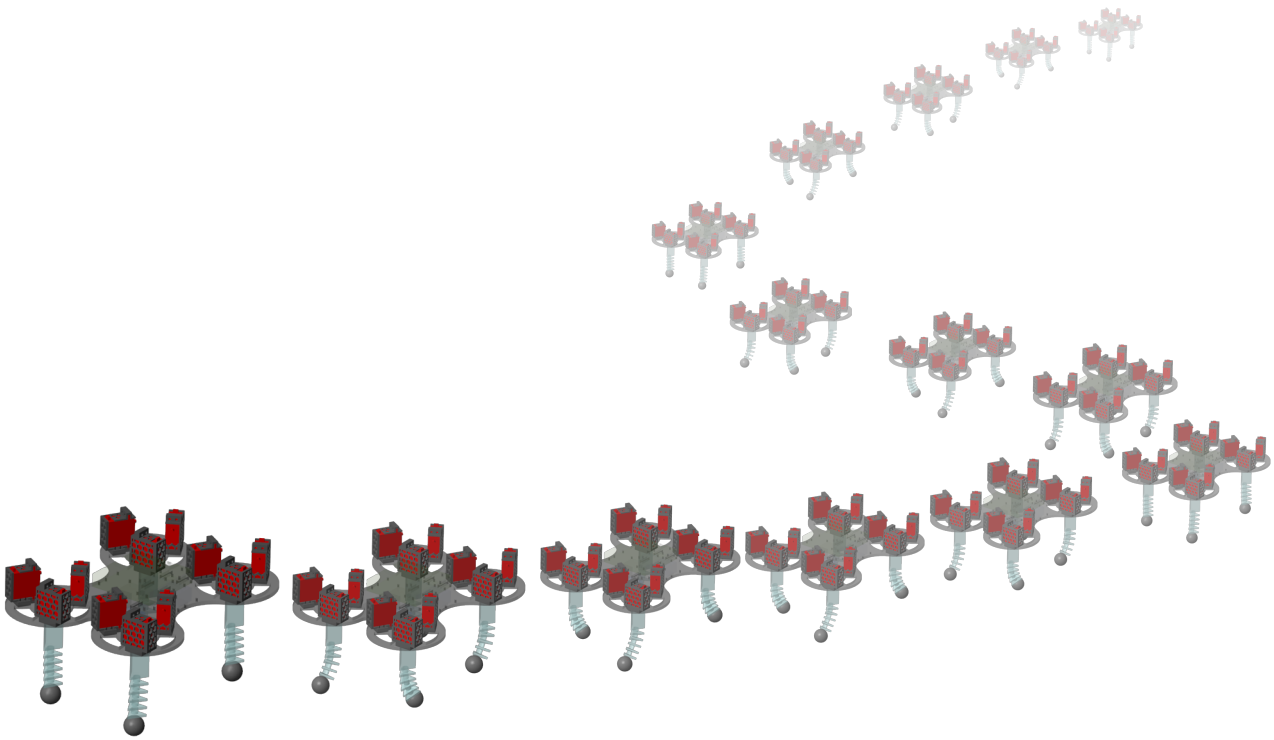
KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Mechatronics

Second Cycle, 30 credits

Optimal Gait Control of Soft Quadruped Robot by Model-based Reinforcement Learning

NIU Xuezhi



Stockholm, Sweden, 2023

Authors

NIU Xuezhi ⟨xuezhin@kth.se⟩
Msc. in Engineering Design - Mechtronics
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner

Lei Feng
Department of Machine Design
KTH Royal Institute of Technology

Supervisor

Tan Kaige
Department of Machine Design
KTH Royal Institute of Technology

Abstract

Quadruped robots offer advantages in navigating complex terrain without human intervention. They possess greater shock absorption capabilities, allowing them to withstand impacts effectively. In contrast, rigid robots, known for their rapid response and precise motion, often face limitations in their range of motion due to their inflexible mechanical components, such as joints and hydraulic actuators. Recent advancements in sensors, actuators, and computing have simplified real-world control of soft quadruped robots, enabling the development of control strategies to enhance their adaptability and efficiency across various terrains and environments. Additionally, the use of reinforcement learning in robotics control facilitates the achievement of autonomous and adaptable behaviors, as robots learn to optimize their performance through interactions with their surroundings.

However, despite advancements in quadruped robot control, the integration of advanced control methods like reinforcement learning requires extensive training and robustness considerations to ensure safe real-world operation. Adapting effective simulation-based control techniques to physical robots faces challenges arising from differences between simulated and real-world environments, necessitating improvements in reinforcement learning methodologies. This thesis aims to make a substantial contribution to enhancing robotic capabilities by employing optimal gait control for soft quadruped robots through model-based reinforcement learning. By addressing the complexities of real-world implementation and refining control structures, it seeks to facilitate the seamless integration of reinforcement learning into the field of soft quadruped robotics, thereby improving performance, adaptability, and autonomy.

This report includes the implementation of model-based reinforcement learning techniques for optimizing gait control, providing details on the simulation setup, reward system, and policy refinement process. It also offers a comprehensive presentation of the results, accompanied by an in-depth analysis that confirms the effectiveness of the proposed methods. This analysis demonstrates significant enhancements in training efficiency and autonomous behavior, validating the efficacy of the approaches taken.

Keywords:

Quadruped Robots, Soft Robotics, Reinforcement Learning, Gait Control, Model-Based Control Optimization

Acknowledgements

I am deeply grateful for the opportunity to complete this Master's Thesis and would like to acknowledge the support and guidance of several individuals who have contributed to the successful completion of this project.

First and foremost, I would like to extend my sincere thanks to the Mechatronics Department of Machine Design at KTH for their unwavering support throughout the duration of this project. I am particularly indebted to my academic supervisor, Mr. Tan Kaige, for his guidance and unwavering commitment to keeping me on track. Moreover, I would like to express my deep appreciation to the examiner, Prof. Lei Feng, for his interest in my work and for his valuable consultation. Dr. Fredrik Asplund deserves my gratitude for his invaluable guidance and advice regarding the research methodology, which helped shape the initial direction of this thesis work. In addition, I appreciate the assistance of Seshagopalan Thorapalli Muralidharan in identifying and resolving potential issues related to the electrical board and wiring.

I cannot overlook the unwavering support of my parents throughout my academic journey, and I deeply appreciate their encouragement, which has been a constant source of motivation. Furthermore, I would like to recognize the contribution of ChatGPT, the language model developed by OpenAI, for its invaluable assistance in reducing the written content in this thesis. You saved my life.

Lastly, I would also like to acknowledge myself, as I am also proud of the dedication, perseverance, and hard work I have put into completing this Master's Thesis.

NIU Xuezhi 牛雪芝

Stockholm, October 2023 ◡̈

Acronyms

3D Three Dimensional

4D Four Dimensional

Adam Adaptive Moment estimation

ANN Artificial Neural Network

ANCOVA Analysis of Covariance

CNN Convolutional Neural Network

COT Cost Of Transport

CPG Central Pattern Generator

CPU Central Processing Unit

CTSA Compressible Tendon-driven Soft Actuator

D-H Denavit-Hartenberg

DNN Deep Neural Network

DoF Degree of Freedom

FEA Finite Element Analysis

FL Front Left

FR Front Right

GPIO General-Purpose Input/Output

I²C Inter-Integrated Circuit

IDE Integrated Development Environment

IMU Inertial Measurement Unit

LSTM Long Short-Term Memory

MBRL Model-Based Reinforcement Learning

MFRL Model-Free Reinforcement Learning

MSE Mean Squared Error

NRMSE Normalized Root Mean Squared Error

PWM Pulse Width Modulation

ReLU Rectified Linear Unit

RL Reinforcement Learning/Rear Left

RMS Root Mean Square

RMSE Root Mean Squared Error

RMSProp Root Mean Square Propagation

RNN Recurrent Neural Network

RR Rear Right

SAC Soft Actor Critic

SGDM Stochastic Gradient Descent with Momentum

SLIP Spring-Loaded Inverted Pendulum

SoftQ A soft quadruped robot developed by KTH Mechatronics and Embedded Control Systems Unit

ToF Time of Flight

UART Universal Asynchronous Receiver-Transmitter

Contents

Chapter 1

Introduction

In this chapter, the background of soft quadruped robots and reinforcement learning approaches to its control is presented. Formulated research questions are listed together with methodologies. In addition, the limitations and delimitation to this thesis are discussed, as well as the ethics and sustainability analysis.

