to the policy. In this thesis, normalization is performed online i.e. observations are normalized at each time step during data collection. The normalization process is based on Welford's online algorithm, as described in Algorithm 4.

Algorithm 4 Welford online normalization

Input: $\bar{\boldsymbol{x}}_{old}$, $\boldsymbol{\sigma}_{old}$, $\bar{\boldsymbol{x}}_{current}$, $\boldsymbol{\sigma}_{current}$, $count_{old}$, $count_{current}$

$count_{update} = count_{old} + count_{current}$

$\boldsymbol{\delta} = \bar{\boldsymbol{x}}_{old} - \bar{\boldsymbol{x}}_{current}$

$\bar{\boldsymbol{x}}_{new} = \bar{\boldsymbol{x}}_{current} + \boldsymbol{\delta} * \frac{count_{current}}{count_{update}}$

$\boldsymbol{M}_a = \boldsymbol{\sigma}_{current} * count_{current}$

$\boldsymbol{M_b} = \boldsymbol{\sigma}_{old} * count_{old}$

$\boldsymbol{M}_2 = \boldsymbol{M}_a + \boldsymbol{M}_b + \sqrt{\boldsymbol{\delta}} * \frac{count_{current} * count_{old}}{count_{update}}$

$\boldsymbol{\sigma}_{new} = \frac{\boldsymbol{M}_2}{count_{update}}$

Return: $\bar{\boldsymbol{x}}_{new}$, $\boldsymbol{\sigma}_{new}$, $count_{update}$

4.2.2 Gradient penalty

Given the close relationship between GAIL and GAN frameworks, techniques developed for GANs can also be adapted to GAIL. One of the challenges in training GANs is instability and gradient penalty has been proposed as an effective solution to this problem [Gul+17].

In the study of robustness of machine learning, Lipschitz continuity is a property used to describe the robustness of neural networks. Consider a classifier $f : \mathbb{R}^d \to \mathcal{Y}$ and a dataset with $(x_i, y_i) \sim \mathbb{P}_{data}$, then a point $\widetilde{x} \in \mathbb{P}(x)$ is adversarial example for f at (x, y) if $f(x) = y$ but $f(\widetilde{x}) \neq y$. This illustrates how a small perturbation in the input space can lead to a significant change in the output. Lipschitz continuity formally constrains such behavior: A function $f : \mathcal{X} \to \mathcal{Y}$ is Lipschitz continuous if $\mathcal{D}_{\mathcal{Y}}(f(x_1), f(x_2)) \leq k \cdot \mathcal{D}_{\mathcal{X}}(x_1, x_2)$, where k is correspondingly the Lipschitz constant of f and f is k-Lipschitz. This condition ensures that the output of the function does not change faster than a linear function of the input change. Importantly, it has been shown that enforcing Lipschitz continuity is equivalent to constraining the gradient norm of the function to be less than or equal to k almost everywhere [Gou+21; HCC18]. This guarantees the worst-case change of the logits given a bounded perturbation of the input.

To enforce the Lipshitz constraint during GAIL training, one approach is to directly add a gradient penalty term into the discriminator's loss function, as expressed in Equation 4.3

$$Loss(D, \tau) = \min_D \mathop{\mathbb{E}}_{\tau \sim \pi}[-\log(D(s,a))] + \mathop{\mathbb{E}}_{\tau_E}[-\log(1 - D(s,a))] + \lambda \mathop{\mathbb{E}}_{\hat{\tau}}[(\|\nabla_{s,a} D(s,a)\| - k)^2] \quad (4.3)$$

where $\hat{\tau}$ is a random mix of the expert trajectory $\boldsymbol{\tau}_E$ and agent trajectory $\boldsymbol{\tau}$ [Gul+17], k is the Lipschitz constant and λ is the coefficient for the gradient penalty.

4.2.3 Spectral normalization

Apart from the gradient penalty introduced in section 4.2.2, another effective approach to constrain the discriminator and improve both stability and performance is to impose constraints directly on the network architecture. To stabilize the training process and control the output, Miyato et al. [Miy+18] proposed applying spectral normalization to each layer of the discriminator. This technique controls the Lipschitz constant of the discriminator by constraining the spectral norm of each layer $g : \boldsymbol{h}_{in} \mapsto \boldsymbol{h}_{out}$. Formally, the Lipschitz norm is defined as $\|g\|_{Lip} = \sup_{\boldsymbol{h}} \sigma(\nabla g(\boldsymbol{h}))$, where $\sigma(A)$ denotes the spectral norm of matrix A, as detailed in Equation 4.4.

$$\sigma(A) = \max_{\boldsymbol{h}:\boldsymbol{h}\neq\boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} = \max_{\|\boldsymbol{h}\|_2 \leq \boldsymbol{1}} \|\boldsymbol{A}\boldsymbol{h}\|_2 \quad (4.4)$$

Consider a discriminator has the following form in Equation 4.5

$$D(x) = \boldsymbol{W}^{L+1} a_L(\boldsymbol{W}^L(a_{L-1}(\boldsymbol{W}^{L-1}(\cdots a_1(\boldsymbol{W}^1 x + \boldsymbol{b}^1)\cdots) + \boldsymbol{b}^{L-1})) + \boldsymbol{b}^L) + \boldsymbol{b}^{L+1} \quad (4.5)$$

where $\boldsymbol{W}^l$, $\boldsymbol{b}^l$ and a_l stand for weights, bias of transfer layers and the activation layers respectively. Then, the Lipschitz norm of a discriminator can be calculated by using

$$\begin{aligned} \|D(x)\|_{Lip} &\leq \|\boldsymbol{W}^{L+1}\|_{Lip} \cdot \|a_L\|_{Lip} \cdot \|\boldsymbol{W}^L\|_{Lip} \cdots \|\boldsymbol{W}^1\|_{Lip} \cdot \|a_0\|_{Lip} \cdot \|\boldsymbol{W}^0\|_{Lip} \\ &= \prod_0^L k_{\boldsymbol{W}}^l k_a^l \sigma(\boldsymbol{W}) \sigma(a) \end{aligned} \quad (4.6)$$

With the spectral norm layer, the weights in the discriminator are rescaled with the spectral norm σ of the weight matrix approximated using power iteration method. During each training iteration, the parameters are updated according to the following formulas

$$\boldsymbol{W}_{SN} = \frac{\boldsymbol{W}}{\sigma(\boldsymbol{W})}, \sigma(\boldsymbol{W}) = \max_{\boldsymbol{h}:\boldsymbol{h}\neq\boldsymbol{0}} \frac{\|\boldsymbol{W}\boldsymbol{h}\|_2}{\|\boldsymbol{h}\|_2} \quad (4.7)$$

4.3 Multiple environments

After having trained the policy successfully on each single environment, a unified policy is further trained across all three environments. However, directly applying the same framework used for single environment training fails to reconstruct the intended gait rehabilitation effects, as the discriminator is able to easily distinguish the agent's behavior from that of the expert. To address this limitation, additional techniques are introduced to assist the PPO network in learning a generalizable policy. Section 4.3.1 presents the technique used to stabilize model motion, while section 4.3.2 and section 4.3.3 introduce the methods for policy training to generalize its availability across diverse environments.

4.3.1 Action chunking

When the control strategy is deployed at a high frequency, the model may suffer from instability or inefficient execution due to excessive sensitivity to small fluctuations in the input. Although relatively simple, action chunking has proven to be a solution to mitigate this issue [Zha+23]. As depicted in Figure 4.1, the key of action chunking is to group a sequence of actions across multiple time steps into a single unit, treating them as one action for the current time step [LHG25]. With action chunking integrated into the policy network, the policy $\pi(a_{t:t+k}|s_t)$ generates actions once for k steps in the future based on the current state s_t, rather than the standard formulation $\pi(a_t|s_t)$, which outputs only one action per time step. The agent then executes the entire chunk of k actions sequentially and transitions directly to the state at time step $t+k$. To further smooth the actions and reduce abrupt changes in motion, temporal ensembling is introduced as a filtering mechanism. This method produces a weighted action at each time step by aggregating previous actions generated across time. The weights are assigned exponentially by $w_i = exp(-m * i)$, where w_0 corresponds to the weight of the eldest action and the decrease coefficient m controls how rapidly the influence of elder actions diminishes [Zha+23].

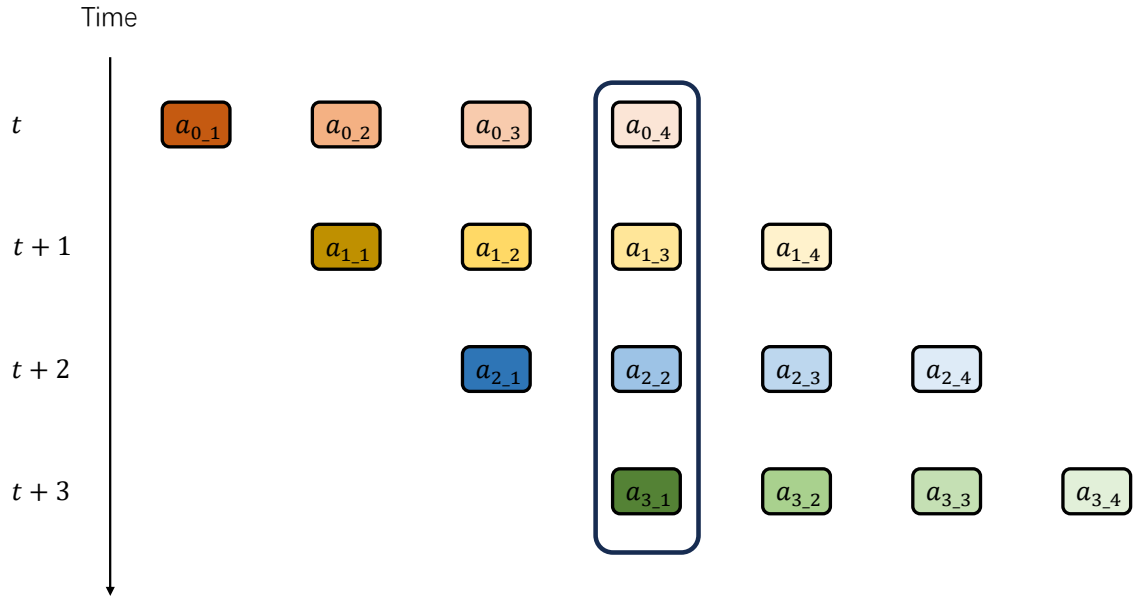

Figure 4.1: Scheme of action chunking. At each time step, the agent predicts the actions for the next k steps and the actions executed in environment are grouped as ensemble.

4.3.2 Combining with VDB

Several methods have shown their success, for example, by augmenting the observation space with additional contextual information [LSE17] or by designing special neural network architectures that make the learned policy adaptable [Boh+24]. We adopt the VDB in an attempt to improve the performance of our policy. Since the gradient penalty plays an important role in stabilizing training, it is used alongside the VDB framework. Similar to the variational autoencoder, we can apply the reparameterization trick to optimize the parameters of the encoder and decoder (or discriminator in our case). Reparameterization reformulates the sampling process by expressing the random variable as a deterministic function of fixed noise and learnable parameters, thereby enabling gradient flow through stochastic layers. With the reparameterization trick applied, the loss function for the discriminator becomes

$$\begin{aligned} Loss(D,E) = \min_{D,E} & \mathop{\mathbb{E}}_{s\sim\pi^*(s)}[\mathop{\mathbb{E}}_{z\sim E(z|s)}[-\log(D(z))]] + \mathop{\mathbb{E}}_{s\sim\pi(s)}[\mathop{\mathbb{E}}_{z\sim E(z|s)}[-\log(1-D(z))]] \\ & + w_{gp} \mathop{\mathbb{E}}_{\substack{s\sim\pi^*(s)\\ \epsilon\sim\mathcal{N}(0,I)}} [(\|\nabla_s D(\mu_E(s)+\Sigma_E(s)\epsilon)\| - \kappa)^2] \\ & s.t. \mathop{\mathbb{E}}_{s\sim\widetilde{\pi}(s)}[\mathbb{KL}[E(z|s)\|r(z)]] \leq I_c \end{aligned} \tag{4.8}$$

where w_{gp} is the coefficient for gradient penalty and $\widetilde{\pi}(s)$ is a balanced mix of expert data and agent data. To summarize, the whole procedure of training one unified policy on different environments with VDB integrated is as follow in Algorithm 5.

4.3.3 Combining with LSTM

Another approach to training a unified policy for all environments is involving an LSTM network [HS97], which first serves as an encoder. The latent representations produced by the LSTM are then used to train the discriminator, actor and critic. Theoretically, since LSTM is able to capture temporal dependencies by maintaining a memory of history observations, it is also possible to use LSTM to generalize the feasibility of the policy on different environments. The overall training process with LSTM follows the same structure as outlined in Algorithm 6, with the important distinction that the gradient penalty is applied to the latent variables generated by the LSTM, rather than directly to the raw observational data from the agent and expert trajectories.

Algorithm 5 VAIL with gradient penalty

Input: initial policy parameters $\boldsymbol{\theta_0}$, initial value parameters $\boldsymbol{\phi_0}$, initial discriminator parameters $\boldsymbol{\omega_0}$, initial encoder parameters $\boldsymbol{\eta_0}$, mutual information I_c, dual step size α_β
for $k = 0, 1, 2, \dots$ **do**
  Collect set of trajectories $D_k = \{\boldsymbol{\tau}_i\}$ by applying policy $\pi_k = \pi(\boldsymbol{\theta}_k, state)$
  for $i = 0, 1, 2, \dots$ **do**
    Sample $z \sim E_{\boldsymbol{\eta}_i}(z|s)$ using reparameterization trick
    Update encoder and discriminator with defined w_{gp} and κ

$$\begin{aligned}\boldsymbol{\omega}_{i+1}, \boldsymbol{\eta}_{i+1} = \arg\min_{\boldsymbol{\omega},\boldsymbol{\eta}} & \mathop{\mathbb{E}}_{\tau\in\boldsymbol{D}_k}[-\log(1-D_{\boldsymbol{\omega}}(z))] + \mathop{\mathbb{E}}_{\tau_E}[-\log(D_{\boldsymbol{\omega}}(z))] \\ & + w_{gp} \mathop{\mathbb{E}}_{\substack{s\sim\pi^*(s)\\ \epsilon\sim\mathcal{N}(0,I)}} [(\|\nabla_s D(\mu_{E_\eta}(s) + \Sigma_{E_\eta}(s)\epsilon)\| - \kappa)^2] \\ & + \beta \mathop{\mathbb{E}}_{\tau\cup\tau_E}[\mathbb{KL}(E_{\boldsymbol{\eta}_i}(z|s)\|N(0,I) - I_c]\end{aligned} \tag{4.9}$$

    Adaptively update Lagrange multiplier β

$$\beta = \max(0, \beta + \alpha_\beta(\mathop{\mathbb{E}}_{s\sim\tilde{\pi}(s)}[\mathbb{KL}[E(z|s)\|N(0,I)]] - I_c)) \tag{4.10}$$

  end for
  Compute the reward logits via discriminator with $z \sim E_{\boldsymbol{\eta}_k}(z|s)$

$$\hat{\boldsymbol{R}} = \log\big(D_{\boldsymbol{\omega}_k}(z)\big) \tag{4.11}$$

  Compute GAE $\boldsymbol{A}_t$ based on current value function $\boldsymbol{V}_{\boldsymbol{\phi}_k}$
  Update $\boldsymbol{\theta}$ by maximizing the PPO objective

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \frac{1}{|\boldsymbol{D}_k|T} \sum_{\tau\in\boldsymbol{D}_k} \sum_{t=0}^{T} \min[\frac{\pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}_k}(s,a)} \boldsymbol{A}^{\pi_{\boldsymbol{\theta}_k}}(s),\ g(\boldsymbol{\theta}, \boldsymbol{\theta}_k, \epsilon, \boldsymbol{A}^{\pi_{\boldsymbol{\theta}_k}}(s))] \tag{4.12}$$

  Update $\boldsymbol{\phi}$ by minimizing the mean square error

$$\boldsymbol{\phi}_{k+1} = \arg\min_{\boldsymbol{\phi}} \frac{1}{|\boldsymbol{D}_k|T} \sum_{\tau\in\boldsymbol{D}_k} \sum_{t=0}^{T} (\boldsymbol{V}_{\boldsymbol{\phi}_k}(s_t) - \hat{\boldsymbol{R}}_t)^2 \tag{4.13}$$

end for

Algorithm 6 GAIL with LSTM and gradient penalty

Input: initial policy parameters $\boldsymbol{\theta_0}$, initial value parameters $\boldsymbol{\phi_0}$, initial discriminator parameters $\boldsymbol{\omega_0}$, initial LSTM parameters $\boldsymbol{\eta}$
for $k = 0, 1, 2, \dots$ **do**
    Collect set of trajectories $D_k = \{\boldsymbol{\tau}_i\}$ by applying policy $\pi_k = \pi(\boldsymbol{\theta}_k, state)$
    Get latent variables $z = L_{\boldsymbol{\eta_k}}(s)$ from LSTM network
    for $i = 0, 1, 2, \dots$ **do**
        Update discriminator with defined w_{gp} and κ

$$\begin{aligned}\boldsymbol{\omega}_{i+1} = \arg\min_{\boldsymbol{\omega}} & \mathop{\mathbb{E}}_{\tau \in \boldsymbol{D}_k}[-\log(1 - D_{\boldsymbol{\omega}}(z))] + \mathop{\mathbb{E}}_{\tau_E}[-\log(D_{\boldsymbol{\omega}}(z))] \\ & + w_{gp} \mathop{\mathbb{E}}_{\hat{\tau}}[(\|\nabla_z D(z)\| - \kappa)^2]\end{aligned} \tag{4.14}$$

    end for
    Compute the reward logits via discriminator

$$\hat{\boldsymbol{R}} = \log\big(D_{\boldsymbol{\omega}_k}(z)\big) \tag{4.15}$$

    Compute GAE $\boldsymbol{A}_t$ based on current value function $\boldsymbol{V}_{\boldsymbol{\phi}_k}$
    Update $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ by maximizing the PPO objective

$$\boldsymbol{\theta}_{k+1}, \boldsymbol{\eta}_{k+1} = \arg\max_{\boldsymbol{\theta}, \boldsymbol{\eta}} \frac{1}{|\boldsymbol{D}_k| T} \sum_{\tau \in \boldsymbol{D}_k} \sum_{t=0}^{T} \min\Big[\frac{\pi_{\boldsymbol{\theta}}(z, a)}{\pi_{\boldsymbol{\theta}_k}(z, a)} \boldsymbol{A}^{\pi_{\boldsymbol{\theta}_k}}(z),\ g(\boldsymbol{\theta}, \boldsymbol{\theta}_k, \epsilon, \boldsymbol{A}^{\pi_{\boldsymbol{\theta}_k}}(z))\Big] \tag{4.16}$$

    Update $\boldsymbol{\phi}$ by minimizing the mean square error

$$\boldsymbol{\phi}_{k+1} = \arg\min_{\boldsymbol{\phi}} \frac{1}{|\boldsymbol{D}_k| T} \sum_{\tau \in \boldsymbol{D}_k} \sum_{t=0}^{T} (\boldsymbol{V}_{\boldsymbol{\phi}_k}(z_t) - \hat{\boldsymbol{R}}_t)^2 \tag{4.17}$$

end for

Chapter 5

Experiments

Building upon the methods described in chapter 4, this chapter presents the results of the experiments. Section 5.1 first introduces closely the metric used in the evaluations, followed by section 5.2 which discusses the experimental results in detail.