1.1 The Background

In the realm of engineering, robotics emerges as a magnificent confluence of mechanical, electrical, and computer science, orchestrating a symphony of autonomous systems designed to push the boundaries of human potential and expanding efficiency[1]. The involvement of robotic systems in our everyday lives has become increasingly commonplace, with its presence being felt across diverse fields such as manufacturing[2], agriculture[3], transportation[4], education[5] and even personal assistance[6]. In the domain of mobile robotics, the conventional approach to locomotion has been through the usage of wheels or tracks, making them apt for navigation on smooth surfaces[7]. Nonetheless, when it comes to maneuvering through unstructured and hazardous environments, such as the ones often encountered during search and rescue missions[8], industrial production lines[9], or scientific research endeavors[10], legged robots, characterized by their flexible structures, have demonstrated their worth. However, recent advancements in material sciences and design have given rise to a new breed of robots known as soft robots. These robots, owing to their deformable structure, have the unique ability to mold themselves according to their surroundings, which makes them ideal for interacting with humans or fragile objects in a safe manner[11]. This necessitates the development of advanced control strategies that can adapt to the dynamic and ever-changing morphology of these robots[12]. This thesis project seeks to investigate innovative control strategies that enable effective motion control of soft quadruped robots, contributing to the advancement of soft robotics technology.

In the previous studies[13] completed by the KTH Mechatronics and Embedded Control Systems Unit, tendon-driven soft continuum actuators were implemented in Three Dimensional (3D) /Four Dimensional (4D) printed structures to operate as legs of quadruped robots. This

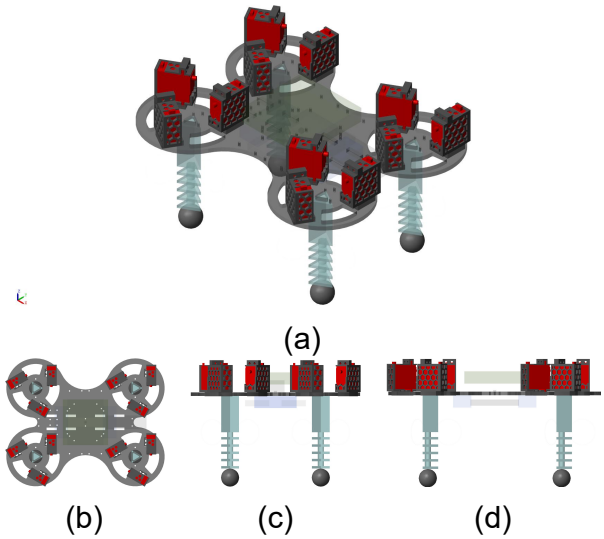


Figure 1.1.1: Rendered View of the Previous Quadruped Robot Model. (a) Isometric view; (b) Top view; (c) Front view; (d) Left view.

soft quadruped robot model developed in MATLAB's Simscape environment as demonstrated in Figure 1.1.1. Specifically, the tendons are used to transmit the force from motors, while the soft material acts as the core and the rigid disc acts as the tendon guide to form the actuator body. Since soft legs are inherently compliant to the terrain, they have had an increased capability of traversing complicated environments. Based on this soft leg, a soft quadrupedal robot prototype was developed and gait analysis was performed to enable the robot to walk[14], namely SoftQ. A "gait" in the context of robotics refers to the specific pattern or sequence of leg movements and body motions that a robot follows while walking or moving[15]. It is essentially the coordinated sequence of steps and motions that allows a robot to achieve stable and efficient locomotion[16].

Modelling for the actuators and legs are quite important for closed loop control, since the robots will benefit from feedback to the closed loop control when interacting with the terrine. Therefore, the KTH team[11] have already investigated the modeling process of a quadruped robot enabled by four tendon-driven continuum actuators on MathWorks Simulink®. Based on the developed soft robot simulation model, a gait controller was developed by a Reinforcement Learning (RL) algorithm Soft Actor Critic (SAC) recently[17].

RL has shown great potential in studying the control of quadruped robot motion due to its ability to learn complex control policies through trial-and-error interactions with the environment[18]. This approach is particularly relevant for soft quadruped robots, as their continuous and deformable morphology poses significant challenges for traditional control methods[19]. Moreover, RL enables the robot to learn from experience, allowing it to optimize its behavior based on feedback received from the environment in the form of a reward signal, thereby leading to the development of more efficient and robust control policies that can

handle complex and unpredictable environments[20]. Consequently, RL is ideally suited for studying the control of complex systems such as soft quadruped robots, which are challenging to model and control using traditional methods[18]. Although various RL algorithms have been used to develop policies for quadruped robots[21–23], challenges remain in developing RL-based gait control strategies. These strategies need to handle the continuous and deformable morphology of the robot while also being computationally efficient[24]. Hence, gait control of soft quadruped robots stands as a promising area of research with the potential to advance the field of soft robotics. However, the traditional learning methods suffer from challenges such as high sample complexity[25], instability during training[26], and the need for careful tuning of hyperparameters[27]. There is a clear need for further research to develop more efficient and effective RL control strategies that can handle the complexity of these robots and enable them to perform tasks in real-world environments[28]. Looking ahead, RL is likely to continue being applied for learning dynamic walking gaits for quadruped robots in both simulated and real-world environments. Moreover, RL is expected to be utilized for controlling the behavior of quadruped robots in various tasks, including but not limited to obstacle avoidance and terrain adaptation. These applications of RL hold great promise for advancing the field of robotics gait control and improving the efficiency and effectiveness of soft quadruped robots in various practical applications.

In short, the core of this thesis lies at the intersection of two rapidly developing research areas, soft-bodied robotics and reinforcement learning. Incorporating Model-Based Reinforcement Learning (MBRL), which involves utilizing models of the environment to improve the efficiency of learning, the thesis focuses on developing innovative control strategies using MBRL for proficient gait regulation of soft quadruped robots, intending to promote progress in the field and introduce flexible, efficient robotic systems proficient in navigating complex environment autonomously.

1.2 Problem Statement

The research at hand adopts a novel approach by utilizing Model-Free Reinforcement Learning (MFRL) for the development of innovative control strategies aimed at proficient movement regulation in soft quadruped robots[17]. Previous work has primarily focused on employing model-free reinforcement learning techniques to address challenges in larger state spaces, offering valuable insights into the potential of reinforcement learning for complex robotic systems. However, there remain several uncovered limitations that warrant meticulous consideration for future endeavors in this domain.

A prominent challenge is the significant time inefficiency inherent in training processes. The iterative nature of MFRL and complecity of simulation to soft robot can lead to protracted training times, hindering the real-time application of learned strategies[17]. To address this

challenge, innovative approaches have been explored. One such solution is the concept of MBRL, which offers a promising approach to mitigate the time efficiency problem. MBRL involves the creation of a functional representation of the robot's interaction with its external environment, closely resembling the physical model of the robot's dynamics, and providing precise state updates in an efficient manner [29]. Within the MBRL framework, the agent learns a surrogate model of past interactions, mapping specific actions to subsequent observations. This model enables the agent to predict the future state of the environment and the potential outcomes of different actions, empowering the agent to plan ahead and make informed decisions, thereby enhancing learning efficiency and task-solving capabilities[30]. The surrogate model within an MBRL framework is used to simulate the environment and predict the outcomes of different actions, rather than directly interacting with the real environment. This approach is considered to speed up learning and improve efficiency.

Furthermore, the complex dynamics of high-dimensional state and action spaces pose a formidable obstacle. The intricate interactions between the soft robot's flexible structure and its environment create a complex learning landscape[31], making exploration and convergence slower. In specific, the input to the RL agent consists of state space and action space, and the state space was defined by available sensor measurements, including robot moving velocity in three directions, rotational angle in three directions and normalized contact force on four feet. The action space was defined by motors on the legs, and three motors on each leg. Therefore, the state-action space of the reinforcement learning reaches 22 dimensions, which expands the computational requirements and leads to sparsity in the reward function. As discussed in the background, the state space of the surrogate model consists of state transitions states and reward function states, which will also reach a significant high dimensions. To address this limitation, exploring dimensionality reduction techniques, such as advanced feature extraction methods[30] or learned representations[32], could offer more concise and effective representations of the state space, thereby enabling quicker and more adaptive learning.

To address these challenges, this research employs approaches such as pattern-defined reinforcement learning and parameterization. In the previous training[17], the gait controllers were learnt from actuator-level, but a popular way to design the locomotion controller in rigid quadruped robots is to define certain gait pattern[33]. These includes trot, pace, bound, pronk, gallop, etc.[33]. Therefore, if the gait controllers could be defined on certain patterns, the gait controllers could be simplified by some certain gaits and some actions could be composed to reduce the state-action space so as to increase the learning efficiency[34]. For instance, the widely adopted trot, known for stability and balance[35], is chosen to restrict the movement of soft quadruped robots in reinforcement learning. Trotting is characterized by a two-beat diagonal gait, where diagonal pairs of legs move in unison. Specifically, the front left and rear right legs move together, followed by the front right and rear left legs [36]. Another method to restrict the state space of the plant model is parameterization, which parameterizing gait

phases and control policies based on the abstraction of a quadruped robot[37], as demonstrated in previous work[38].

In conclusion, the investigation centers on enhancing the control strategies of soft quadruped robots, bridging the gap between learned strategies and real-time application. The research seeks to contribute to the advancement of robotics by addressing the challenges posed by time efficiency and dimensionality while harnessing the potential of reinforcement learning in the context of soft robotic locomotion control. The subsequent section outlines the research questions framed to guide this study's exploration and investigation.

Research Questions

To elaborate, the problem addressed in this thesis was initially formulated by a set of research questions, which aimed to identify and explore the challenges and opportunities associated with gait control of soft quadruped robots. The following questions were designed to guide the research process and help frame the problem in a meaningful and relevant way.

1. How to restrict the state space or design a surrogate model with high estimation accuracy of soft quadruped robots plant compared to using Simulink Multibody functions? This model could be extracted as a representation of the real system, allowing for efficient and accurate simulations and training. Some methods considered to restrict the state space:

   (a) Pattern-defined reinforcement learning, it involves selecting a subset of features based on the certain pattern of quadruped robots, i.e. trot.

   (b) Parameterization, the higher-level abstractions of the state-action space, it will use phases between gait and real motors to parameterize gaits of quadruped robots.

2. In comparison to model-free RL, to what extend can the model-based RL approach generate a better RL agent and enhance the ability of a soft quadruped robot to walk in terms of stability, walking speed, and cost-of-transport? The enhancement than model-free RL is a benchmark of this project. Furthermore, it is important to evaluate the performance of model-based RL, and the evaluation of performance of this project focus on the simulation benefices, so what the trade-off is among the learning efficiency, the simulation accuracy and the long-term planning accuracy in order to train an optimal policy for gait control of soft quadruped robot?

The ultimate goal of this thesis was to advance our understanding of gait control for soft quadruped robots and contribute to the development of effective strategies for controlling their motion. This was achieved by addressing a set of research questions, the answers to which are presented in Chapter 6.

1.3 Scope

The objective of this study is to investigate and improve the learning efficiency associated with the current methods utilized for an optimal gait control of soft quadruped robots. This thesis proposes a MBRL approach to improve the learning efficiency and accuracy of optimal gait control while developing resilient and efficient gait control policies that can handle the continuous and deformable morphology of the robot. Moreover, the research seeks endeavors to make a valuable contribution to the development of more efficient and effective RL control strategies that can facilitate soft quadruped robots to perform tasks in real-world scenarios. To validate these concepts, practical physical tests are conducted to corroborate the research's practical applicability and effectiveness. Ultimately, this thesis also introduces a new software architecture tailored to the specific requirements of the soft quadruped robot SoftQ.

Limitations

The research faces the challenge of the "sim-to-real gap", where simulations may not perfectly replicate real-world conditions. Simulated environments, while controlled, may fail to capture the intricacies of real-world contexts. Consequently, results derived from simulations may not always align with real-world performance. This limitation stems from the potential disparity between simulation outcomes and real-world behaviors, affecting the practical applicability of simulation-based findings. Additionally, the interconnected nature of the robot's legs restricts the variety of gaits that can be explored, limiting the robot's capacity for certain types of locomotion that require individual leg movement. This constraint hampers the exploration of locomotion strategies relevant to real-world scenarios. Thus, controller evaluation primarily focuses on the robot's stability, walking speed, and cost-of-transport, rather than intricate algorithm details.

Delimitation

This thesis specifically concentrates on addressing challenges related to model-based reinforcement learning for optimal gait control in soft quadruped robots. To provide clarity and focus, specific delimitation bounds the scope of this study:

1. Exclusivity to Soft Quadruped Robots: The research exclusively considers soft quadruped robots and does not encompass other robot categories or diverse robotic systems. This deliberate focus ensures an in-depth analysis of challenges specific to soft quadruped robots.

2. Exclusion of External Factors: External factors like wind, varying terrains, and obstacles are intentionally omitted from consideration. While these factors significantly impact

robotic system performance, their exclusion allows for a concentrated exploration of model-based reinforcement learning challenges in gait control.

3. Assumption of Fixed Hardware Design: The study operates under the assumption that the hardware design of the soft quadruped robot is both fixed and operational. Variations or modifications in hardware design fall outside the scope of this research.

4. Limitation to Specific Algorithm: The research confines its investigation to a particular model-based reinforcement learning algorithm, namely, SAC. Alternative approaches, methodologies, or algorithms for gait control are not explored within this study.

These delimitations establish the boundaries within which this research operates, enabling a focused and comprehensive examination of the selected challenges in the defined context.

1.4 Methodology

In this degree project, a set of methodologies and methods have been employed to address the research questions and achieve the objectives. The methodology employed comprises four main stages, namely data collection and processing, model-based RL algorithm development and comparison, evaluation and analysis, and validation. Detailed description of these methodologies and methods will be presented in Chapter 3.

This research introduces a novel approach by suggesting that we separately train a surrogate model using RL methods. In this new method, the surrogate model is trained independently from the RL agent, separating the learning processes. This innovative strategy aims to make training more efficient. The surrogate model's job is to simulate the robot's environment and predict what will happen when different actions are taken. This approach, used within the MBRL framework, makes learning faster and more efficient.

Firstly, the signals to motors from the typical trot of quadruped robots were extracted by studying the actuation of the soft quadruped robot trot pattern. Subsequently, a model of the soft quadruped robot was obtained and used to generate simulation data using the Simscape model based on the extracted state space. A surrogate model was then designed with high estimation accuracy of the soft quadruped robot plant based on the processed data, which could effectively simulate the dynamics of the system and train an optimal policy for gait control. After that, a model-based reinforcement learning algorithm was developed for gait control of the soft quadruped robot using the extracted features and the reduced state space. The algorithm's performance was evaluated in simulation using metrics such as stability, walking speed, and cost-of-transport, and its design and parameters were iteratively modified to improve its performance. The previous model-free reinforcement learning algorithm's performance was also evaluated and compared using the same metrics. In the next step, the trade-offs between

learning efficiency, simulation accuracy, and long-term planning accuracy were evaluated in the context of training an optimal policy for gait control of the soft quadruped robot. The results were analyzed to draw conclusions about the effectiveness of the proposed model-based reinforcement learning approach compared to model-free reinforcement learning. Finally, the proposed approach was validated by implementing the optimal policy on the physical soft quadruped robot and measuring its performance in a real-world setting in terms of walking speed, stability, and cost-of-transport. The physical robot's performance was compared with the simulated results to validate the accuracy of the simulation and the effectiveness of the proposed approach. The graphical representation depicted in the Figure 1.4.1 provides a comprehensive outline of this thesis and enumerates the areas of investigation that have been explored.