5.1 Metric

As it has been observed after training that the corrected gaits may exhibit different frequencies compared to those of the expert. The DTW loss metric is employed to better quantify the training performance and mitigate biases caused by frequency mismatches and temporal shifts. The distance $d(x, y)$ is computed using the square error over all combined features jointly. During the alignment process, since the current state can be derived from the previous state, the optimization problem can be solved using dynamic programming (DP) via the formulation

$$\begin{aligned} g(\varphi(t)) &= \min_{\varphi(t-1)} g(\varphi(t-1)) + d(\boldsymbol{x}_{\varphi_x(t)}, \boldsymbol{y}_{\varphi_y(t)}) w(t) \\ Loss(\boldsymbol{\tau}_{\boldsymbol{x}}, \boldsymbol{\tau}_{\boldsymbol{y}}) &= \frac{g(T)}{\sum_{t=1}^{T} w(t)} \end{aligned} \tag{5.1}$$

with the initial states as

$$g(\varphi(1)) = d(\boldsymbol{x}_{\varphi_x(1)}, \boldsymbol{y}_{\varphi_y(1)}) w(1) \tag{5.2}$$

This thesis adopts a symmetric weighting scheme, in which matching between data points at the same time step is neither explicitly encouraged nor discouraged. Under this approach, the transition cost is uniformly applied, ensuring temporal neutrality in the alignment. The transition formula is defined as follows and applies to all states, including the initial states

$$g(i, j) = \min \begin{cases} g(i, j-1) + d(i, j) \\ g(i-1, j-1) + d(i, j) \\ g(i-1, j) + d(i, j) \end{cases} \tag{5.3}$$

To better process the data alignment, the DTW algorithm is adapted to an open end setting. In this configuration, the constraint requiring alignment of the endpoints of both the query and reference trajectories is relaxed. The final algorithm used for evaluation is presented in Algorithm 7.

Algorithm 7 DTW with open end setting

Input: Two sequences $A = (\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n})$ and $B = (\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_m})$
Initialize cost matrix $D \in \mathbb{R}^{(n+1)\times(m+1)}$ with $D(0,0) = 0$ and $D(i,0) = D(0,j) = \infty$ for $i = 1..n$, $j = 1..m$
for $i = 1, 2, \cdots, n$ **do**
    for $j = 1, 2, \cdots, m$ **do**
        $cost \leftarrow d(a_i, b_j)$
        $D(i,j) \leftarrow cost + \min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\}$
    end for
end for
return $D(-1, \arg\min_k D(-1,k))$

5.2 Results

To evaluate the possibility of using GAIL and its variants to train policies for correcting pathological gaits, experiments are conducted in both single environment and multiple environments. Section 5.2.1 presents the results obtained in a single environment, while section 5.2.2 further explores performance across multiple environments, including zero-shot generalization scenarios.

5.2.1 Single environment

This section presents the evaluation results for each individual environment. The aim is to demonstrate the necessity of integrating additional techniques like gradient penalty or spectral normalization into the discriminator. To assess the feasibility of the IRL control method, we simulate scenarios where an exoskeleton is deployed on a patient model exhibiting pathological gait caused by three distinct impairments: short hamstring, hamstring weakness and iliopsoas weakness. Each condition represents a medically recognized musculoskeletal disorder [Aal+25; Att+19; Bel+24; Aka+16; The+05]. The specific model parameters used to simulate each impairment are detailed in Table 5.1.

Environment	Model setting
Short hamstring	80% original optimal CE length of hamstring MTU
Hamstring weakness	20% original max. isometric force of hamstring MTU
Iliopsoas weakness	50% original max. isometric force of iliopsoas MTU

Table 5.1: Impairments in environments used for training.

All three environments share the same fundamental configuration. The model used in this thesis is a bipedal musculoskeletal humanoid model, implemented in SCONE. SCONE is a free and open source software used specifically for predictive simulation in the area of human and animal motion [Gei19]. To leverage the accuracy of OpenSim [Del+07] and the computational efficiency of Mujoco [TET12], Hyfydy is developed as a plugin for the SCONE [Gei21]. All the experiments conducted in this thesis are run with the Hyfydy license under SCONE. The musculoskeletal model is based on the *H*0914 model provided in SCONE. In addition to the MTUs, four joint actuators are applied directly to the hips and knees in the sagittal plane, simulating functionality of the lower limb exoskeleton. The simulated

exoskeleton assists movement by the manipulation on hips and knees. All the four motors are assumed to share the same rotational axis as their corresponding joints. The torque output of each motor ranges from –50 to 50 N•m and is bounded using *Tanh* function to ensure smooth saturation. The objective of the simulation is to correct pathological gait patterns and assist the patient in walking with a gait resembling that of a healthy individual. The biomechanical model includes 9 DoF and 14 MTUs, which function as muscle actuators. A reference dataset is obtained from a healthy model walking forward at an average speed of 1.2 m/s, controlled solely by reflex-based controller with no external torque applied at the joints. As an instance, we focus on the environment simulating a short hamstring condition. To simulate this pathological gait, the optimal length of the hamstring MTU in the patient model is reduced by 20%, mimicking a common medical condition. An illustration comparing the healthy and pathological gait patterns is shown in Figure 5.1.

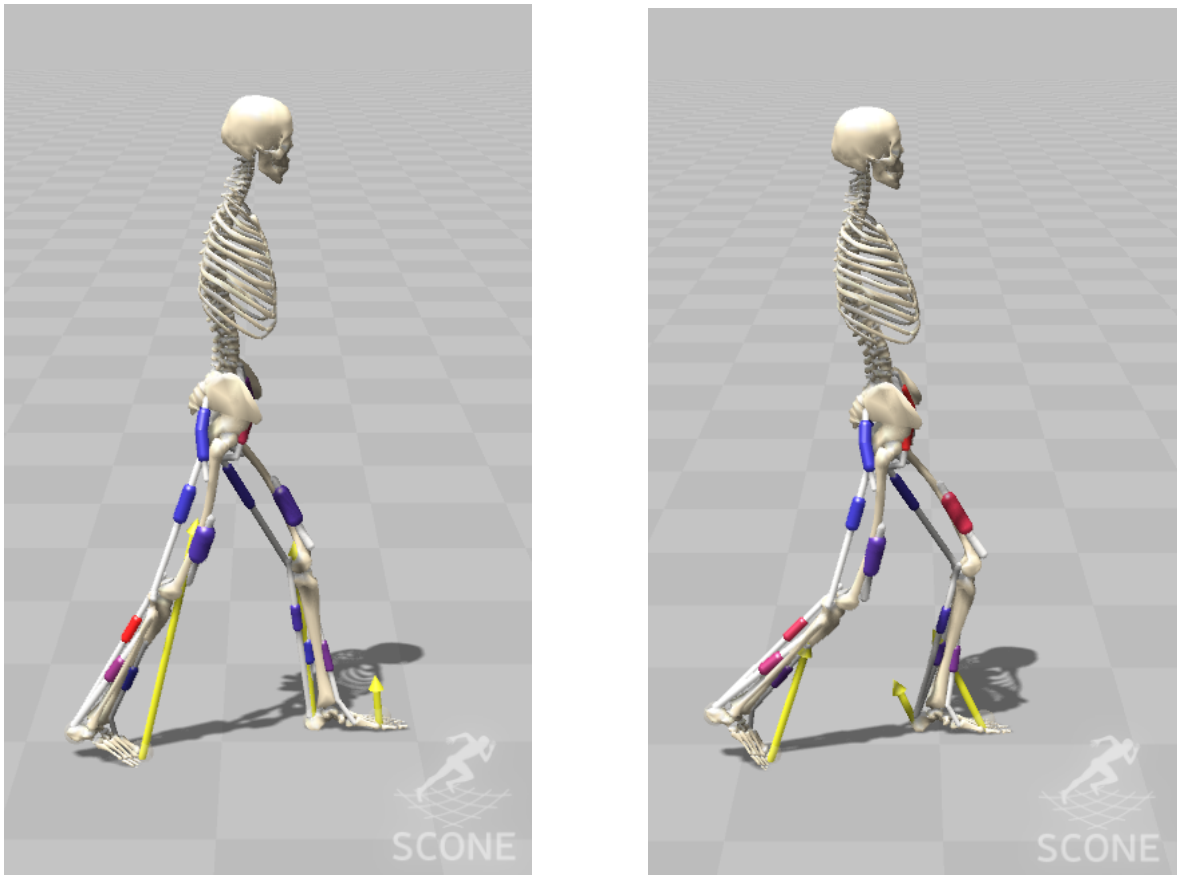

Figure 5.1: Expert gait (left) and pathological gait (right) with short hamstring whose MTU has 20% shorter optimal length. Both models are controlled solely by the low-level controller

Based on the patient model which is controlled exclusively by the low-level reflex controller, the policy of the exoskeleton is trained through IRL. Both the actor and critic networks are comprised of two hidden layers, with 512 units each. The actor network receives an 18-dimensional observation vector, which includes joint angles and joint velocities. The outputs is a 4-dimensional action corresponding to the external torques applied to the joints. The discriminator network is composed of two hidden layers, with 64 units each. For the experiments with gradient penalty, the coefficient is selected as $w_{gp} = 1$ and $\kappa = 0$.

The agent is trained for 1.8×10^7 steps (approximately 5 hours). Figure 5.3 presents a comparison of joint trajectories from the expert, the impaired gait and the corrected gait. It can be seen that the corrected gait resembles the healthy expert gait more closely in terms

of both alignment and magnitude. In addition, a series of screenshots from video comparing the gaits are shown in Figure 5.2. In these visuals, both the original pathological gait and the corrected gait are overlaid on the expert gait. The animation video has been processed using the DTW algorithm, so the alignments are plotted based on the matched pairs identified by DTW. These time sequenced screenshots demonstrate that the corrected gait achieves a better alignment with the expert gait compared to the original pathological gait.

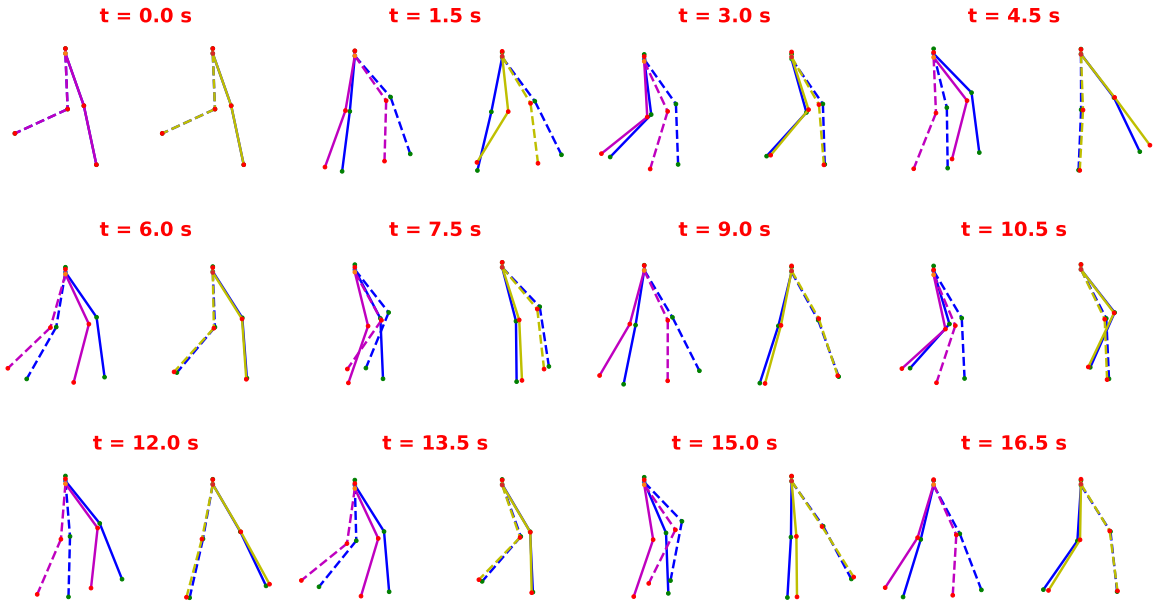

Figure 5.2: Overlaid stick figures comparing gaits at different time steps along the trajectory: healthy gait (blue), pathological gait (purple) and corrected gait (yellow). The left panel compares pathological vs. healthy, the right panel compares corrected vs. healthy. The gait has the impairment with shortened hamstring. Both are aligned using DTW.

To provide a detailed presentation of the DTW matching results, Figure 5.4 demonstrates an example of the alignment between the observation trajectories of the expert and those of the patient after being corrected by the policy. Table 5.2 compares the differences between pathological vs. healthy and corrected vs. healthy gait with all kinds of policy learning networks. The losses are all computed by summing all feature terms, averaged over 10 rollouts and all timesteps. Joint trajectories consist of values of each joint in [rad]. These quantitative results confirm that the corrected gait more closely approximates the expert trajectories compared to the pathological gait. The comparative loss curves of using different techniques are shown in Figure 5.5. The results indicate that the policy trained with gradient penalty achieves the steepest decrease in loss and exhibits greater stability and DTW loss is a better metric for the training evaluation.

Similarly, the same evaluations are conducted on medical cases where the model has weaker hamstring or iliopsoas muscles. The corresponding trajectory plots, loss curves and loss tables are presented in the Appendix.

5.2.2 Multiple environments

After section 5.2.1 having proved the feasibility of using IRL method to train the policy, this section first presents the evaluation results of the trained unified policy on the same environments used during training. To further assess the generalization capability of the policy,

Algorithm	Gait	MSE	MAE	DTW
Vanilla	Pathological	0.967 ± 0.301	1.975 ± 0.339	0.530 ± 0.015
	Corrected	0.902 ± 0.171	1.708 ± 0.233	0.190 ± 0.037
With spectral norm	Pathological	0.950 ± 0.287	1.953 ± 0.328	0.529 ± 0.019
	Corrected	1.105 ± 0.048	1.961 ± 0.049	0.413 ± 0.054
With gradient penalty	Pathological	0.909 ± 0.300	1.904 ± 0.344	0.530 ± 0.015
	Corrected	0.995 ± 0.137	1.890 ± 0.163	0.173 ± 0.130
With spectral norm and gradient penalty	Pathological	0.912 ± 0.291	1.907 ± 0.335	0.529 ± 0.019
	Corrected	0.941 ± 0.040	1.793 ± 0.050	0.139 ± 0.014

Table 5.2: Quantified comparisons of different training techniques on the corrected gait at training step 1.8×10^7. The gait has the impairment with shortened hamstring.

zero-shot experiments are conducted to evaluate its performance in unseen environments. As there is little difference between the method of using purely gradient penalty and using both the gradient penalty and spectral norm, following framework only uses the gradient penalty technique for the ease of computation.