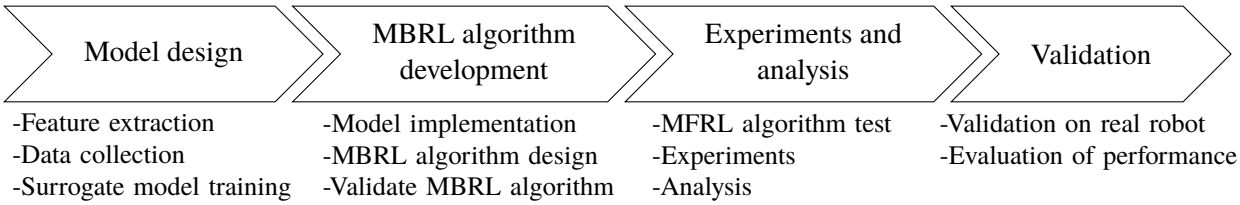


Figure 1.4.1: Overview of the methods in this thesis

1.5 Ethics and Sustainability

The use of robotics and artificial intelligence raises ethical and sustainability considerations that need to be addressed in this project. One significant ethical consideration revolves around the potential uses of the quadruped robot in fields like search and rescue, where it can play a crucial role in saving lives and aiding in disaster response. By developing an optimal gait controller for soft quadruped robots, this research contributes to the ethical benefit of improving the efficiency and effectiveness of these robots in life-saving scenarios. Furthermore, the project emphasizes the importance of non-military applications and research, ensuring that the quadruped robot's capabilities are channeled toward humanitarian and constructive purposes. This approach aligns with ethical principles, safeguarding privacy and human rights while maximizing the robot's positive impact.

Sustainability considerations also play a vital role in the project's framework. This encompasses the environmental impact of materials used in constructing the robot, energy consumption during operation, and waste generation. The project actively addresses these sustainability concerns by prioritizing the use of environmentally-friendly materials in robot construction and focusing on energy-efficient design and testing practices. By doing so, the project seeks to minimize its environmental footprint and contribute to the responsible development of robotic technology.

1.6 Outline

This paper is structured into 6 chapters, beginning with an introductory Chapter 1 that provides readers with necessary contextual information and a clear overview of this thesis. Chapter 2 is dedicated to a comprehensive review of existing literature in the field, with a particular focus on defining key concepts and providing a thorough problem description. This chapter also deals with the design of soft continuum actuator and the design of the basic structure used to train the neural networks and RL agent. Chapter 3 outlines the research methods utilized to answer the research questions. Chapter 4 is devoted to the implementation of detailed design and the experiments conducted for the research questions. Chapter 5 presents the results of the experiments, which also includes the major modifications made to the existing RL training method for the improvement. Finally, Chapter 6 provides a comprehensive summary of the research project and its future prospects.

Chapter 2

State of the Art

This chapter presents a comprehensive review of the literature, evaluating the current advances in the fields and identifying research gaps in the control of soft quadruped robots with the model-based reinforcement learning.

2.1 Robot Control

Robot control, especially for soft quadruped robots with a much higher degree of freedom than traditional quadruped robots, is a vital discipline that encompasses the movement and interaction of robots with their surroundings, enabling them to navigate complex terrains effectively[39]. Typically, the process of implementing control for a robot involves a hierarchical approach[40], which initiates with the lower-level control mechanisms and subsequently advances to more complex, task-specific controls. A representation of this hierarchical control structure in quadruped robots is provided in Figure 2.1.1. However, the hierarchy is not always concrete, different control methods can be employed for different subsystems of the robot.

At the lowest level, control of actuators, primarily motors, is achieved by developing firmware or low-level code[41], which enables precise task manipulation and regulates key parameters such as motor speed, rotation direction, or torque. This crucial aspect of motor control can be further integrated into a broader system, directly influencing the robot's kinematics and dynamics[42]. By considering the robot's mechanical structure through the application of kinematics and dynamics principles, control of robot body can be enabled by factoring in joint angles, leg positions, and forces exerted by a rigid quadruped robot[43]. Ideally, this aspect of robot kinematics and dynamics also extends to the orchestration of the robot's gait patterns, enabling more coordinated and efficient locomotion, but some are not[42].

Once these lower-level controls are established, the robot can achieve locomotion and perform basic movements. The incorporation of mid-level controls within the robot's control system endows the robot with enhanced capabilities and increased autonomy[44], wherein most controls in this level aims to optimize the performance. For instance, trajectory control

methodologies[45], which are of significant value for tasks necessitating precise positioning or adherence to specific motion profiles, permit the robot to adhere to predetermined routes or execute exact movements with precision and smoothness. Stability control strategies allow the robot to sustain balance and stability[46], even amidst challenging environments or during dynamic motions, thereby mitigating the possibility of instability or the risk of falling. Incorporated obstacle avoidance mechanisms within the robot's control system allow it to perceive environmental obstructions[47], generate optimal paths or control signals to bypass them, thus ensuring its safe, collision-free navigation and improved efficiency in traversing complex environments.

Finally, task-specific control strategies can be implemented, designating the comprehensive goals that the robot should accomplish. The strategic level represents the apex of the hierarchical structure of robot control systems and in essence, it centralizes on "what" the tasks are rather than "how" to carry them out. This involves complex high-level planning and decision-making[48], which are later converted into specific actions at lower levels. The comprehensive understanding and systematic implementation of these control mechanisms enable the development of robotic systems capable of precise and Optimal control in diverse environments and applications.

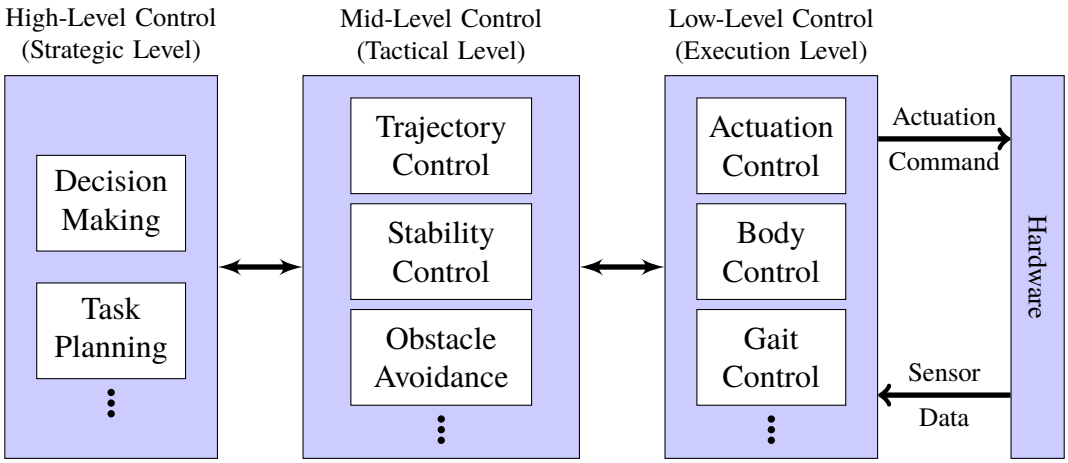


Figure 2.1.1: Hierarchical control levels in quadruped robots.

The control of soft quadruped robots is an area of active research due to the inherent difficulties involved. The KTH soft quadruped robot SoftQ represents a significant advancement in this field, and this thesis aims to contribute to its control system development at the lower-level control hierarchy, specifically motion control, gait control, and body control, there are several key considerations and techniques that can be explored to enable walking and achieve stable locomotion.

2.1.1 Kinematics and Dynamics

Kinematics and dynamics are two fundamental aspects of robotic systems that are closely related to motion control. Kinematics refers to the study of the geometric properties of motion without considering the forces or torques involved. In the case of rigid quadruped robots, kinematics focuses on determining the relationships between joint angles, leg positions, and the resulting end-effector positions and orientations. In the context of soft robots, kinematics involves understanding the relationships between deformations in the robot's soft structure and the resulting end-effector positions and orientations. By comprehending the kinematics of soft robots, it becomes possible to calculate the deformations required to achieve specific end-effector poses. The kinematic equations for soft robots can be challenging to derive due to the complex and deformable nature of their structures. Traditional kinematic models used for rigid robots, such as transformation matrices and Denavit-Hartenberg (D-H) parameters, may not be directly applicable to soft robots[49]. Instead, with precise modeling formulation of soft materials, Finite Element Analysis (FEA)[50] and the Jacobian method of statics[51] have been employed to accurately model the kinematics of soft robots. However, it is worth noting that these advanced methods can be computationally expensive, particularly when used in conjunction with RL training processes.