VDB

The first approach used to train a unified policy is to integrate the VDB technique. The idea of VDB is training the discriminator on latent variables obtained from an encoder and using the resulting reward to guide policy learning. Using the same models and three environments described in Table 5.1, the mutual information constraint is set to $I_c = 0.1$ and the KL-divergence term is symmetrically computed from both agent and expert data. The latent space of the encoder is 8-dimensional. The input and output dimensions of the policy remain consistent with those used in the single environment experiments. As demonstrated in section 5.2.1 that the inclusion of a gradient penalty is critical during training, it is incorporated into the discriminator's loss function with a weighting factor $w_{gp} = 1$ and $\kappa = 0$. Additionally, to address redundant walking behavior observed when applying the original action output at each time step, action chunking and ensemble techniques are introduced. Specifically, the policy outputs actions for the next four time steps at each time step, serving as a filter to improve motion smoothness.

Similar to Section 5.2.1, Figure 5.6 shows the trajectories of the corrected gait in an environment with shorter hamstring as an example, while Figure 5.7 presents the corresponding DTW alignment plot between the agent's and expert's trajectories. To provide an overview, Figure 5.8 shows the DTW loss curves across of all the three environments and Table 5.3 quantifies these losses. The results demonstrate the success of employing VDB techniques to train a unified policy in multiple environments. All the other materials are provided in the Appendix.

LSTM

The second method employed to train a unified policy is to integrate a LSTM module before feeding the observation data into the networks. The LSTM used in the experiments has a latent space of 128 dimensions. Unlike the VDB approach, where latent variables are used exclusively for training the discriminator, all observations are first filtered through the LSTM before being passed to both the discriminator and the policy network. Accordingly, the gradient penalty term is computed on the LSTM filtered variables rather than on the original observations. The gradient penalty coefficient is set to $w_{gp} = 1$ and $\kappa = 0$ as well. Action

Environment	Gait	MSE	MAE	DTW
Short hamstring	Pathological	0.935 ± 0.295	1.934 ± 0.338	0.530 ± 0.015
	Corrected	0.983 ± 0.031	2.046 ± 0.032	0.503 ± 0.016
Hamstring weakness	Pathological	0.889 ± 0.281	1.706 ± 0.321	0.358 ± 0.075
	Corrected	1.014 ± 0.063	1.856 ± 0.063	0.267 ± 0.025
Iliopsoas weakness	Pathological	0.897 ± 0.203	1.718 ± 0.224	0.352 ± 0.021
	Corrected	0.997 ± 0.053	1.840 ± 0.061	0.171 ± 0.048

Table 5.3: Comparisons of losses across different environments for the agent trained by VAIL. The results are collected after 2.0×10^7 training steps. The VAIL framework includes a gradient penalty.

chunking and ensemble techniques are also introduced. Figure 5.9 shows the trajectories of the corrected gait in an environment with shorter hamstring as an example. In summary, Figure 5.10 presents the DTW loss curves across all three environments and Table 5.4 provides a quantitative comparison of the losses. The results suggest that incorporating LSTM in the training procedure does not improve the performance of the unified policy across the different environments.

Environment	Gait	MSE	MAE	DTW
Short hamstring	Pathological	0.957 ± 0.284	1.958 ± 0.327	0.530 ± 0.016
	Corrected	0.580 ± 0.284	1.485 ± 0.297	0.533 ± 0.057
Hamstring weakness	Pathological	0.867 ± 0.290	1.674 ± 0.339	0.359 ± 0.082
	Corrected	0.925 ± 0.436	1.776 ± 0.403	0.516 ± 0.060
Iliopsoas weakness	Pathological	0.897 ± 0.203	1.718 ± 0.224	0.352 ± 0.021
	Corrected	1.080 ± 0.271	1.956 ± 0.242	0.629 ± 0.103

Table 5.4: Comparisons of losses across different environments for the agent trained by GAIL combined with LSTM. The results are collected after 2.0×10^7 training steps. The framework includes a gradient penalty.

Zero-shot

In addition to evaluations conducted on the training environments, assessments are also performed on previously unseen environments, categorized as zero-shot experiments. Four new impairments, each corresponding to a distinct environment, are introduced and detailed in Table 5.5.

Environment	Model setting
Short hamstring 0.9	90% original optimal CE length of hamstring MTU
Hamstring weakness 0.4	40% original max. isometric force of hamstring MTU
Hamstring weakness 0.8 and Short hamstring 0.9	80% original max. isometric force of hamstring MTU and 90% original optimal CE length of hamstring MTU
Short Iliopsoas 0.7	70% original optimal CE length of hamstring MTU

Table 5.5: Impairments in zero-shot environments used for training.

The four environments are categorized into three types. Both `shortham0.9` and `hamwk0.4` represent the same impairments as those used during training but with different severity levels. The environment `hamwk0.8&shortham0.9` combines impairments of shorter hamstring

and weaker hamstring though with a less severe condition than those in training. While this combined impairment may be less representative from a biomechanical perspective, it remains valuable from an engineering standpoint, as it combines features of both original impairments. The `iliowk0.7` environment introduces a completely novel and unseen impairment. Figure 5.11 presents the DTW loss curves for all four environments, none of which is included in the training set, and demonstrates that the policy trained with the help of VDB maintains a good performance on these unseen cases. These findings are further supported by the data summarized in Table 5.6. The initial low value is due to the model consistently falling within the first few time steps when the policy has not been sufficiently trained, which undermines the reliability of the DTW metric. Additionally, the results for the environment `hamwk0.8&shortham0.9` are not representative, as the policy fails to support the model in walking forward for a sustained duration.

Environment	Gait	MSE	MAE	DTW
Short hamstring 0.9	Pathological	0.967 ± 0.104	1.917 ± 0.139	0.258 ± 0.004
	Corrected	1.004 ± 0.056	1.887 ± 0.054	0.239 ± 0.037
Hamstring weakness 0.4	Pathological	1.083 ± 0.045	1.942 ± 0.054	0.334 ± 0.005
	Corrected	0.971 ± 0.186	1.841 ± 0.237	0.324 ± 0.047
Hamstring weakness 0.8 and Short hamstring 0.9	Pathological	0.230 ± 0.126	0.854 ± 0.315	0.273 ± 0.082
	Corrected	0.490 ± 0.193	1.321 ± 0.265	0.503 ± 0.114
Short iliopsoas 0.7	Pathological	0.981 ± 0.044	1.860 ± 0.065	0.252 ± 0.002
	Corrected	0.914 ± 0.070	1.808 ± 0.085	0.241 ± 0.024

Table 5.6: Comparisons of losses across zero-shot environments for the agent trained by VAIL. The results are collected after 2.0×10^7 training steps. The VAIL framework includes a gradient penalty.

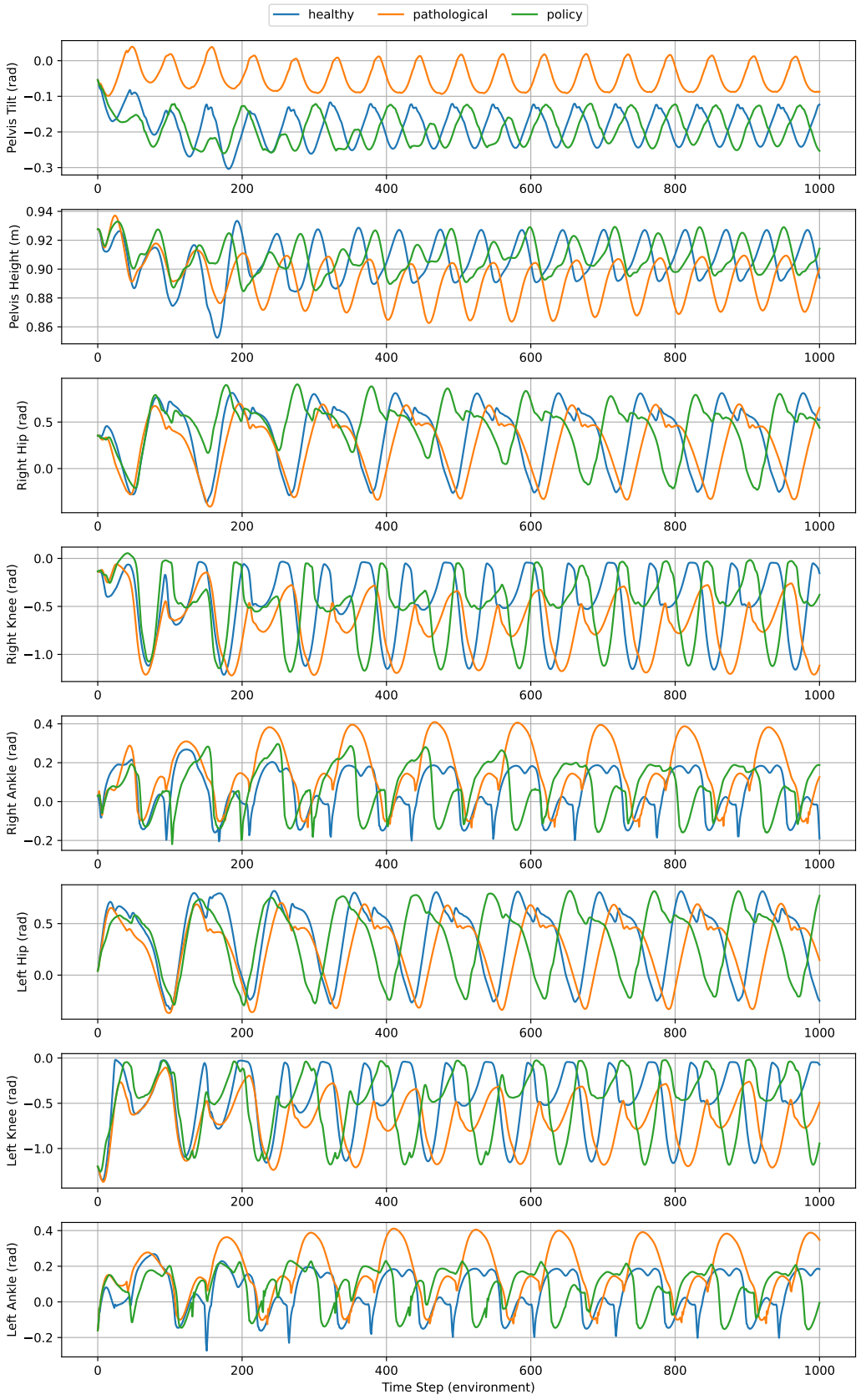

Figure 5.3: Trajectories of various DoF at training step 1.8×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with shortened hamstring.

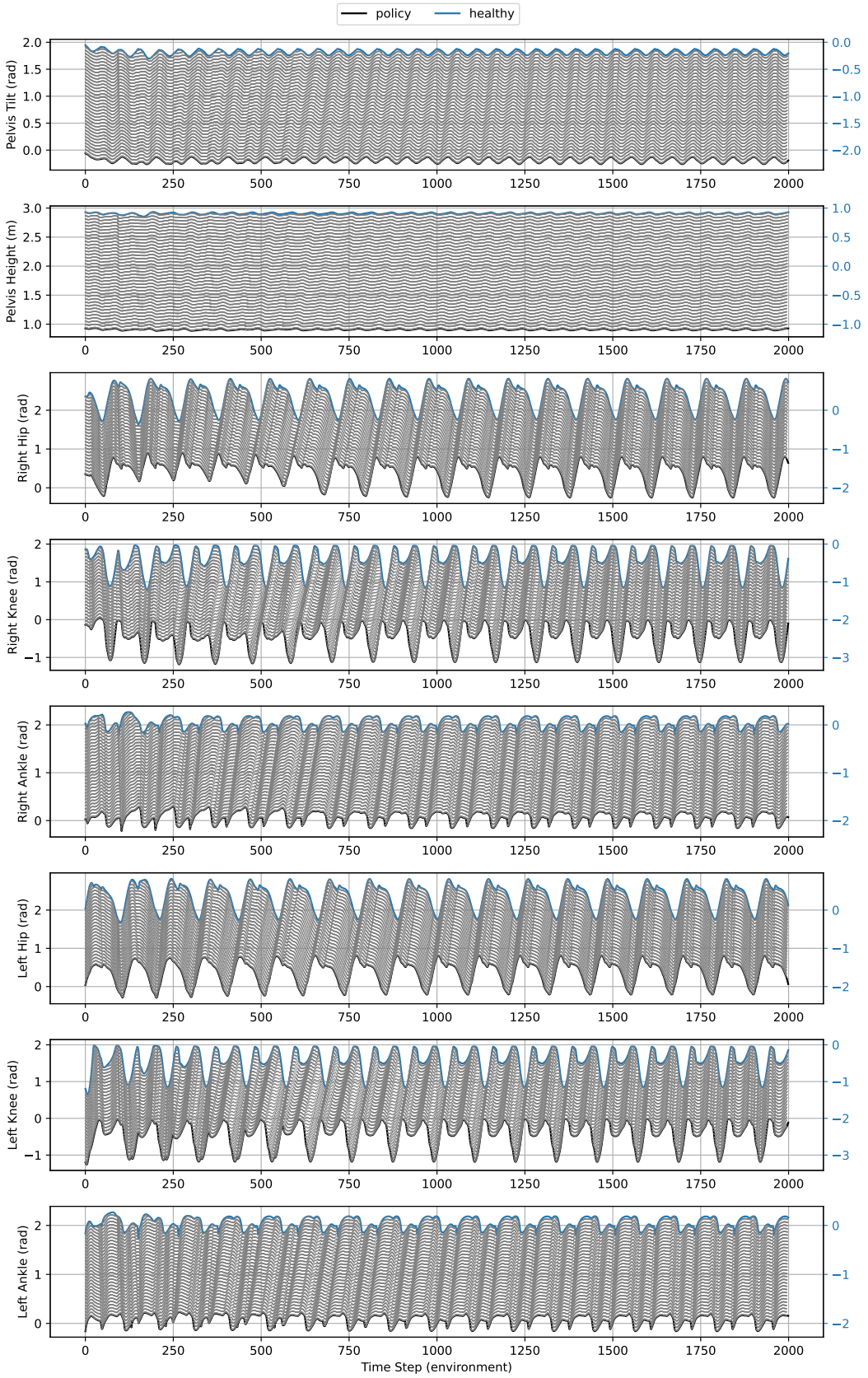

Figure 5.4: DTW matching results at training step 1.8×10^7: healthy gait (blue), corrected gait (black) and matching pairs (grey). The model has a shortened hamstring impairment. Healthy gait trajectories are shifted by two units for clarity.

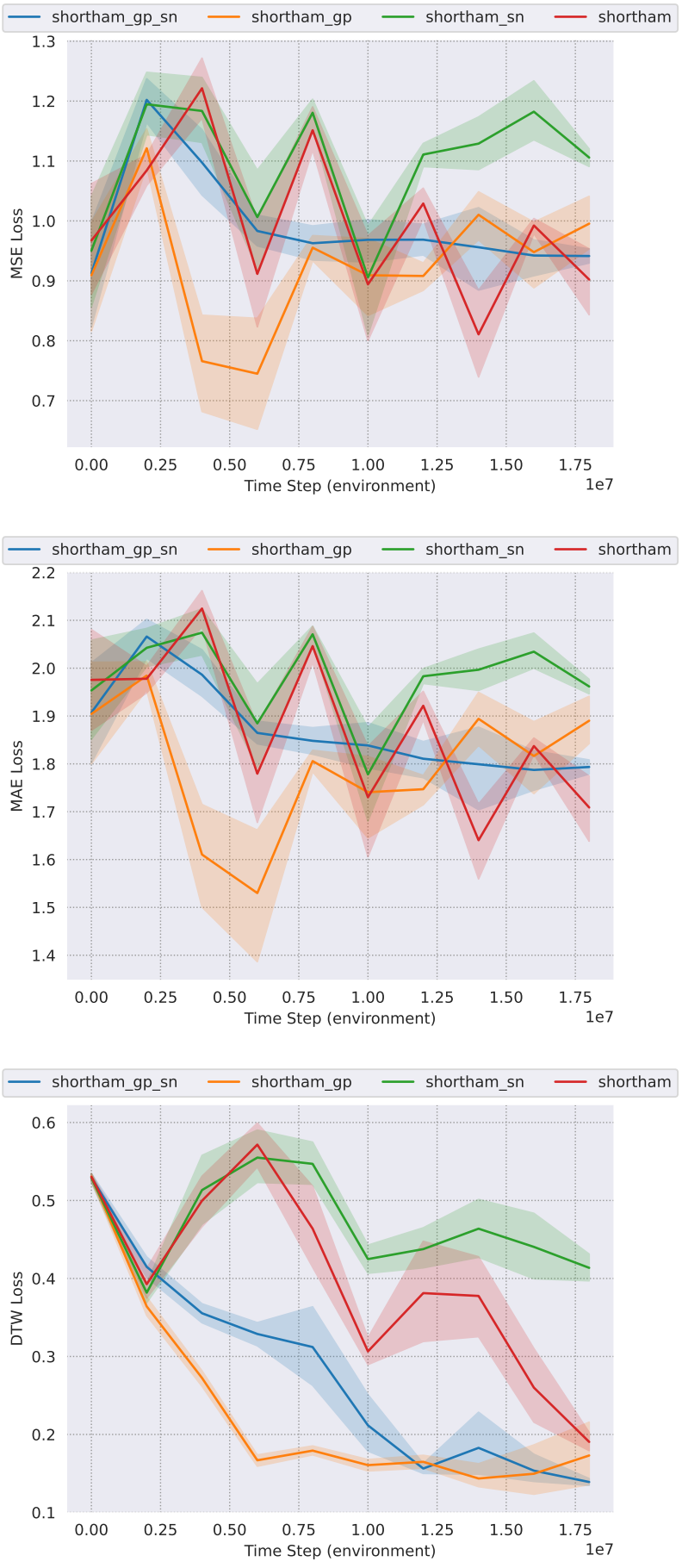

Figure 5.5: Training loss curves of the model with shortened hamstring impairment.

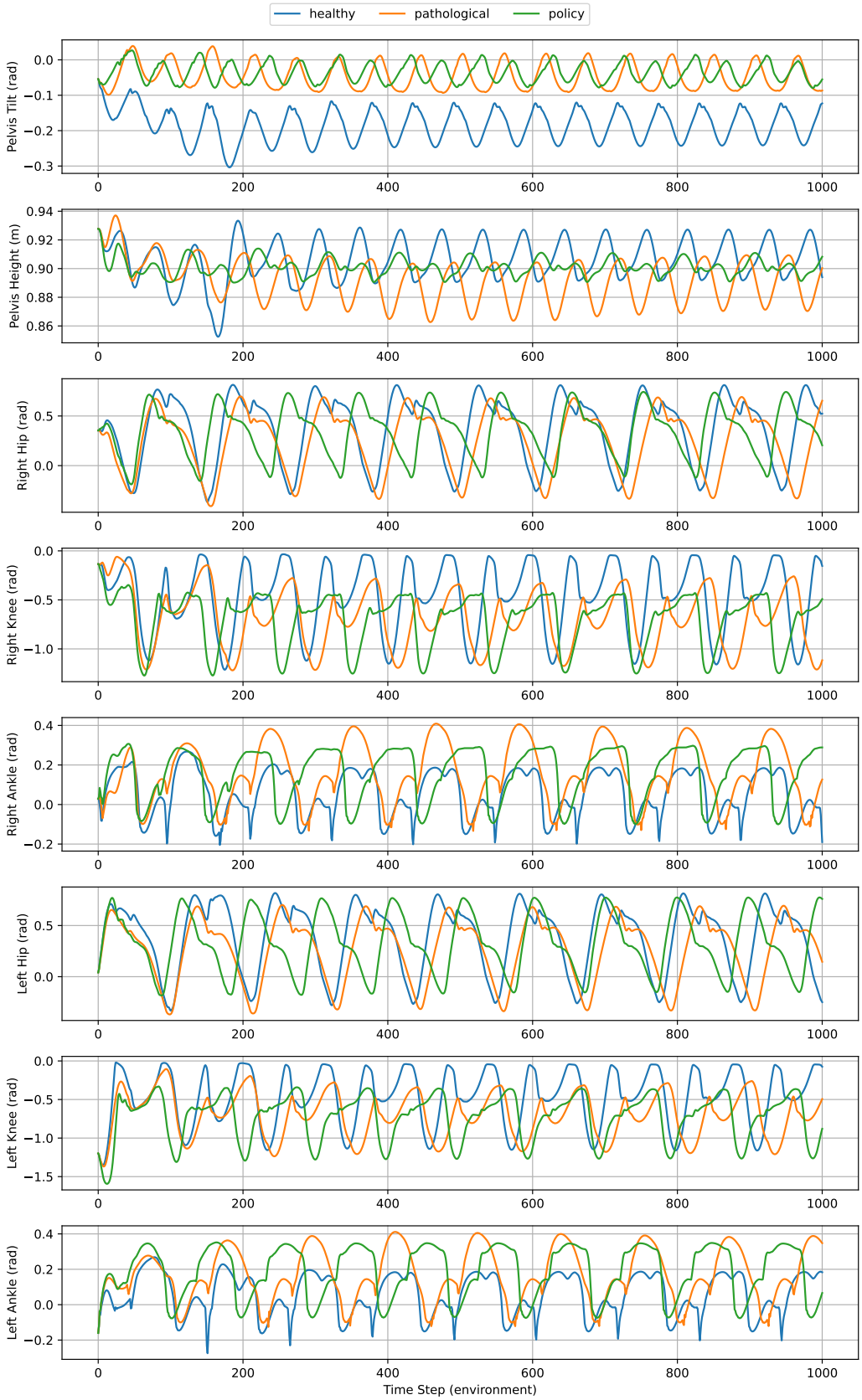

Figure 5.6: Trajectories of various DoF at training step 2.0×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with shortened hamstring and is trained by VAIL with gradient penalty.

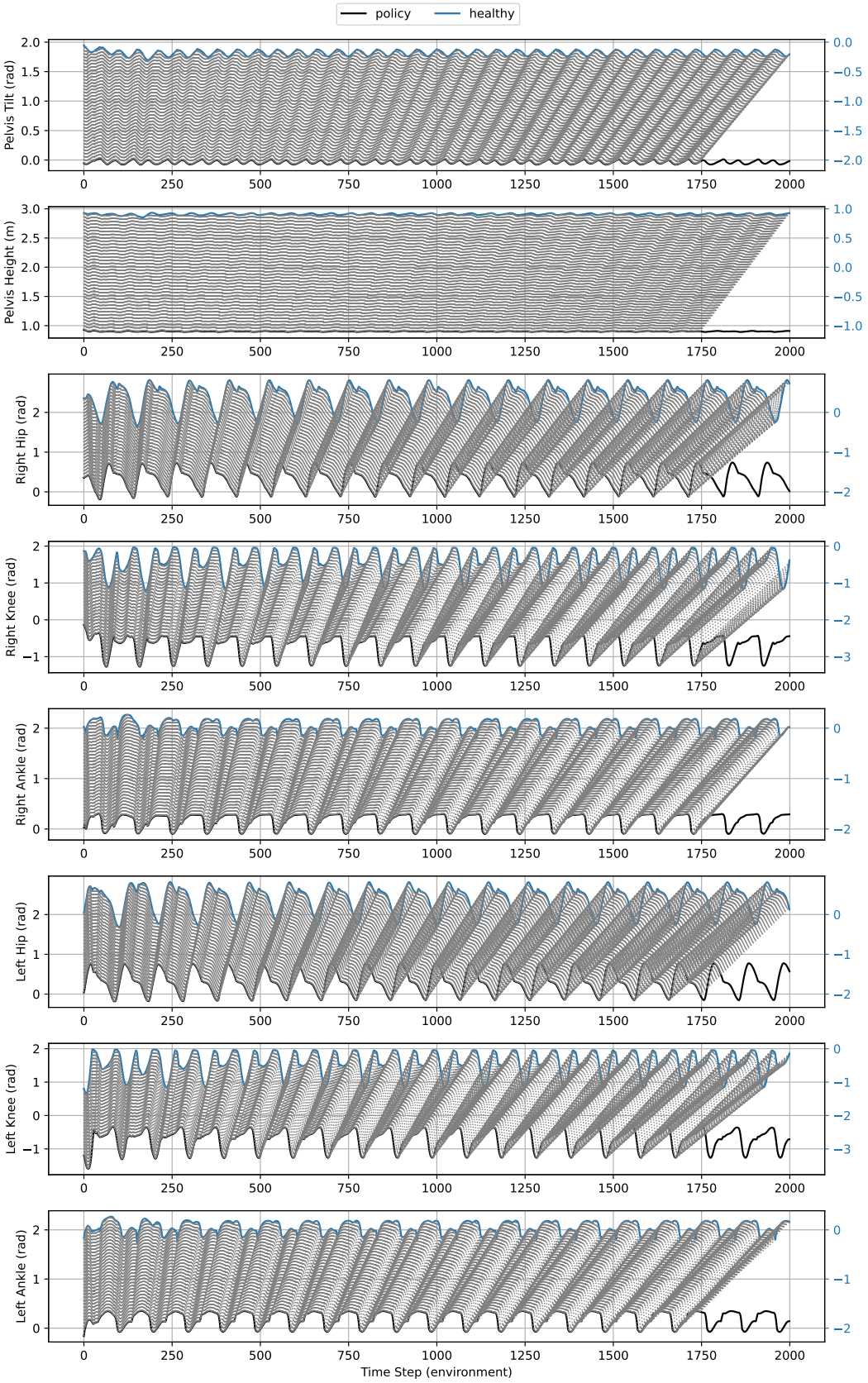

Figure 5.7: DTW matching results at training step 2.0×10^7: healthy gait (blue), corrected gait (black) and matching pairs (grey). The model has a shortened hamstring impairment and is trained by VAIL with gradient penalty. Healthy gait trajectories are shifted by two units for clarity.

Figure 5.8: Training loss curves of the model with different impairments. The model is trained by VAIL with gradient penalty.

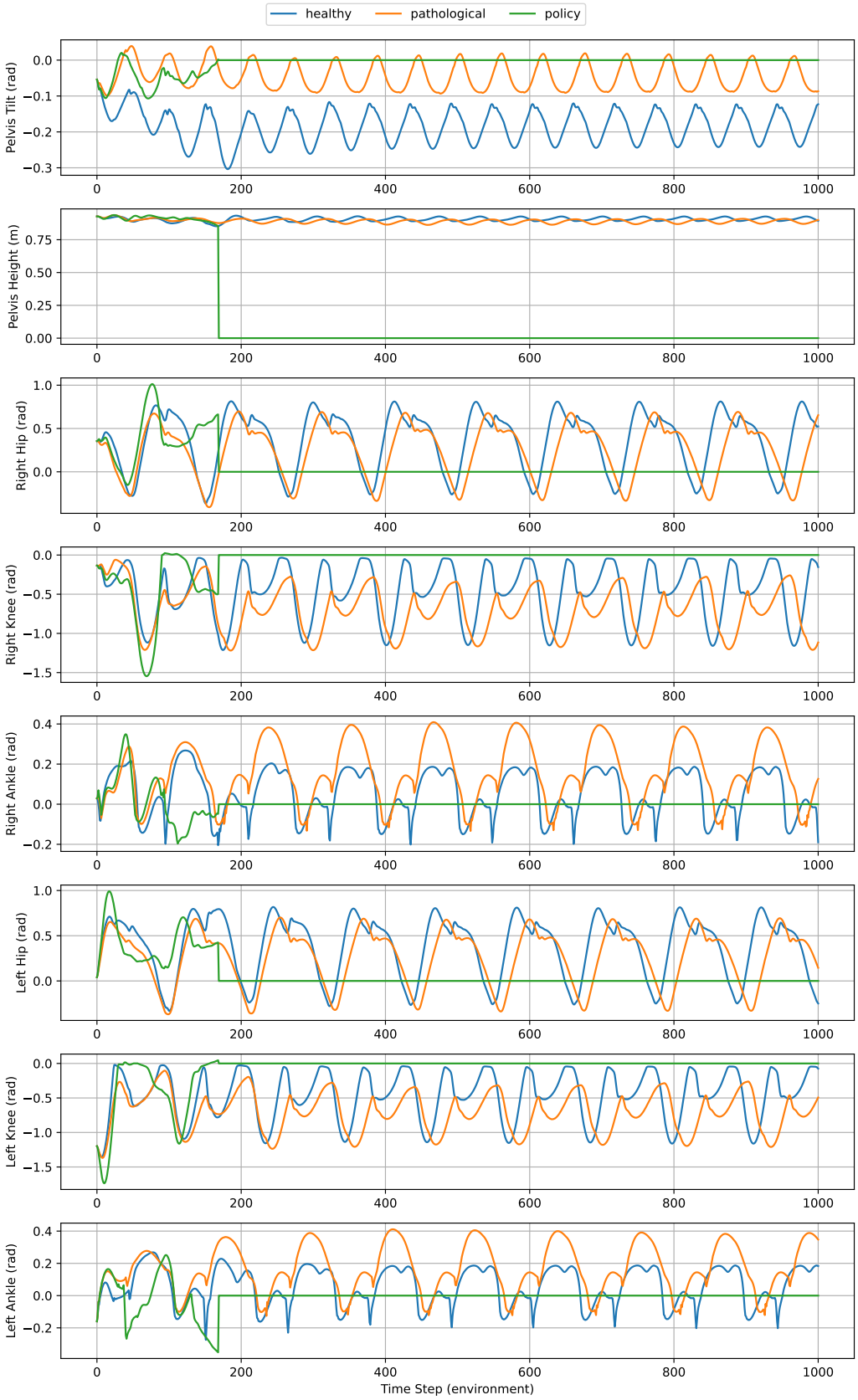

Figure 5.9: Trajectories of various DoF at training step 2.0×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with shortened hamstring and is trained by GAIL combined with LSTM and gradient penalty.

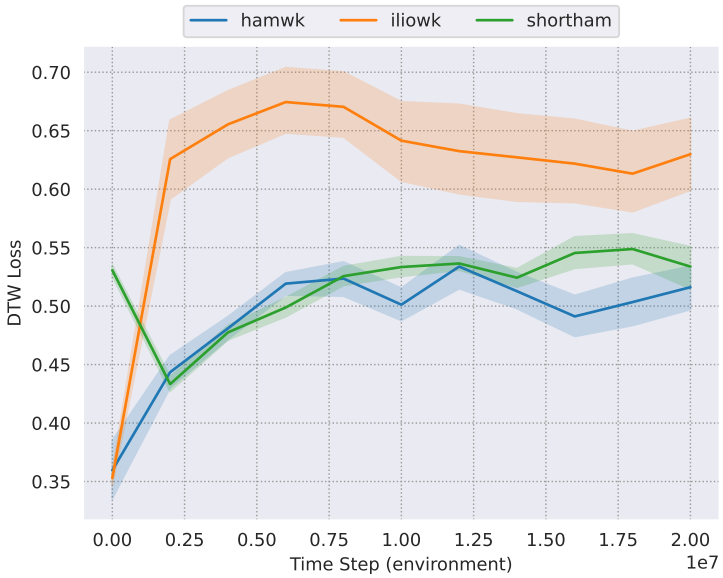

Figure 5.10: Training loss curves of the model with different impairments. The model is trained by GAIL combined with LSTM and gradient penalty.

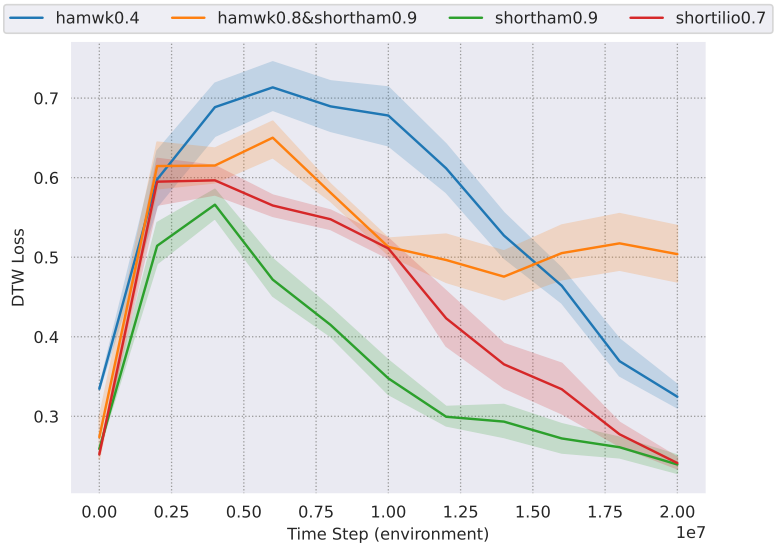

Figure 5.11: Training loss curves of the model with zero-shot impairments. The model is trained by VAIL. Note: The low values at the beginning of training are due to the randomly initialized policy failing quickly.

Chapter 6

Conclusion

This thesis demonstrates the feasibility of using IRL to train a high-level control policy for a lower limb exoskeleton. The goal of this approach is to assist patients with various lower limb impairments in walking as closely as possible to healthy individuals. To make the simulation more realistic, a musculoskeletal humanoid model is first constructed. Unlike models that rely on joint mounted motors, this model is actuated by multiple MTUs, which mimic human muscle function. The overall control architecture consists of two levels: The low-level controller simulates reflex driven muscle activation, representing the spinal level control of movement. Building upon this reflex controlled model, the high-level controller is developed to govern the exoskeleton at four key joints. This layered control structure enables coordinated and adaptive assistance during locomotion.

The high-level control strategy is based on IRL, which eliminates the effort for manual reward engineering that typically required in RL. Unlike conventional IRL methods that try to recover the underlying reward function analytically, GAIL seeks to directly learn the expert policy from demonstration data. In this framework, a discriminator is trained to distinguish between the observations from the agent and those from the expert. This setup shares conceptual similarities with GAN and the output of the discriminator is used as a reward signal for training the policy network.