Dynamics, on the other hand, involves the study of the forces and torques that influence the robot's motion. In the context of quadruped robots, dynamics principles help in understanding how the forces and torques at each joint and leg affect the overall motion and stability of the robot. By modeling the dynamics of the system, one can accurately predict the forces and torques required to achieve desired motions or maintain balance. One of the famous models used to describe the locomotion dynamics of legged robots is the Spring-Loaded Inverted Pendulum (SLIP) model[52], which approximates a legged system as a single mass with a spring and/or a damper and it exhibits simplicity and effectiveness in capturing the fundamental dynamics of hopping and running.

Furthermore, control in robotics involves various models, from simple templates to a full rigid body model, a wide range of models existed to control quadruped robot in practice[53]. However, when it comes to soft quadruped robots, modeling them with simple kinematics and dynamics functions becomes challenging due to the presence of hysteresis, nonlinearity and other complex factors. Despite some research teams from KTH[14, 54] have successfully incorporated factors such as flexible body dynamics, non-linear material properties, and complex joint constraints, a research gap still exists in the development of real-time solutions, as existing models can be computationally expensive, limiting their practical applications.

2.1.2 Gait Control

By analyzing the kinematics and dynamics of robots, it is possible for us to identify gait patterns, joint coordination strategies, or control parameters that enhance the locomotion performance. In the case of quadruped robots, gait control plays a crucial role in coordinating leg movements for effective locomotion, which can be roughly categorized into two groups: modular design by employing the heuristic and optimal design to minimize the cost.

A modular controller approach involves breaking down a complex control problem into smaller, more manageable subproblems. Each subproblem is then addressed by a specific control module, which has a well-defined input-output interface and can be replaced or modified to generate different behaviors. Many research teams have developed modular controllers for specific robots[38, 55, 56], achieving impressive performance in locomotion and dynamic gaits. Despite their success, modular controller designs have certain limitations. Firstly, the development process is time-consuming and complex, requiring significant expertise and modifications for different tasks. Secondly, the simplifications and lack of accounting for coupling effects between modules can lower overall control performance, especially in dynamic maneuvers. Lastly, optimizing the contact schedule, a plan for when the robot's parts touch the ground, remains a challenging problem that current algorithms cannot handle[57]. As a result, most controllers resort to heuristic or hand-coded contact schedules.

The optimization-based gait control utilizes an effective cost function within a single optimization framework to find optimal motions for diverse tasks without a specific architecture for each task. The optimization of smooth and convex objective functions with multiple linear constraints has been extensively studied and is well-established in various fields of applied mathematics[58]. However, when it comes to planning trajectories for legged systems, the complexity of the problem poses significant challenges, and common optimization approaches are often not directly applicable. One key challenge in trajectory planning for legged systems is the need to consider multiple possible contact points with the environment. To simplify the problem, many existing algorithms make strong assumptions, such as limiting the number of contact points[59] and pre-specifying their locations[60]. By reducing the problem to a set of specific contact points, the complexity associated with collision primitives of the robot can be mitigated. Consequently, the task of finding an optimal contact trajectory becomes a convex problem that involves considering various combinations. To solve the combinational problems, one popular approach is to "softly" comply with contact dynamics by using a novel cost function[61]. Another approach involves predefining the contact sequence and effectively converting the model to a switched system with time-dependent switching[62]. All these approaches have some sort of limitations, a common criticism of the first approach is that the generated motion may violate physical constraints, leading to a lack of realism and the computational cost for the second approach can be substantial, making online optimization

impractical for legged systems.

As for soft quadruped robots, the challenges and considerations in gait control are further amplified. In the context of soft quadruped robots, gait control methods need to account for the deformations and interactions between the robot's body and the environment. Traditional approaches based on rigid-body dynamics and contact modeling may not be directly applicable in this case. However, the combination of modular design and optimization-based approaches can also be beneficial in gait control for soft quadruped robots.

2.1.3 Whole Body Control

Whole body control is a comprehensive approach that considers the robot as a whole system, coordinating the movements of all its limbs and body to achieve desired behaviors and tasks while accounting for interactions with the environment. By combining modular design and optimization-based approaches, whole body control capitalizes on the strengths of both methods. Modular design offers modularity and reusability of control modules, facilitating efficient development and behavior modification. On the other hand, the optimization-based approach provides flexibility and adaptability to various tasks and environments, enabling the generation of optimal control signals.These control architectures have demonstrated impressive performance on real systems. For instance, some research groups[56, 63] have successfully implemented such a controller for the quadruped robot, LittleDog and Mini Cheetah(Fig.2.1.2). Their controller comprised five modules: a footstep planner, a pose finder, a body trajectory generator, a foot trajectory planner, and a tracking controller. Each module produces desired values of physical quantities. This controller achieved significant success in the challenge program[64], Learning Locomotion (L2) from Defense Advanced Research Projects Agency (DARPA) and remains a state-of-the-art approach in rough-terrain legged locomotion control even after a decade of research.

(a) (b)

Figure 2.1.2: Rigid quadruped robot named (a) LittleDog[63] and (b) Mini Cheetah[56], both have whole body controllers implemented.

In soft quadruped robots, gait control becomes even more challenging due to the complex

and compliant nature of the robot's body, which also applies to the broader concept of whole body control. Soft robots require novel approaches to gait control that consider factors such as the interaction between soft materials, real-time sensing, and adaptation to varying terrain and environmental conditions. The integration of advanced control mechanisms, as discussed above, serves as a solid foundation for the introduction of robot learning into the control framework. By integrating these factors into the control system and leveraging machine learning techniques, it is possible to achieve stable and efficient locomotion in soft robots. Ongoing research in this area aims to develop advanced gait control strategies for soft quadruped robots, pushing the boundaries of their locomotion capabilities[65].

2.2 Robot Learning

While control algorithms govern robot behavior, robot learning complements this by enabling robots to acquire knowledge and improve their control strategies through experience and interaction with the environment. By incorporating supervised learning techniques, robots can go beyond pre-programmed behaviors and adapt their control policies based on real-time sensory information and learned models[66]. Robot learning refers to the ability of a robot to acquire knowledge and improve its performance through experience, interaction with the environment, and data-driven methods. It involves enabling robots to learn from their own actions, from human guidance, or from analyzing large amounts of data. The goal of robot learning is to equip robots with the ability to adapt, generalize, and improve their behavior over time without being explicitly programmed for every possible scenario. Robot learning empowers robots to autonomously acquire new skills, optimize their performance, and adapt to novel situations, enhancing their adaptability, flexibility, and efficiency.

By combining the principles of robust control with the capabilities of robot learning, the field of robotics continues to advance, striving to develop intelligent and optimal systems that can autonomously learn, adapt, and perform complex tasks in dynamic and uncertain environments. It is worth noting that the field of gait control for quadruped robots is continuously evolving, and researchers are actively exploring new approaches to control the gait of different robots. Among these approaches[44, 67–69], some utilize a parameterized control policy representation, where the parameters can be optimized using learning-based methods or hand-tuned. Two particularly popular approaches in this direction are Central Pattern Generator (CPG) and RL. The categorization here was based on historical perspectives rather than fundamental differences.

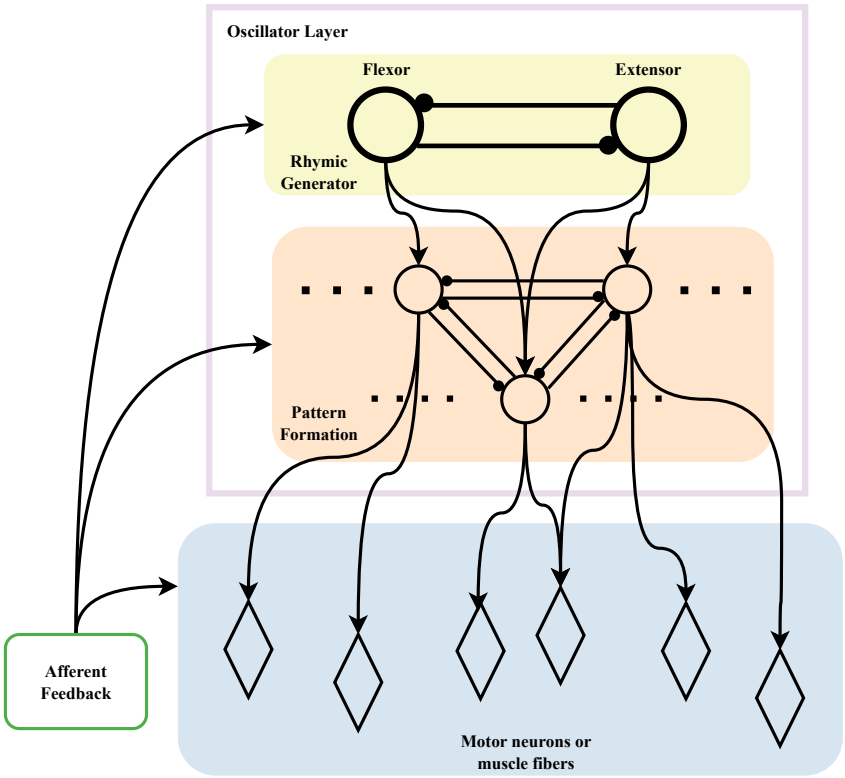


Figure 2.2.1: Schematic illustration of hypothetical CPG structure for mammalian movement control, reproduced from Li et al.[70]

2.2.1 Central Pattern Generator

CPG in robot can be thought of as a scheduler, responsible for a rhythmic pattern of leg movements that enables the robot to move in a coordinated manner. Originally found in the spinal cords of animals, CPGs provide the fundamental neural control required to generate rhythmic muscle activations, which drive various forms of locomotion like walking, running, or swimming. The Figure 2.2.1 depicts a three-level CPG concept from human for locomotion. It consists of a half-center rhythm generator, a pattern formation network, and a motor neuron layer. The yellow area represents the rhythmic generator layer with oscillators producing rhythmic signals as input to the pattern formation layer. The red area represents the pattern formation network, which contains interneuron populations that distribute and synchronize the rhythmic input to motor neuron pools via inhibitory connections. The motor neuron layer integrates sensory feedback and pattern formation network outputs to determine muscle activation. These networks demonstrate a fundamental understanding of how neural networks can generate rhythmic patterns for coordinated leg movements during locomotion. By emulating the functionality of CPGs, a gait control mechanisms for quadruped robots were successfully developed in a pioneering works[71], and it also plays an important role in recent advanced quadruped robots like the MIT Cheetah[72]. CPGs can be regarded as a specialized neural networks to generate the rhythmic patterns that facilitate locomotion. Artificial Neural Network (ANN)s are computational models that emulate the architecture and functionality

of biological neural networks, comprising interconnected nodes (neurons) that process and transmit information using weighted connections. Given the shared structural characteristics between CPGs and ANNs, it is common to consider the application of the training methods developed for ANNs to some CPGs.

2.2.2 Reinforcement Learning

Unlike other approaches, RL is built upon a flexible problem framework that can effectively handle various control challenges, includes a diverse range of scenarios such as stochastic, nonlinear, nonconvex, and nonsmooth problems, if sufficient training samples are available. Moreover, one of the key strengths of RL is its ability to tackle complex control tasks without relying on control architectures manually crafted by humans. Instead, RL only need a general approximation (e.g., a deep neural network) to learn and optimize control policies autonomously, degrading the need for extensive human design efforts. RL is a subfield of machine learning that focuses on how an agent can learn to make optimal decisions through interaction with an environment. It is inspired by the way humans and animals learn from trial and error to achieve desired goals. While employing RL approaches, the agent is designed to execute actions that maximize cumulative rewards within a defined time horizon. By assigning rewards or penalties based on the robot's performance, the algorithms will iteratively adjust the control policies to optimize the desired behavior. Through repeated interactions with the environment, the robot can learn to generate effective gaits and adapt to different terrains and disturbances. This approach has shown promising results in achieving dynamic and robust locomotion for quadruped robots[73].

As the complexity of robotics increases, RL is increasingly required to develop robust control algorithms with high Degree of Freedom (DoF)s and time variant properties[19]. The rapid development of AI provides an alternative solution to take into account the nonlinear properties of novel materials and soft robots[74]. In the reinforcement learning era, many new algorithms tailored for continuous control of robot have made it possible to learn to perform complex tasks. These algorithms, such as SAC[25] and Deep Deterministic Policy Gradient (DDPG)[75], have been successful in enabling continuous control in reinforcement learning[76, 77]. DDPG is a popular algorithm for continuous control. DDPG combines the power of deep neural networks with the deterministic policy gradient algorithm. Through the training process, DDPG learns an actor network, responsible for directly mapping states to actions, and a critic network, which estimates the expected return[75]. SAC is an off-policy algorithm that utilizes maximum entropy reinforcement learning to encourage exploration[25]. This algorithm excels in scenarios with continuous action spaces[76], making it highly suitable for a wide range of robotic control tasks. By optimizing a stochastic policy and simultaneously learning a value function to estimate the expected return, SAC achieves effective exploration and robust learning[76, 77].

Therefore, Ji et al. [17] selected SAC to train the robot to walk for better exploration efficiency and suitability for continuous action spaces.

Model-based Reinforcement Learning

Although Ji et al.[17]'s controller showed promising results, achieving an effective policy may still require prolonged training over several days or even weeks. The algorithm they proposed belongs to the category of MFRL, which involves complex simulations and can be significantly slowed down by them. In MBRL, an agent learns optimal behavior by creating a model of the environment through action and observation. Instead of explicitly modeling the environment, MBRL agents make decisions based on their experiences and rewards. This approach involves mapping observations to actions without delving into the underlying dynamics of the environment. To achieve that, a model of the environment comprising state transition distribution (P_η) and reward function (R_η) is created in MBRL.

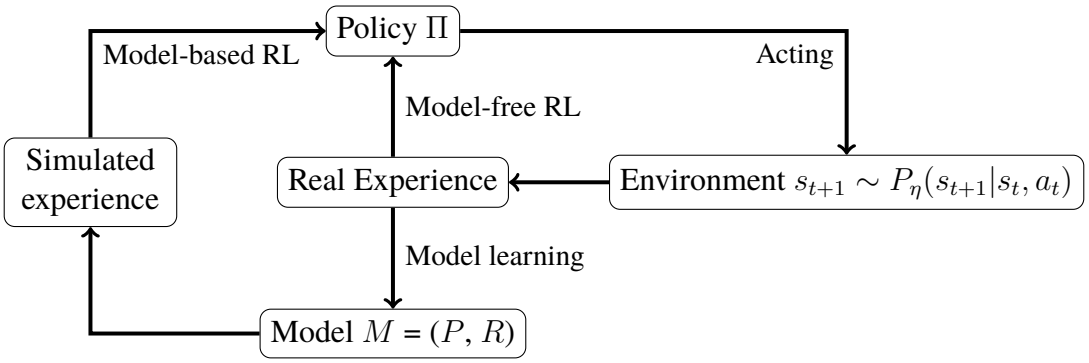


Figure 2.2.2: Model-free RL vs. Model-based RL

Typically, the model can be considered as a combination of state transition distribution P_η and reward function R_η,

$$M = (P, R) \text{ where } s_{t+1} \sim P_\eta(s_{t+1}|s_t, a_t) \text{ and } r_{t+1} \sim R_\eta(r_{t+1}|s_t, a_t)$$

s_t is the state at time t, r_t is the reward at time t and a_t is the action at time t.These components capture how the state transitions and rewards evolve as the agent takes actions in the environment. In other words, they describe the agent's interactions with the environment. By utilizing a learned model of the environment, MBRL allows for efficient exploration and faster learning. Safety considerations are also addressed, as MBRL allows for risk assessment and the avoidance of potentially harmful actions. In environments with sparse rewards, MBRL shines by generating informative trajectories, mitigating the challenge of exploration.