The experimental evaluation consists of two parts. First, the proposed method is tested independently across three different environments. Subsequently, a unified policy is trained and evaluated across all environments to assess its generalizability. To validate the scalability and robustness of the unified policy, it is tested on four more previously unseen environments. The evaluation metric is based on a modified DTW loss, accounting for variations in trajectory frequency and phase. This adjustment ensures a more reliable comparison between generated and reference trajectories. The consistent decrease in DTW loss across nearly all test cases supports the motivation for using IRL to train high-level control strategies for exoskeletons.

The results in this thesis confirm the feasibility of this method in addressing a range of gait impairments. However, the experimental setup involving multiple environments could benefit from further analysis. In the current implementation, hyperparameters for both VAIL and the LSTM modules were selected somewhat heuristically. The experiments in Chapter 5 demonstrate feasibility using these fixed hyperparameters, but do not explore how performance may vary with different configurations.

Future work may explore the use of more sample-efficient IRL algorithms, such as Least-Squares Inverse Q-Learning [Al-+23], to improve learning efficiency. To better simulate real-world applications, future studies could investigate scenarios involving kinematic misalignments between exoskeleton actuators and human joints. Moreover, learning the gain and offset parameters of reflex controllers through alternative methods such as RL could offer a more realistic representation of practical deployment, where control strategies may

need to be individually adapted for different users.

Additionally, to better simulate real-world human motion, it would be valuable to study the current model in a 3-dimensional setup. This includes incorporating additional degrees of freedom, such as hip adduction and abduction, as well as rotational movements at the hips, knees and ankles in the frontal plane. Expanding the model in this way would enable a more comprehensive representation of human biomechanics and allow the control strategy to address a wider range of motion patterns and gait abnormalities.

Appendix A

Appendix 1

A.1 Single environment: Weaker hamstring

Algorithm	Gait	MSE	MAE	DTW
Vanilla	Pathological	0.861 ± 0.301	1.669 ± 0.348	0.362 ± 0.082
	Corrected	0.885 ± 0.049	1.763 ± 0.042	0.288 ± 0.051
With spectral norm	Pathological	0.898 ± 0.271	1.707 ± 0.320	0.360 ± 0.087
	Corrected	0.857 ± 0.189	1.704 ± 0.205	0.300 ± 0.096
With gradient penalty	Pathological	0.874 ± 0.294	1.685 ± 0.345	0.348 ± 0.072
	Corrected	0.862 ± 0.052	1.714 ± 0.049	0.213 ± 0.092
With spectral norm and gradient penalty	Pathological	0.853 ± 0.300	1.660 ± 0.349	0.357 ± 0.079
	Corrected	0.851 ± 0.055	1.702 ± 0.065	0.215 ± 0.083

Table A.1: Quantified comparisons of different training techniques on the corrected gait at training step 1.8×10^7. The gait has the impairment with weakened hamstring.

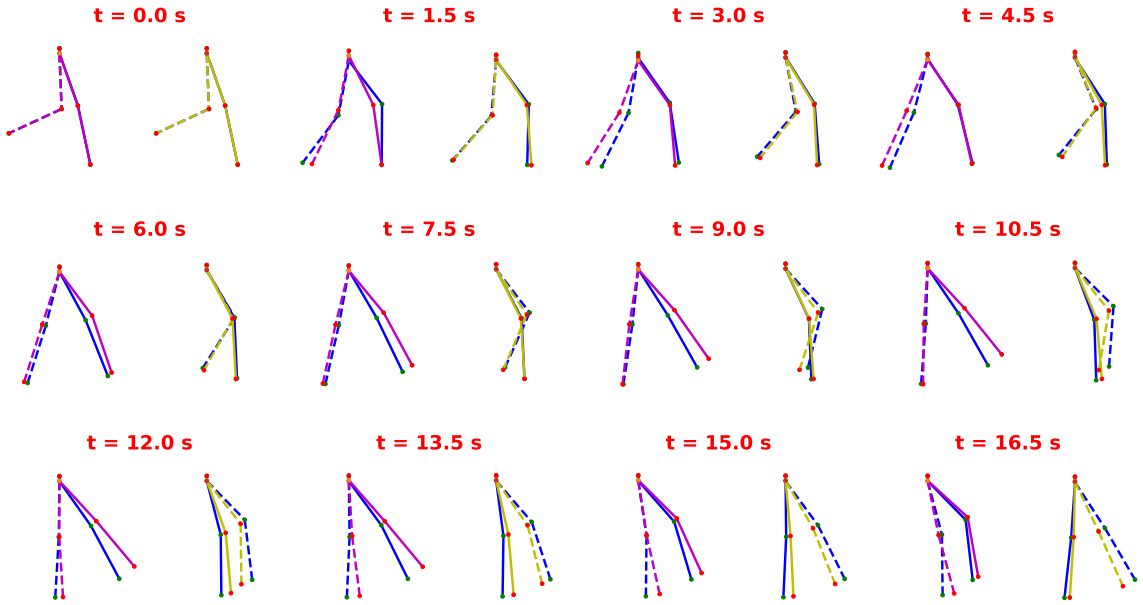

Figure A.1: Overlaid stick figures comparing gaits at different time steps along the trajectory: healthy gait (blue), pathological gait (purple) and corrected gait (yellow). The left panel compares pathological vs. healthy, the right panel compares corrected vs. healthy. The gait has the impairment with weakened hamstring. Both are aligned using DTW.

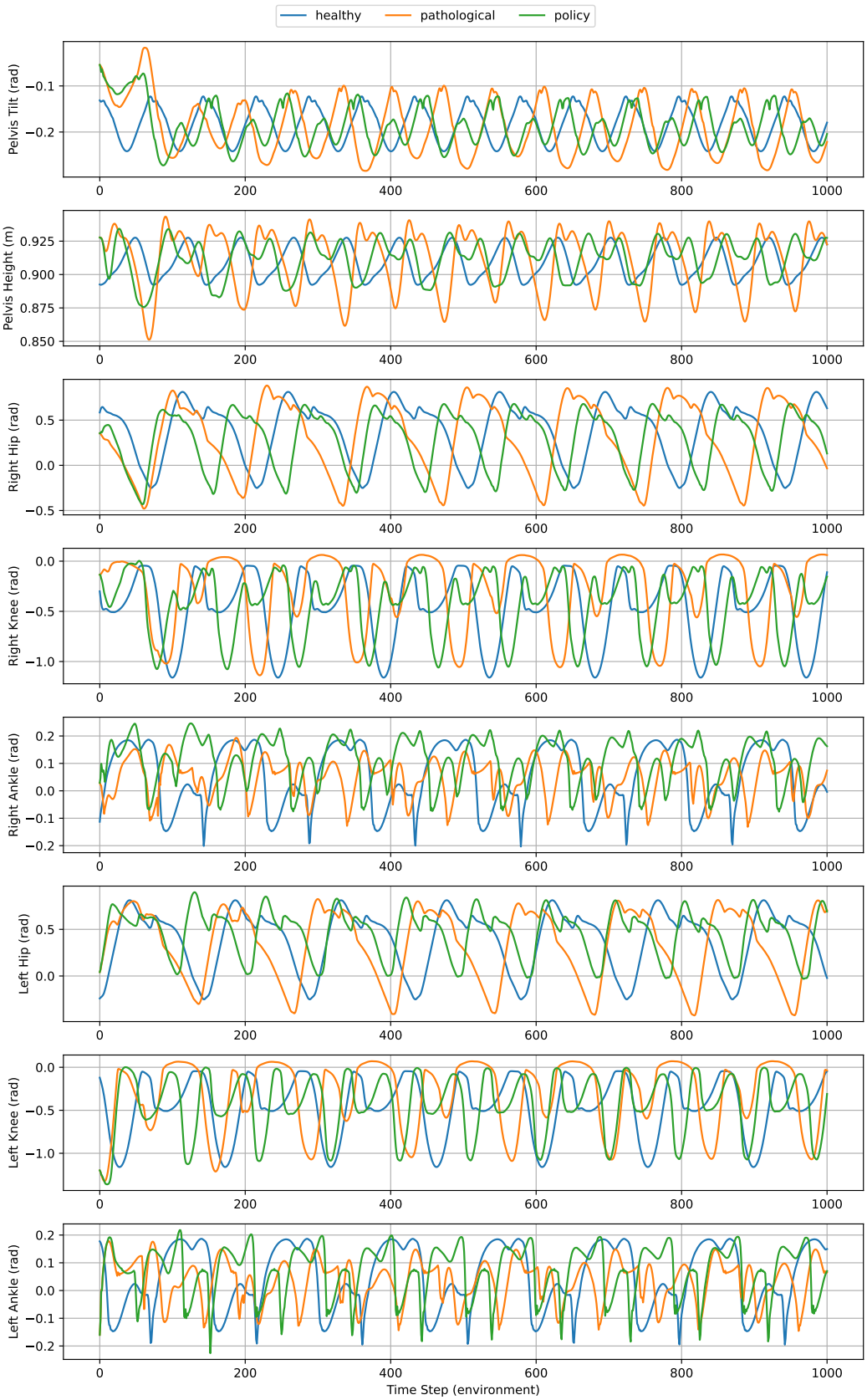

Figure A.2: Trajectories of various DoF at training step 1.8×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with weakened hamstring.

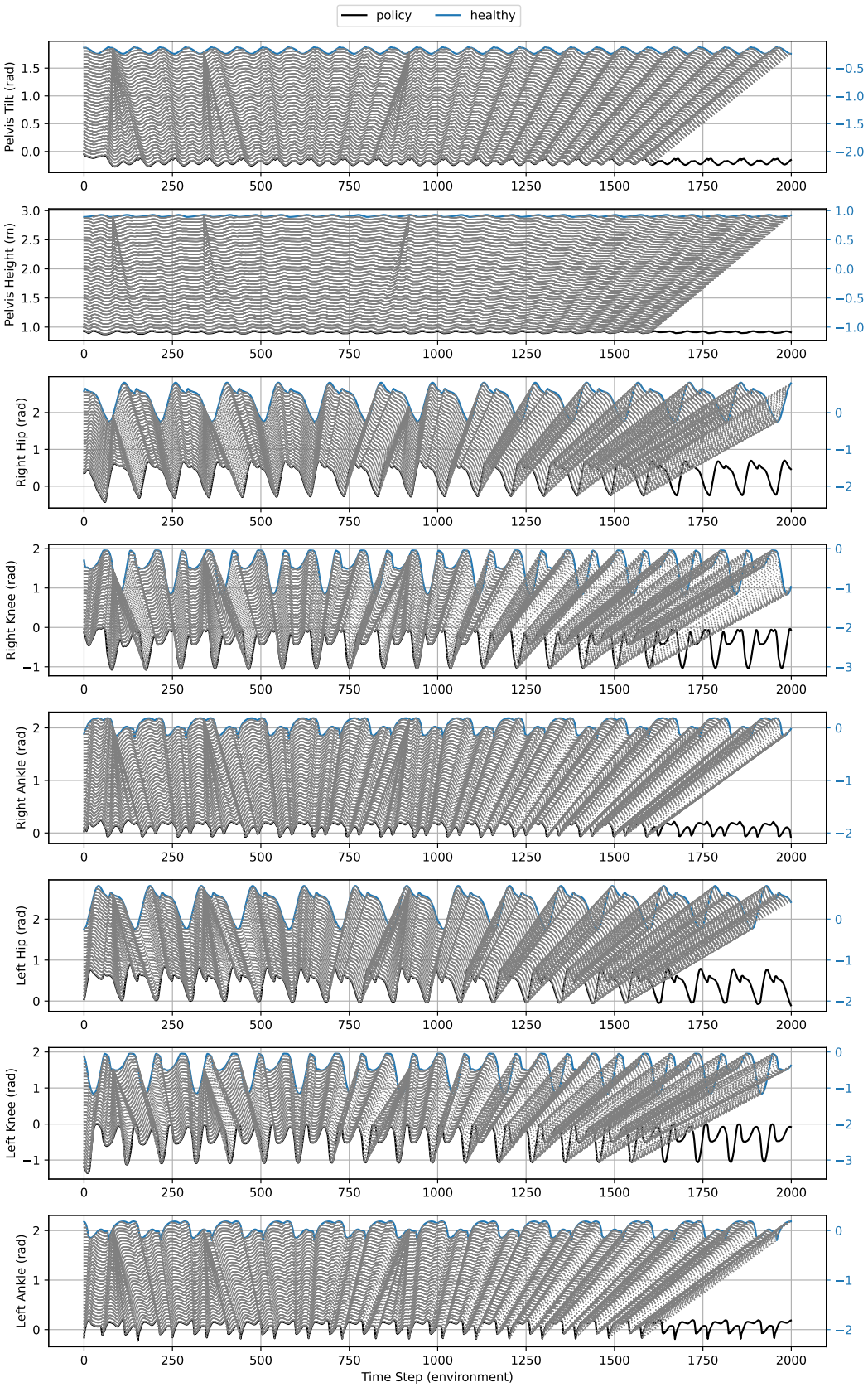

Figure A.3: DTW matching results at training step 1.8×10^7: healthy gait (blue), corrected gait (black) and matching pairs (grey). The model has a weakened hamstring impairment. Healthy gait trajectories are shifted by two units for clarity.

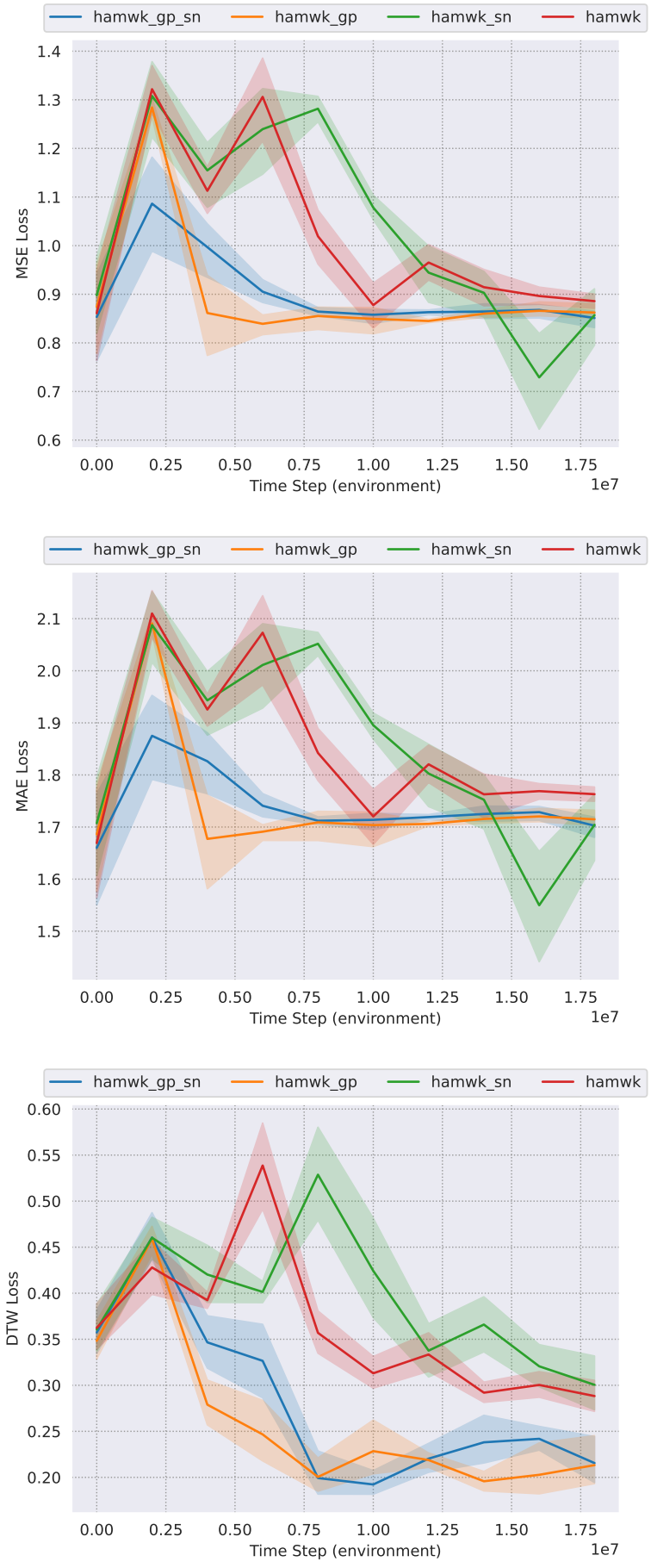

Figure A.4: Training loss curves of the model with weakened hamstring impairment.

A.2 Single environment: Weaker iliopsoas

Algorithm	Gait	MSE	MAE	DTW
Vanilla	Pathological	0.897 ± 0.203	1.718 ± 0.224	0.352 ± 0.021
	Corrected	1.019 ± 0.116	1.915 ± 0.154	0.248 ± 0.126
With spectral norm	Pathological	0.897 ± 0.203	1.718 ± 0.224	0.352 ± 0.021
	Corrected	0.891 ± 0.047	1.776 ± 0.057	0.297 ± 0.062
With gradient penalty	Pathological	0.897 ± 0.203	1.718 ± 0.224	0.352 ± 0.021
	Corrected	0.840 ± 0.348	1.661 ± 0.316	0.356 ± 0.253
With spectral norm and gradient penalty	Pathological	0.897 ± 0.203	1.718 ± 0.224	0.352 ± 0.021
	Corrected	0.844 ± 0.146	1.710 ± 0.142	0.434 ± 0.246

Table A.2: Quantified comparisons of different training techniques on the corrected gait at training step 1.8×10^7. The gait has the impairment with weakened iliopsoas.

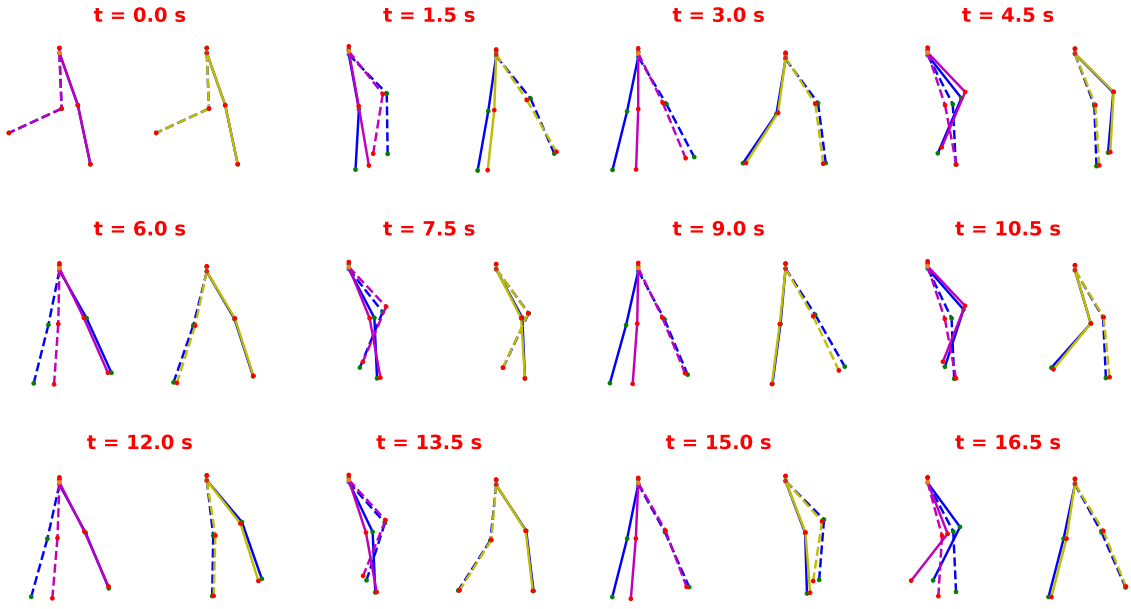

Figure A.5: Overlaid stick figures comparing gaits at different time steps along the trajectory: healthy gait (blue), pathological gait (purple) and corrected gait (yellow). The left panel compares pathological vs. healthy, the right panel compares corrected vs. healthy. The gait has the impairment with weakened iliopsoas. Both are aligned using DTW.

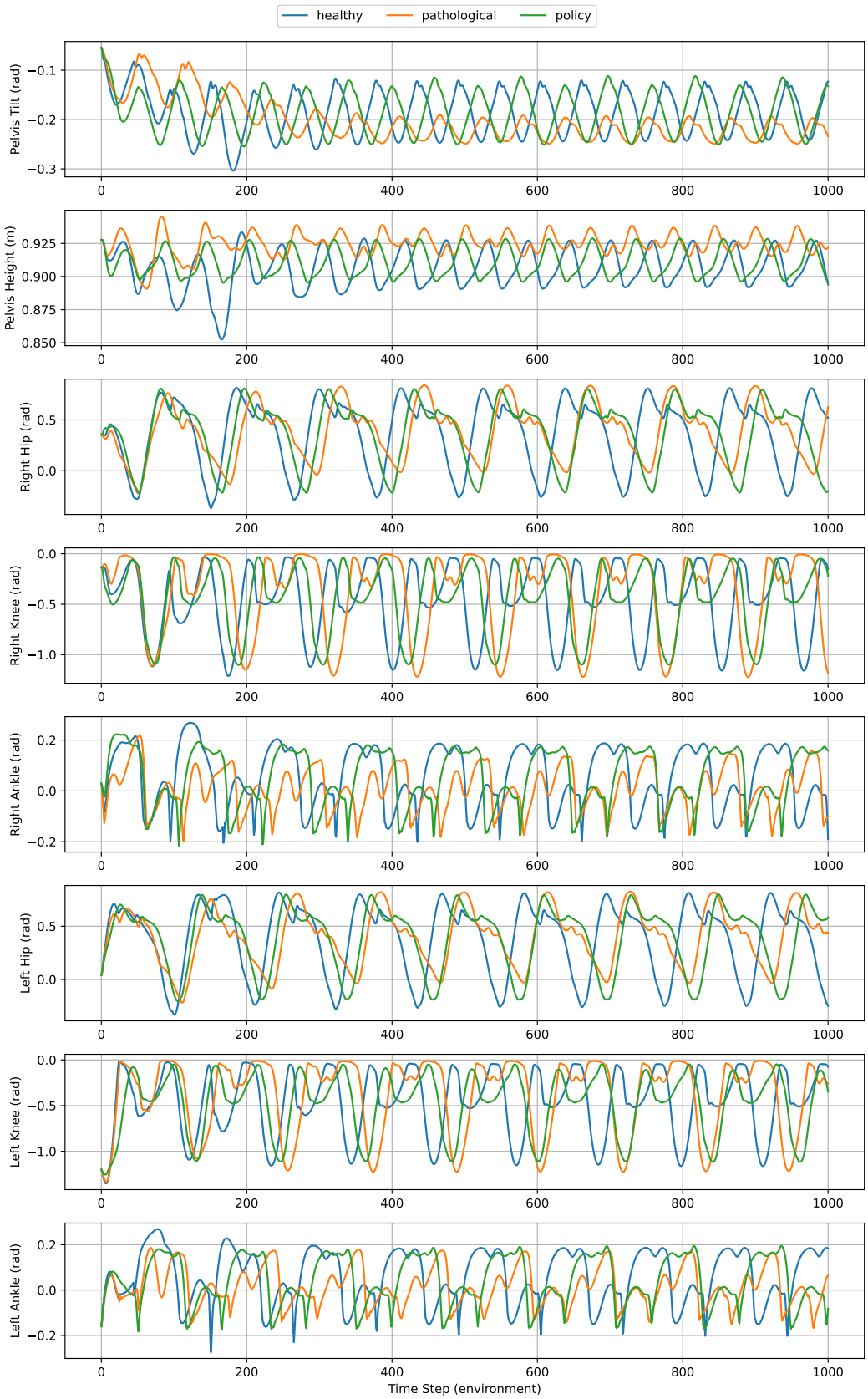

Figure A.6: Trajectories of various DoF at training step 1.8×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with weakened iliopsoas.

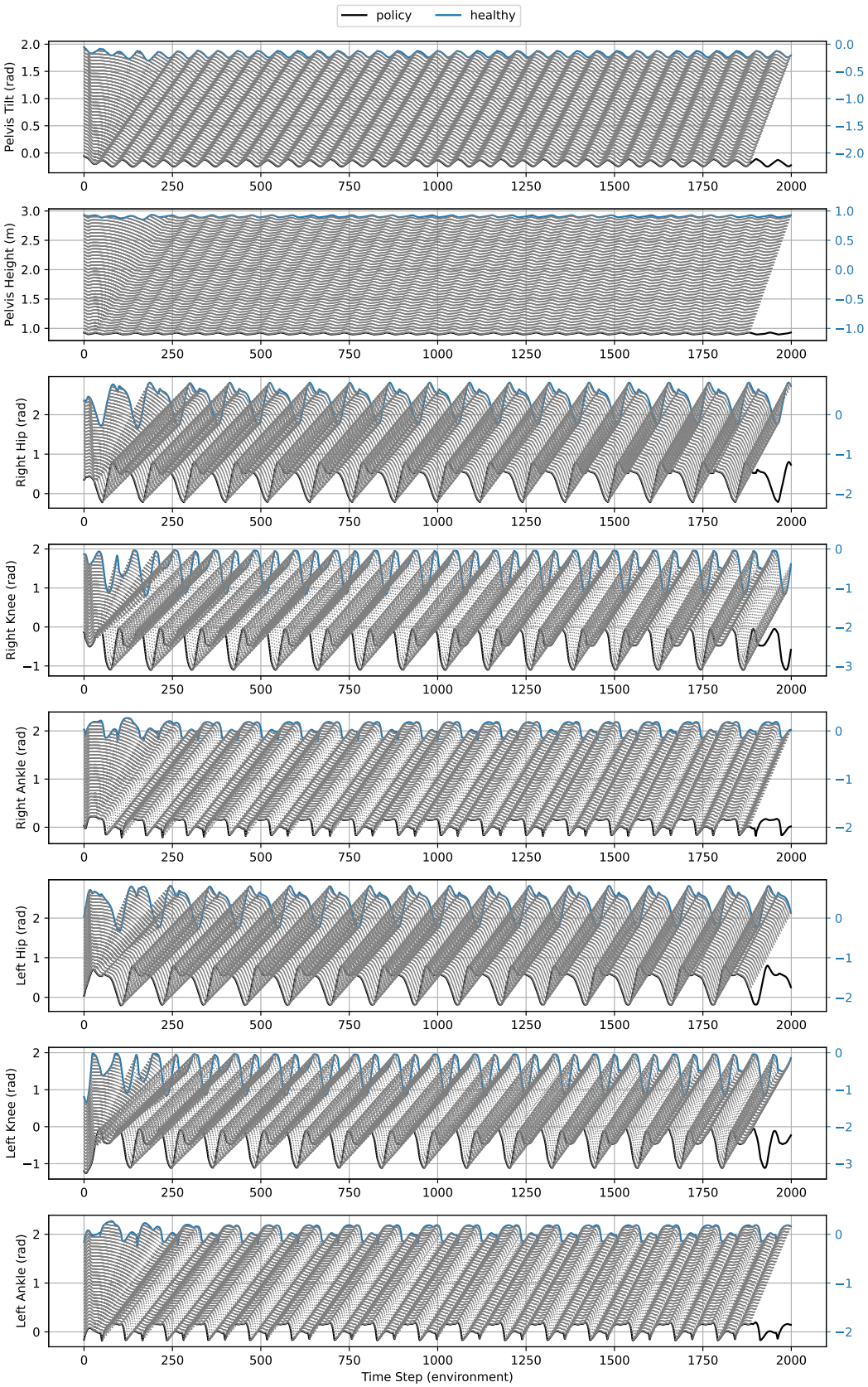

Figure A.7: DTW matching results at training step 1.8×10^7: healthy gait (blue), corrected gait (black) and matching pairs (grey). The model has a weakened iliopsoas impairment. Healthy gait trajectories are shifted by two units for clarity.

Figure A.8: Training loss curves of the model with weakened iliopsoas impairment.

A.3 Multiple environments with VDB: Weaker hamstring

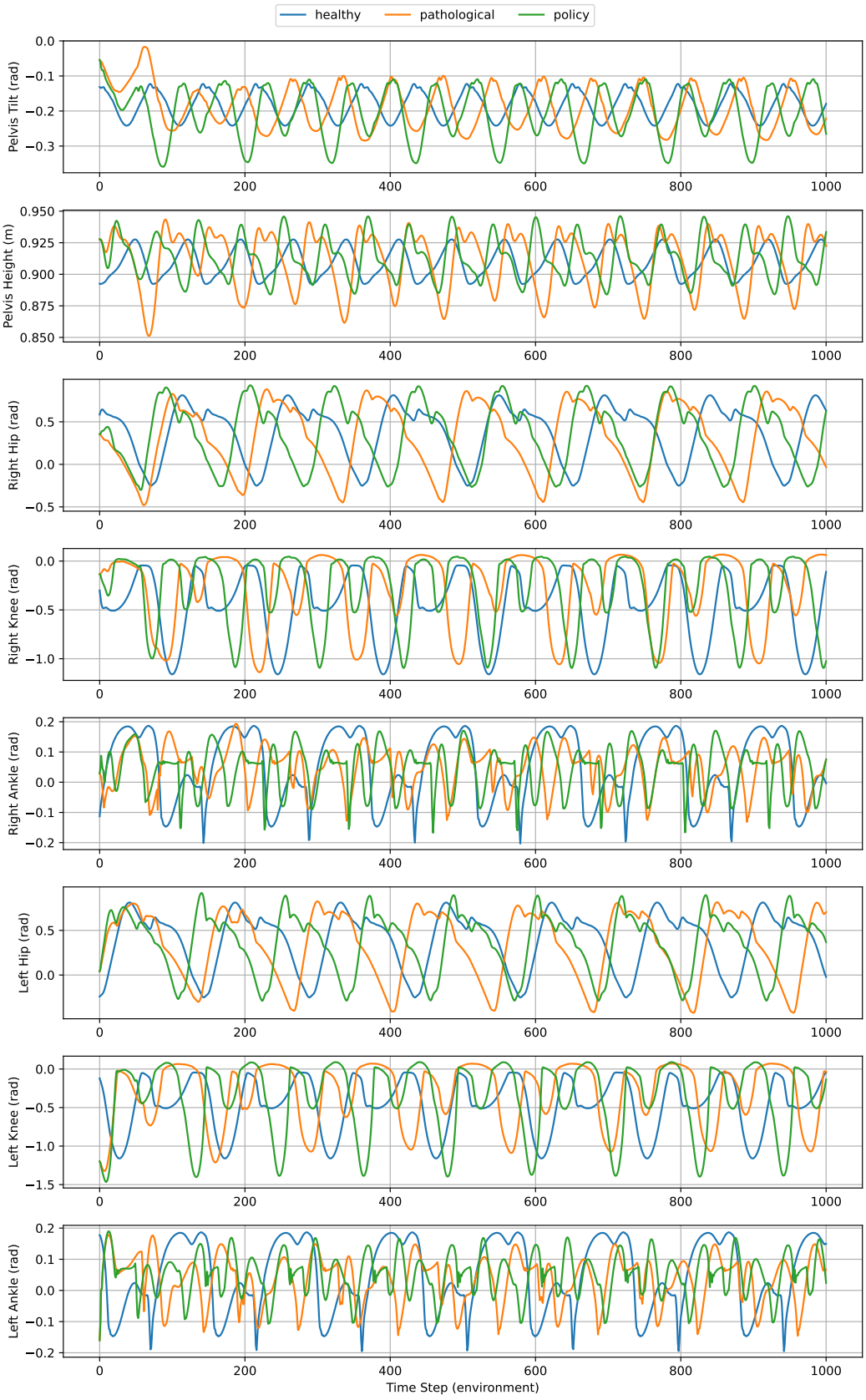

Figure A.9: Trajectories of various DoF at training step 2.0×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with weakened hamstring and is trained by VAIL with gradient penalty.