While MBRL requires the agent to build an abstract model of the environment, it needs to accurately represent the true behavior of the system for effective learning. However, in cases like the simulation of SoftQ conducted in Simulink, It is challenging to train the surrogate model

simultaneously with RL training. Thus, the training of a surrogate for a soft quadruped robot interacts with the environment using a data-driven approach.

The training of a gait controller for a soft quadruped robot using a data-driven approach in MBRL involves gathering empirical data to inform the control policy. However, an alternative MBRL approach utilizes a model, such as Neural Networks, of the robot and its environment to generate training data swiftly and cost-effectively. This approach facilitates the exploration of diverse control policies and behaviors while offering scalability and efficiency advantages over a purely data-driven approach. By employing deep learning techniques like neural networks, robots can effectively process and interpret complex sensory data, enabling them to understand and respond to their surroundings more comprehensively.

2.2.3 Neural Network

As a basic type of Neural Networks, ANNs are computational models that draw inspiration from the structural and functional characteristics of biological neurons[78]. The biological neuron serves as the fundamental building block of the nervous system, specifically designed to transmit electrical signals and facilitate signal processing. Its remarkable computational capabilities are evident through its capacity to process and integrate intricate information using electrochemical signaling[79]. It excels in generalization and adaptation by extracting shared characteristics from patterns, enabling the application of knowledge to novel scenarios. Additionally, biological neurons exhibit robustness and fault tolerance[80], ensuring reliable information processing even in the presence of noise, damage, or missing data. The parallel processing nature of neural networks enables efficient computation[81], as multiple neurons can concurrently process information, resulting in swift and simultaneous operations. Hence, emulating the neuron and constructing a network seems highly promising.

An artificial neuron, as a simplified mathematical model, designed to takes input signals, performs a weighted sum of those inputs, applies an activation function to produce an output, and passes it on to the next layer of neurons, the details is shown as Figure 2.2.3. The connections between neurons in an ANN are represented by weights, which determine the strength or importance of the signal transmitted between them. This learning process often involves techniques like supervised learning[82], unsupervised learning[83], or reinforcement learning[17].

Furthermore, the development of some advanced architectures and learning algorithms have been developed to address specific challenges. For instance, typical Deep Neural Network (DNN), often referred to feed forward neural network, employ a unidirectional flow of data, moving from the input layer to the output layer without revisiting any nodes. This architecture is ideal for tasks that require feature extraction and pattern recognition in static dataset[85]. Convolutional Neural Network (CNN) utilizes convolutional layers and pooling layers to

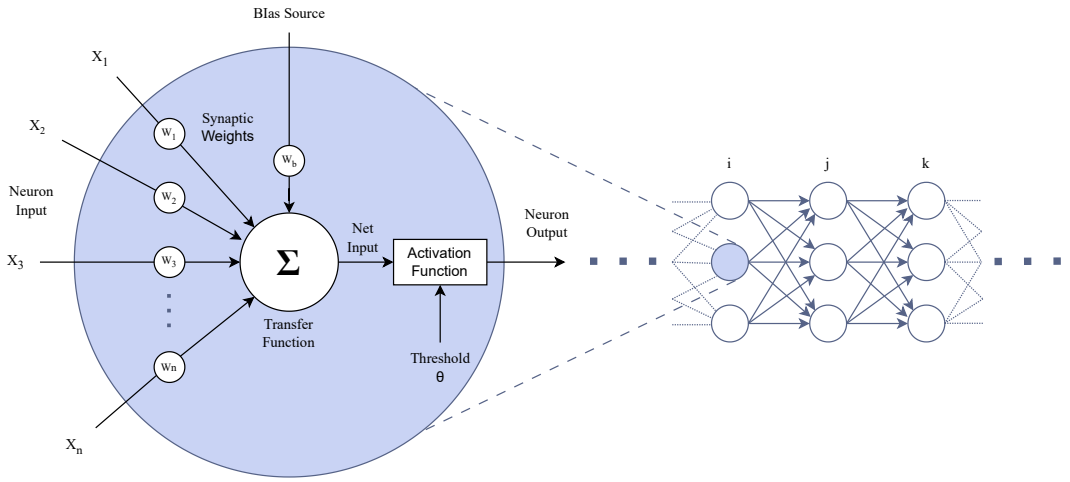


Figure 2.2.3: ANN Neuron structure and hidden layers distribution, reproduced from Shashwat et al.[84]

automatically capture spatial features and ensure computationally efficient[86]. Recurrent Neural Network (RNN) maintains an internal memory to be able to accommodate variable input lengths benefit them in processing sequential data[87]. Generative Adversarial Network (GAN) learn to capture underlying patterns and variations in the training data through adversarial training process[88].

2.3 Inspiration

In terms of current trends, efficient locomotion and interaction in quadruped robots, including soft quadruped robots with higher degrees of freedom, depend on effective robot control. A commonly employed approach to robot control is the hierarchical framework, or called modular framework, which begins with low-level control of actuators and progresses to higher-level task-specific controls. In specific, the precise control of kinematics and dynamics is crucial for ensuring the robot's movement and stability, while gait control coordinates leg movements to achieve efficient locomotion. Furthermore, the integration of modular frameworks and optimization-based approaches, for the whole body control, enhances the robot's capabilities and autonomy by considering the robot as a cohesive system. Additional control algorithms, robot learning methods such as Central Pattern Generators (CPG) and Reinforcement Learning (RL) enable robots to acquire knowledge and improve their control strategies through experience and interaction with the environment. The utilization of neural networks, such as Artificial Neural Networks (ANN), is prevalent in robot control and learning, enabling robots to effectively model complex relationships between sensory inputs and control outputs, thereby facilitating stable and adaptive gaits.

However, despite the progress made in these areas, certain research gaps still exist. One such gap lies in the realm of real-time solutions for modeling the intricate kinematics and dynamics

of soft quadruped robots, as existing models tend to be computationally demanding. Moreover, optimizing gait control for soft quadruped robots, taking into account the intricate interaction between their bodies and the environment, remains a challenging endeavor. Furthermore, while reinforcement learning algorithms exhibit promise in the realm of gait control, achieving an effective policy may necessitate extended training duration. Addressing this challenge could be accomplished through the adoption of model-based reinforcement learning approaches.

In summary, the synthesis of the key findings underscores the vital role that robot control plays in facilitating efficient locomotion and interaction in both rigid quadruped robots but the classic methods might not be able to merged in soft robotics control directly. The existing literature suggests that model-based RL has the potential to develop optimal policies for quadruped robots, including soft quadruped robots. However, the effectiveness of the approach depends on the accuracy of the learned model and the complexity of the environment. In addition, most existing studies have focused on simulated environments, and there is a need for further research on applying model-based RL to physical soft quadruped robots in real-world settings. However, research gaps persist, including the need for real-time solutions for modeling soft quadruped robots and optimizing gait control for their compliant nature. These trends indicate the ongoing exploration of control and learning techniques tailored to soft quadruped robots, with the ultimate goal of enhancing their adaptability and efficiency.

Chapter 3

Methodology and Experimental Setup

This chapter motivates the methodologies to address the research questions while incorporating engineering subject matter, such as models, hardware platforms, and software environments pertinent to this degree project.

3.1 Modelling

The process of modeling a robot is a fundamental step in developing an effective controller. Modeling soft-legged robots with soft actuators is challenging due to the non-linearity, time-variant properties, and unpredictable behaviors of soft materials. Despite these difficulties, research groups from KTH Royal Institute of Technology[13, 14, 17, 20] have successfully constructed a soft quadruped robot using soft continuum actuators, enabling it to walk. However, the robot's movement is not yet optimal, emphasizing the necessity for continued research and development in this field. This thesis work was based on this complete soft quadruped robot model, SoftQ, which is shown in Fig. 3.1.1(a).