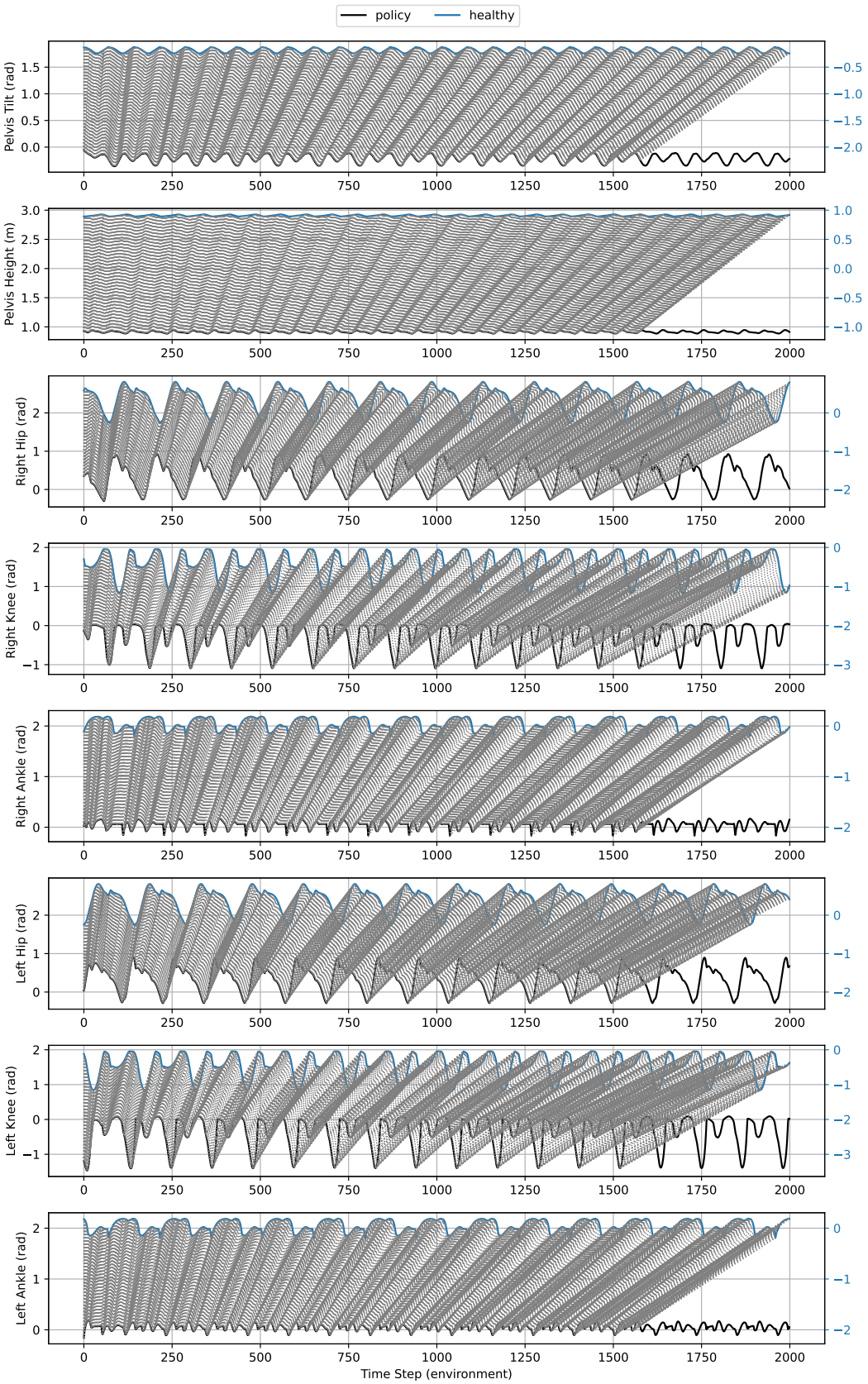

Figure A.10: DTW matching results at training step 2.0×10^7: healthy gait (blue), corrected gait (black) and matching pairs (grey). The model has a weakened hamstring impairment and is trained by VAIL with gradient penalty. Healthy gait trajectories are shifted by two units for clarity.

A.4 Multiple environments with VDB: Weaker iliopsoas

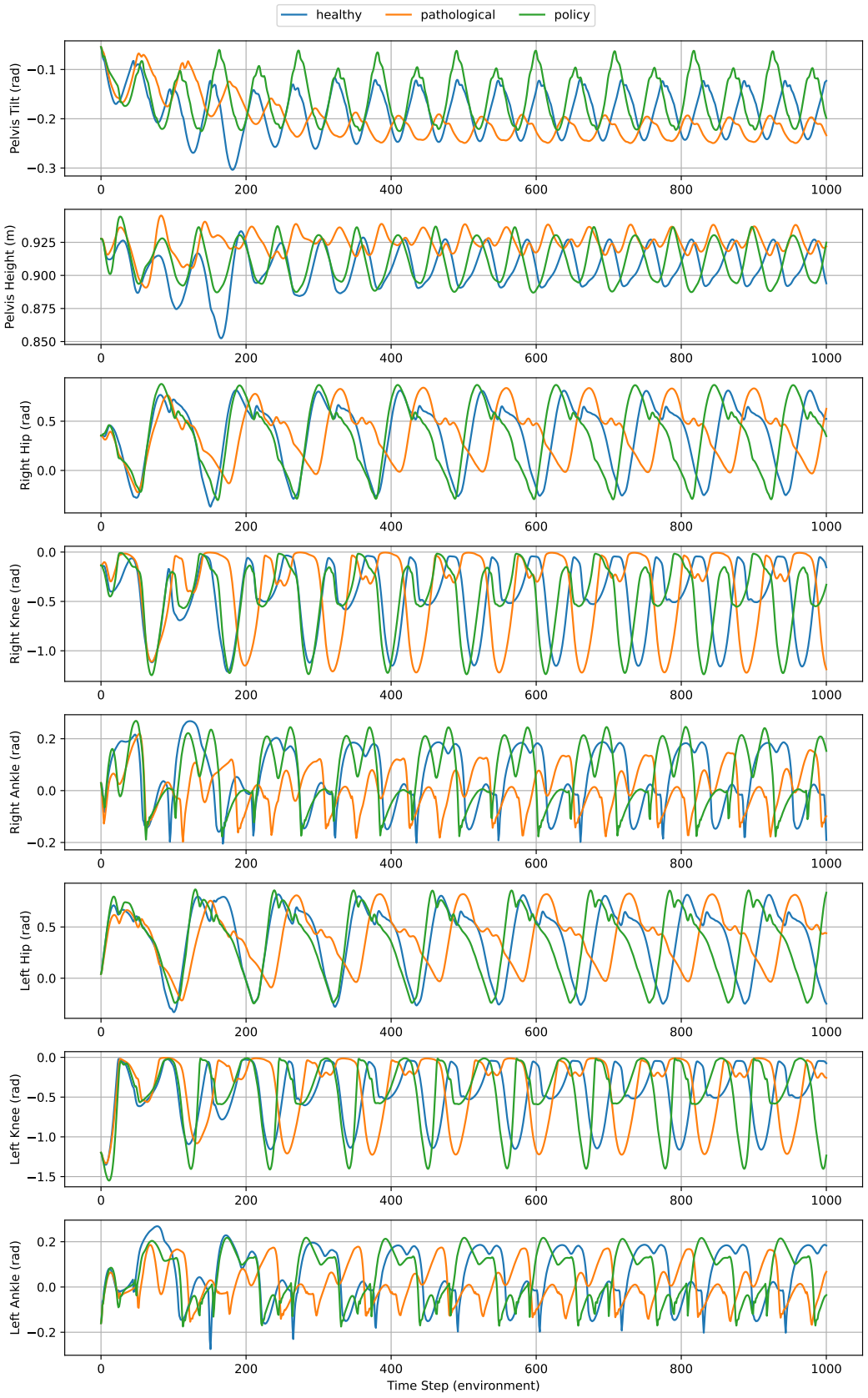

Figure A.11: Trajectories of various DoF at training step 2.0×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with weakened iliopsoas and is trained by VAIL with gradient penalty.

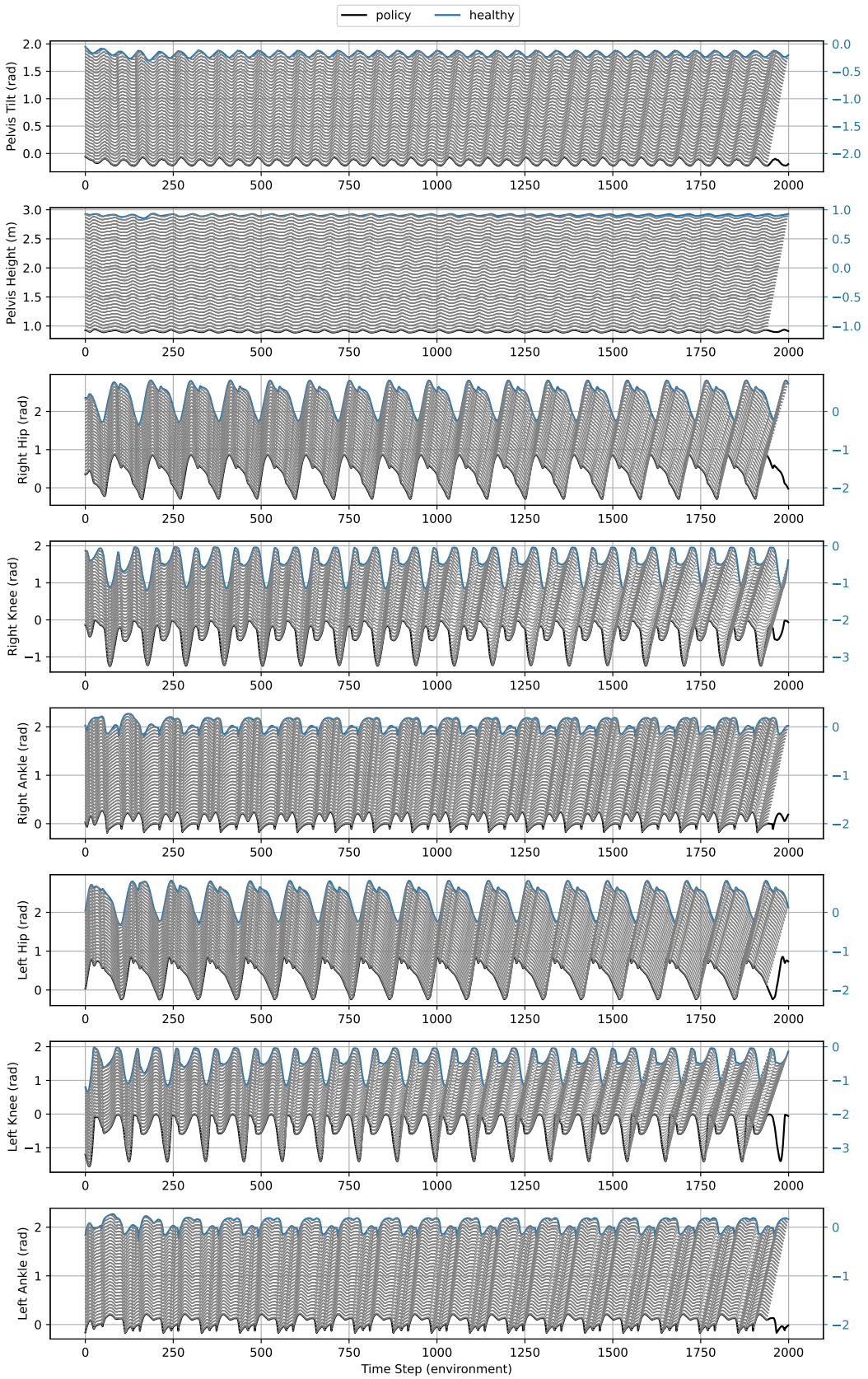

Figure A.12: DTW matching results at training step 2.0×10^7: healthy gait (blue), corrected gait (black) and matching pairs (grey). The model has a weakened iliopsoas impairment and is trained by VAIL with gradient penalty. Healthy gait trajectories are shifted by two units for clarity.

A.5 Multiple environments with LSTM: Weaker hamstring

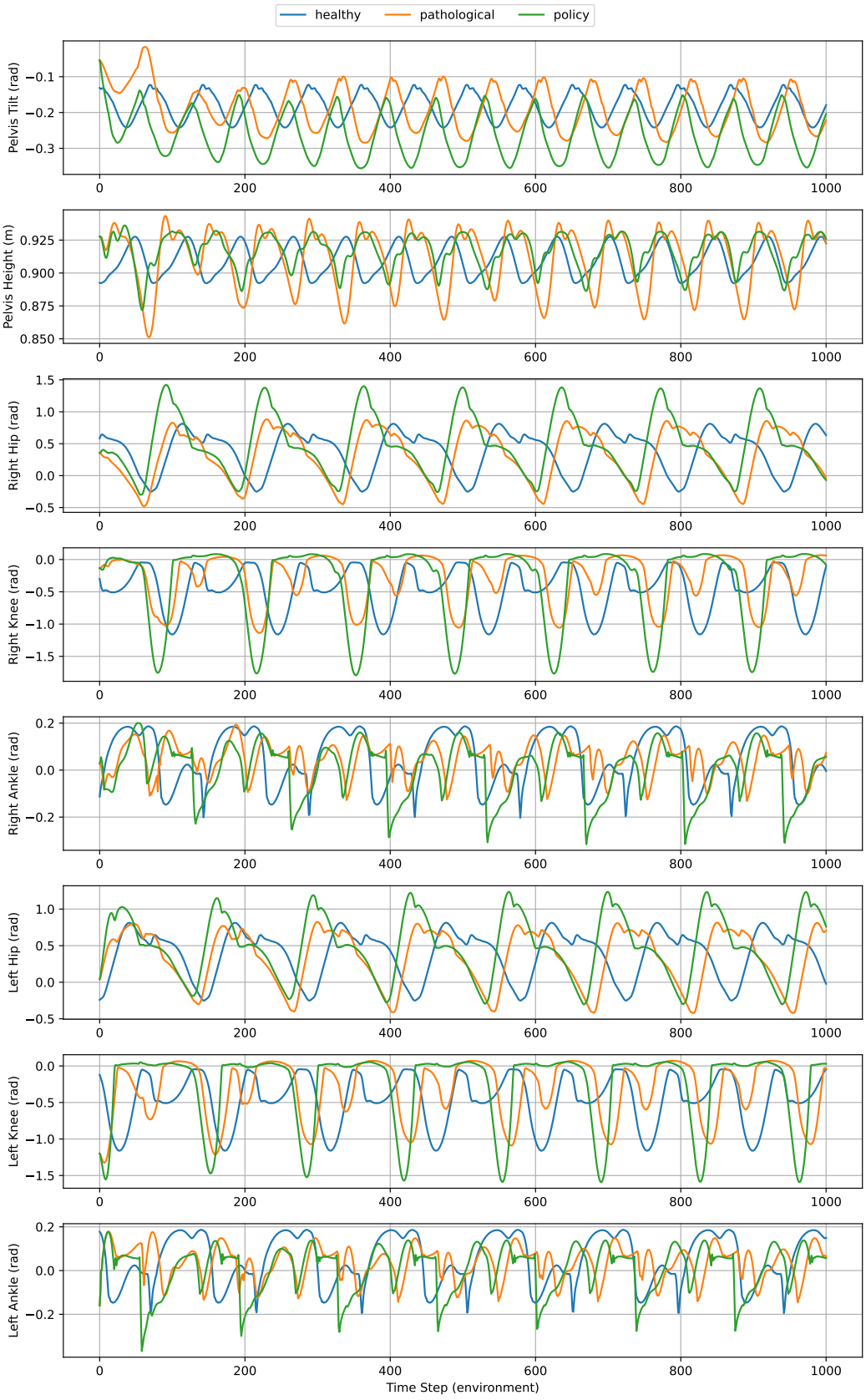

Figure A.13: Trajectories of various DoF at training step 2.0×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with weakened hamstring and is trained by GAIL combined with LSTM and gradient penalty.

A.6 Multiple environments with LSTM: Weaker iliopsoas

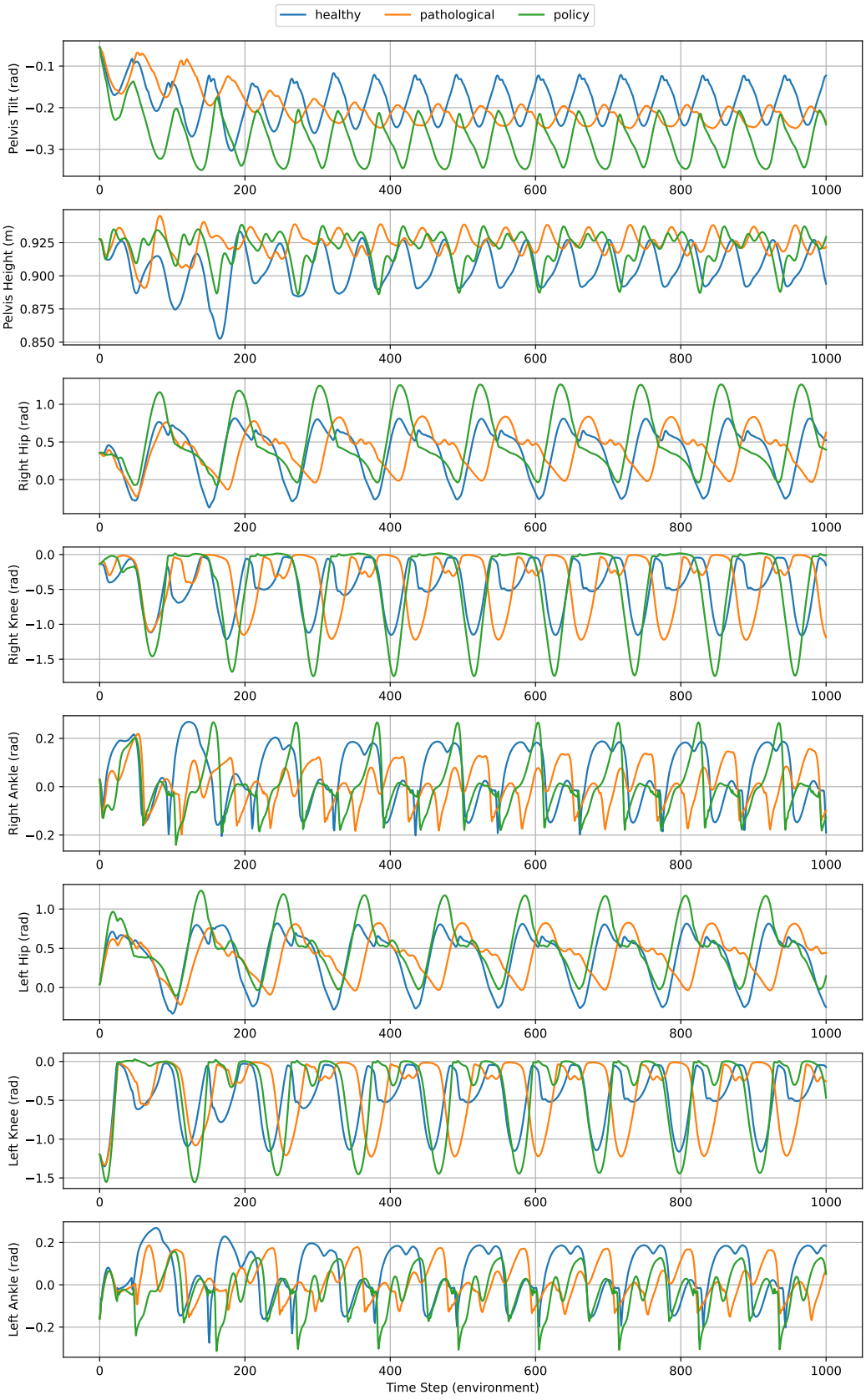

Figure A.14: Trajectories of various DoF at training step 2.0×10^7: healthy gait (blue), pathological gait (orange) and corrected gait (green). The model has the impairment with weakened iliopsoas and is trained by GAIL combined with LSTM and gradient penalty.

Bibliography

[Aal+25] Aali, S., Rezazadeh, F., Ardigò, L. P., and Badicu, G. “Altered lower extremity muscle activity patterns due to Iliopsoas tightness during single-leg landing”. In: *Scientific Reports* 15.1 (2025), p. 9477.

[AY20] Aguirre-Ollinger, G. and Yu, H. “Lower-limb exoskeleton with variable-structure series elastic actuators: Phase-synchronized force control for gait asymmetry correction”. In: *IEEE Transactions on Robotics* 37.3 (2020), pp. 763–779.

[Aka+16] Akalan, N. E., Kuchimov, S., Apti, A., Temelli, Y., and Nene, A. “Weakening iliopsoas muscle in healthy adults may induce stiff knee pattern”. In: *Acta Orthopaedica et Traumatologica Turcica* 50.6 (2016), pp. 642–648.

[Ale+17] Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. “Deep Variational Information Bottleneck”. In: *International Conference on Learning Representations*. 2017.

[An02] An, K.-N. “Muscle force and its role in joint dynamic stability.” In: *Clinical Orthopaedics and Related Research (1976-2007)* 403 (2002), S37–S42.

[Att+19] Attias, M., Bonnefoy-Mazure, A., De Coulon, G., Cheze, L., and Armand, S. “Kinematics can help to discriminate the implication of iliopsoas, hamstring and gastrocnemius contractures to a knee flexion gait pattern”. In: *Gait & posture* 68 (2019), pp. 415–422.

[ABH05] Au, S. K., Bonato, P., and Herr, H. “An EMG-position controlled system for an active ankle-foot prosthesis: an initial experimental study”. In: *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.* IEEE. 2005, pp. 375–379.

[Bel+24] Bellosta-López, P., Giner-Nicolás, R., Molina-Molina, A., Rubio-Peirotén, A., Roche-Seruendo, L. E., and Doménech-García, V. “Recovery of spatio-temporal gait and functional parameters following unilateral eccentric exercise-induced muscle damage in the hamstrings”. In: *Journal of Science and Medicine in Sport* 27.6 (2024), pp. 387–393.

[BSF94] Bengio, Y., Simard, P., and Frasconi, P. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[BHF23] Bengler, K., Harbauer, C. M., and Fleischer, M. “Exoskeletons: A challenge for development”. In: *Wearable Technologies* 4 (2023), e1.

[Bin+20] Bing, Z., Lemke, C., Cheng, L., Huang, K., and Knoll, A. “Energy-efficient and damage-recovery slithering gait design for a snake-like robot based on reinforcement learning and inverse reinforcement learning”. In: *Neural Networks* 129 (2020), pp. 323–333.

[Boh+24] Bohlinger, N., Czechmanowski, G., Krupka, M., Kicki, P., Walas, K., Peters, J., and Tateo, D. “One Policy to Run Them All: an End-to-end Learning Approach to Multi-Embodiment Locomotion”. In: *Conference on Robot Learning* (2024).

[BV04] Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

[Buc+04] Buchanan, T. S., Lloyd, D. G., Manal, K., and Besier, T. F. "Neuromusculoskeletal modeling: estimation of muscle forces and joint moments and movements from measurements of neural command". In: *Journal of applied biomechanics* 20.4 (2004), pp. 367–395.

[Che+22] Chen, L., Chen, C., Ye, X., Wang, Z., Liu, Y., Cao, W., Chen, S., and Wu, X. "A portable waist-loaded soft exosuit for hip flexion assistance with running". In: *Micromachines* 13.2 (2022), p. 157.

[Csi75] Csiszár, I. "I-divergence geometry of probability distributions and minimization problems". In: *The annals of probability* (1975), pp. 146–158.

[De +19] De Luca, A., Bellitto, A., Mandraccia, S., Marchesi, G., Pellegrino, L., Coscia, M., Leoncini, C., Rossi, L., Gamba, S., Massone, A., et al. "Exoskeleton for gait rehabilitation: effects of assistance, mechanical structure, and walking aids on muscle activations". In: *Applied Sciences* 9.14 (2019), p. 2868.

[Del+07] Delp, S. L., Anderson, F. C., Arnold, A. S., Loan, P., Habib, A., John, C. T., Guendelman, E., and Thelen, D. G. "OpenSim: open-source software to create and analyze dynamic simulations of movement". In: *IEEE transactions on biomedical engineering* 54.11 (2007), pp. 1940–1950.

[Elm90] Elman, J. L. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[FB21] Ficht, G. and Behnke, S. "Bipedal humanoid hardware design: A technology review". In: *Current Robotics Reports* 2.2 (2021), pp. 201–210.

[Fir+25] Firouzi, V., Seyfarth, A., Song, S., Stryk, O. von, and Ahmad Sharbafi, M. "Biomechanical models in the lower-limb exoskeletons development: A review". In: *Journal of NeuroEngineering and Rehabilitation* 22.1 (2025), p. 12.

[Gei+24] Geiβ, H.-J., Al-Hafez, F., Seyfarth, A., Peters, J., and Tateo, D. "Exciting action: investigating efficient exploration for learning musculoskeletal humanoid locomotion". In: *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*. IEEE. 2024, pp. 205–212.

[Gei19] Geijtenbeek, T. "Scone: Open source software for predictive simulation of biological motion". In: *Journal of Open Source Software* 4.38 (2019), p. 1421.

[Gei21] Geijtenbeek, T. *The Hyfydy Simulation Software*. https://hyfydy.com. Nov. 2021. URL: https://hyfydy.com.

[GH10] Geyer, H. and Herr, H. "A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities". In: *IEEE Transactions on neural systems and rehabilitation engineering* 18.3 (2010), pp. 263–273.

[GSB03] Geyer, H., Seyfarth, A., and Blickhan, R. "Positive force feedback in bouncing gaits?" In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 270.1529 (2003), pp. 2173–2183.

[Goo+20] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[Gou+21] Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. "Regularisation of neural networks by enforcing lipschitz continuity". In: *Machine Learning* 110 (2021), pp. 393–416.

[GMA19] Grabke, E. P., Masani, K., and Andrysek, J. "Lower limb assistive device design optimization using musculoskeletal modeling: a review". In: *Journal of Medical Devices* 13.4 (2019), p. 040801.

[Gul+17] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. "Improved training of wasserstein gans". In: *Advances in neural information processing systems* 30 (2017).

[GDG21] Guo, S., Ding, Y., and Guo, J. "Control of a lower limb exoskeleton robot by upper limb semg signal". In: *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE. 2021, pp. 1113–1118.

[Al-+23] Al-Hafez, F., Tateo, D., Arenz, O., Zhao, G., and Peters, J. "LS-IQ: Implicit Reward Regularization for Inverse Reinforcement Learning". In: *Eleventh International Conference on Learning Representations (ICLR)*. 2023. URL: https://openreview.net/pdf?id=o3Q4m8jg4BR.

[Ham+17] Hamaya, M., Matsubara, T., Noda, T., Teramae, T., and Morimoto, J. "Learning assistive strategies for exoskeleton robots from user-robot physical interaction". In: *Pattern Recognition Letters* 99 (2017), pp. 67–76.

[HE16] Ho, J. and Ermon, S. "Generative adversarial imitation learning". In: *Advances in neural information processing systems* 29 (2016).

[HS97] Hochreiter, S. and Schmidhuber, J. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[HZG90] Hoy, M. G., Zajac, F. E., and Gordon, M. E. "A musculoskeletal model of the human lower extremity: the effect of muscle, tendon, and moment arm on the moment-angle relationship of musculotendon actuators at the hip, knee, and ankle". In: *Journal of biomechanics* 23.2 (1990), pp. 157–169.

[HCC18] Huster, T., Chiang, C.-Y. J., and Chadha, R. "Limitations of the lipschitz constant as a defense against adversarial examples". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2018, pp. 16–29.

[LHG25] Lai, L., Huang, A. Z., and Gershman, S. J. "Action chunking as conditional policy compression". In: *Cognition* 264 (2025), p. 106201.

[Lee+23] Lee, Y.-H., Ko, L.-W., Hsu, C.-Y., and Cheng, Y.-Y. "Therapeutic effects of robotic-exoskeleton-assisted gait rehabilitation and predictive factors of significant improvements in stroke patients: a randomized controlled trial". In: *Bioengineering* 10.5 (2023), p. 585.

[LSE17] Li, Y., Song, J., and Ermon, S. "Infogail: Interpretable imitation learning from visual demonstrations". In: *Advances in neural information processing systems* 30 (2017).

[Li+22] Li, Z., Zhang, T., Huang, P., and Li, G. "Human-in-the-loop cooperative control of a walking exoskeleton for following time-variable human intention". In: *IEEE Transactions on Cybernetics* 54.4 (2022), pp. 2142–2154.

[Luo+23] Luo, S., Androwis, G., Adamovich, S., Nunez, E., Su, H., and Zhou, X. "Robust walking control of a lower limb rehabilitation exoskeleton coupled with a musculoskeletal model via deep reinforcement learning". In: *Journal of neuroengineering and rehabilitation* 20.1 (2023), p. 34.

[Luo+21] Luo, S., Androwis, G., Adamovich, S., Su, H., Nunez, E., and Zhou, X. “Reinforcement learning and control of a lower extremity exoskeleton for squat assistance”. In: *Frontiers in Robotics and AI* 8 (2021), p. 702845.

[Meh+23] Mehr, J. K., Guo, E., Akbari, M., Mushahwar, V. K., and Tavakoli, M. “Deep reinforcement learning based personalized locomotion planning for lower-limb exoskeletons”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5127–5133.

[Miy+18] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. “Spectral Normalization for Generative Adversarial Networks”. In: *International Conference on Learning Representations*. 2018.

[MFS23] Monteiro, S., Figueiredo, J., and Santos, C. “Towards a more efficient human-exoskeleton assistance”. In: *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE. 2023, pp. 181–186.

[Naz+23] Nazari, F., Mohajer, N., Nahavandi, D., Khosravi, A., and Nahavandi, S. “Applied exoskeleton technology: A comprehensive review of physical and cognitive human–robot interaction”. In: *IEEE Transactions on Cognitive and Developmental Systems* 15.3 (2023), pp. 1102–1122.

[PA10] Pandy, M. G. and Andriacchi, T. P. “Muscle and joint function in human locomotion”. In: *Annual review of biomedical engineering* 12.1 (2010), pp. 401–433.

[Pen+18] Peng, X. B., Abbeel, P., Levine, S., and Van de Panne, M. “Deepmimic: Example-guided deep reinforcement learning of physics-based character skills”. In: *ACM Transactions On Graphics (TOG)* 37.4 (2018), pp. 1–14.

[Pen+19] Peng, X. B., Kanazawa, A., Toyer, S., Abbeel, P., and Levine, S. “Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow”. In: *International Conference on Learning Representations*. 2019.

[PB24] Perry, J. and Burnfield, J. *Gait analysis: normal and pathological function*. CRC Press, 2024.

[PMA10] Peters, J., Mulling, K., and Altun, Y. “Relative entropy policy search”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 24. 1. 2010, pp. 1607–1612.

[QYF22] Qian, Y., Yu, H., and Fu, C. “Adaptive oscillator-based assistive torque control for gait asymmetry correction with a nSEA-driven hip exoskeleton”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 30 (2022), pp. 2906–2915.

[Qiu+20] Qiu, S., Guo, W., Caldwell, D., and Chen, F. “Exoskeleton online learning and estimation of human walking intention based on dynamical movement primitives”. In: *IEEE Transactions on Cognitive and Developmental Systems* 13.1 (2020), pp. 67–79.

[RLF21] Rodríguez-Fernández, A., Lobo-Prat, J., and Font-Llagunes, J. M. “Systematic review on wearable lower-limb exoskeletons for gait training in neuromuscular impairments”. In: *Journal of neuroengineering and rehabilitation* 18.1 (2021), p. 22.

[RBN22] Rose, L., Bazzocchi, M. C., and Nejat, G. “A model-free deep reinforcement learning approach for control of exoskeleton gait patterns”. In: *Robotica* 40.7 (2022), pp. 2189–2214.

[San+19] Sanchez-Villamañan, M. d. C., Gonzalez-Vargas, J., Torricelli, D., Moreno, J. C., and Pons, J. L. "Compliant lower limb exoskeletons: a comprehensive review on mechanical design principles". In: *Journal of neuroengineering and rehabilitation* 16.1 (2019), p. 55.

[Sch+16] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016.

[Sch+23] Schumacher, P., Haeufle, D. F., Büchler, D., Schmitt, S., and Martius, G. "DEP-RL: Embodied Exploration for Reinforcement Learning in Overactuated and Musculoskeletal Systems". In: *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*. 2023.

[Sha+24] Shan, X., Huang, Y., Bing, Z., Zhang, Z., Yao, X., Huang, K., and Knoll, A. "Locomotion Generation for a Rat Robot based on Environmental Changes via Reinforcement Learning". In: *arXiv preprint arXiv:2403.11788* (2024).

[SB+98] Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[SLK21] Sylvester, A. D., Lautzenheiser, S. G., and Kramer, P. A. "A review of musculoskeletal modelling of human locomotion". In: *Interface Focus* 11.5 (2021), p. 20200060.

[The+05] Thelen, D. G., Chumanov, E. S., Hoerth, D. M., Best, T. M., Swanson, S. C., Li, L., Young, M., and Heiderscheit, B. C. "Hamstring muscle kinematics during treadmill sprinting". In: *Med Sci Sports Exerc* 37.1 (2005), pp. 108–114.

[TET12] Todorov, E., Erez, T., and Tassa, Y. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[Unl+11] Unluhisarcikli, O., Pietrusinski, M., Weinberg, B., Bonato, P., and Mavroidis, C. "Design and control of a robotic lower extremity exoskeleton for gait rehabilitation". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 4893–4898.

[WFD23] Wakeling, J. M., Febrer-Nafría, M., and De Groote, F. "A review of the efforts to develop muscle and musculoskeletal models for biomechanics in the last 50 years". In: *Journal of biomechanics* 155 (2023), p. 111657.

[Wan+13] Wang, L., Wang, S., Asseldonk, E. H. van, and Kooij, H. van der. "Actively controlled lateral gait assistance in a lower limb exoskeleton". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 965–970.

[Xue+19] Xue, T., Wang, Z., Zhang, T., and Zhang, M. "Adaptive oscillator-based robust control for flexible hip assistive exoskeleton". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3318–3323.

[Yua+19] Yuan, Y., Li, Z., Zhao, T., and Gan, D. "DMP-based motion generation for a walking exoskeleton robot using reinforcement learning". In: *IEEE Transactions on Industrial Electronics* 67.5 (2019), pp. 3830–3839.

[Zha+22] Zhang, S., Guan, X., Ye, J., Chen, G., Zhang, Z., and Leng, Y. "Gait deviation correction method for gait rehabilitation with a lower limb exoskeleton robot". In: *IEEE Transactions on Medical Robotics and Bionics* 4.3 (2022), pp. 754–763.

[Zha+24] Zhang, Z., Huang, Y., Zhao, Z., Bing, Z., Cai, C., Knoll, A., and Huang, K. “Autonomous locomotion of a rat robot based on model-free reinforcement learning”. In: *2024 International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE. 2024, pp. 339–344.

[Zha+23] Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware”. In: *Proceedings of Robotics: Science and Systems*. 2023.

[Zho+19] Zhou, L., Chen, W., Chen, W., Bai, S., Wang, J., and Zhang, J. “Design of a compact rotary series elastic actuator with nonlinear stiffness for lower limb exoskeletons”. In: *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2019, pp. 68–73.

[Zhu+21] Zhu, F., Kern, M., Fowkes, E., Afzal, T., Contreras-Vidal, J.-L., Francisco, G. E., and Chang, S.-H. “Effects of an exoskeleton-assisted gait training on post-stroke lower-limb muscle coordination”. In: *Journal of neural engineering* 18.4 (2021), p. 046039.

[Zie+08] Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. “Maximum entropy inverse reinforcement learning.” In: *AAAI*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.