3.1.1 Physical Model

SoftQ is a tendon-driven and lightweight robotic system that capable of locomotion similar to quadruped animals. The physical body of SoftQ is comprised of discrete subsystems, each playing a pivotal role in its operational intricacy. At its core, the robot's main body assumes the form of a hollow lightweight panel, housing an array of essential constituents, including servo motors, STM32 microcontroller, Raspberry Pi, electrical interface board, voltage converter, Inertial Measurement Unit (IMU) sensor, Time of Flight (ToF) sensors, batteries, and complex wiring infrastructure. The servo motor plays a pivotal role by transducing pulse signals into rotational torque via its shaft, subsequently translating this torque through a spool mechanism onto a connected tendon string. This interaction engenders the generation of tendon force, which is then transmitted to the actuator to induce controlled deformation. The deformation process is guided by the equilibrium between the applied tendon force and the reactive force exerted by the actuator. Because of the morphology of actuation systems, the whole actuator is so called

Compressible Tendon-driven Soft Actuator (CTSA) and the details are illustrated in 3.1.1(b)-(d). The robot's locomotion is enabled by these four CTSAs that are strategically grouped and positioned to govern the movement of individual legs, and each leg is actuated by three motor and CTSA systems. Notably, the CTSAs present a pioneering feature of SoftQ, encompassing

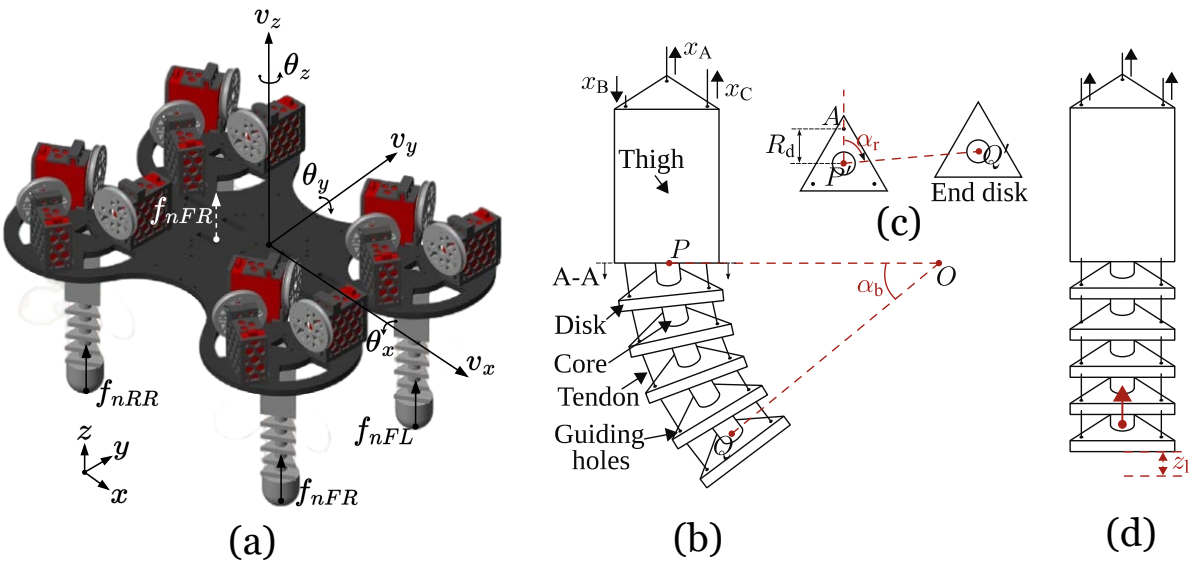


Figure 3.1.1: Graphical overview of SoftQ and the Compressible Tendon-driven Soft Actuator (CTSA) to drive the robot. (a) Rendered robot with key state notations. (b) The structure and notations of the CTSA. The upper part from section A-A is the rigid thigh for maintaining the actuator length and the lower part is compressible and bendable. The bending angle of the lower part is noted as α_b. (c) The top view of the bent CTSA of section A-A. α_r refers to the rotational angle of the CTSA. (d) The CTSA compression is realized by pulling the three tendons by the same amount, with the compression length being noted as z_l, originated from Ji et al.[17, 38]

soft continuum actuators that facilitate continuous bending or twisting motion throughout their length. Through three motor and CTSA systems that evenly distribute forces at radial locations, the robot's four legs attain the capacity for versatile omnidirectional bending[38]. Between the CTSA and the foot, sensory feedback of contact force is furnished by force sensors attached between them, detecting forces resulting from ground contact, depicted in Figure 3.1.2(d). The electrical interface board plays a vital role in this robotic system, facilitating the transmission of both signals and power among motors, microcontrollers, and power sources. Beneath this board, a Raspberry Pi is plugged and mounted by screws, while seamlessly integrated upon it are a STM32 microcontroller and an IMU. Additionally, ToF sensors are strategically positioned in varying orientations via external hubs, enabling multi-directional sensory input. Accompanying these components, voltage converters and batteries find their placement beneath the panel. Through interconnecting cables, they provide power to the servo motors via the interface board. This distinctive mechanism is visually demonstrated in Figure 3.1.1, effectively showcasing the robot's versatile and agile locomotion capabilities.

Of particular significance, the architecture of the CTSA entails a sequential arrangement of

rigid disks and flexible cores, the details of actuation are illustrated in Fig. 3.1.2. The interface between the servo motor's shaft and a connected spool serves to convert motor torque into tendon-based force. The end of the tendon are tethered to the spool and the terminal disk of the contiguous continuum structure. Rotation of the spool induces a corresponding change in tendon length. The central position of the spool, designated as position 0, corresponds to the neutral configuration where the actuator assumes a linear form. Positive spool rotations induce tendon retraction, while negative rotations become slack. Tendon-induced forces elicit actuator deformation, with the balance between tendon force and actuator reaction force governing deformation dynamics. The three driven tendons adopt a radial distribution around the actuator's rigid disks, passing through guiding apertures on said disks. The length of the actuators with core and disk structures has been modified, with the upper thigh region reinforced to enhance rigidity. The guiding apertures on rigid disks enable unfettered tendon motion while maintaining positional stability, as depicted in Figure 3.1.1. This specific attribute eliminates the requirement for discrete joints or segments, thereby contributing to the robot's innate flexibility and agility. To achieve optimal actuator bending while preserving overall robot balance, the servo motor's maximum rotational angle is capped at $\bar{a}_M = \pi/6$, corresponding to a rotational range of [-$\bar{a}_M$,$\bar{a}_M$]. This angular range corresponds to a Pulse Width Modulation (PWM) signal interval of [6.67%, 8.33%].

Furthermore, the inverse kinematics model of CTSA using three tendons can be derived through geometric analysis as discussed in [11], and is expressed as follows:

$$\begin{aligned} x_A &= R_d \alpha_b \cdot \cos(\alpha_r) \\ x_B &= R_d \alpha_b \cdot \cos(\alpha_r + \frac{2}{3}\pi) \\ x_C &= R_d \alpha_b \cdot \cos(\alpha_r + \frac{4}{3}\pi) \end{aligned} \tag{3.1}$$

Here, $x_{A,B,C}$ denote the tendon displacement. The positive values of $x_{A,B,C}$ indicate that the tendon pulls the respective actuator end, while negative values signify tendon release and slackening. R_d represents the radius of the circle formed by the three guiding holes on a disk as depicted in Figure 3.1.1(b). The geometric correlation between $[\alpha_b, \alpha_r]^\top$ and $[x_A, x_B, x_C]^\top$ can be encapsulated as a function: $g : \mathbb{R}^2 \rightarrow \mathbb{R}^3$. For the described CTSA, assuming that the compressed length of the actuator z_l results solely from equal-length pulling of the three tendons, the actuator's total degrees of freedom are established as three: $[\alpha_b, \alpha_r, z_l]^\top$. Consequently, the inverse kinematic model of the CTSA becomes $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$:

$$\begin{bmatrix} x_A \\ x_B \\ x_C \end{bmatrix} = f(\begin{bmatrix} \alpha_b \\ \alpha_r \\ z_l \end{bmatrix}) = g(\begin{bmatrix} \alpha_b \\ \alpha_r \end{bmatrix}) + z_l \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.2}$$

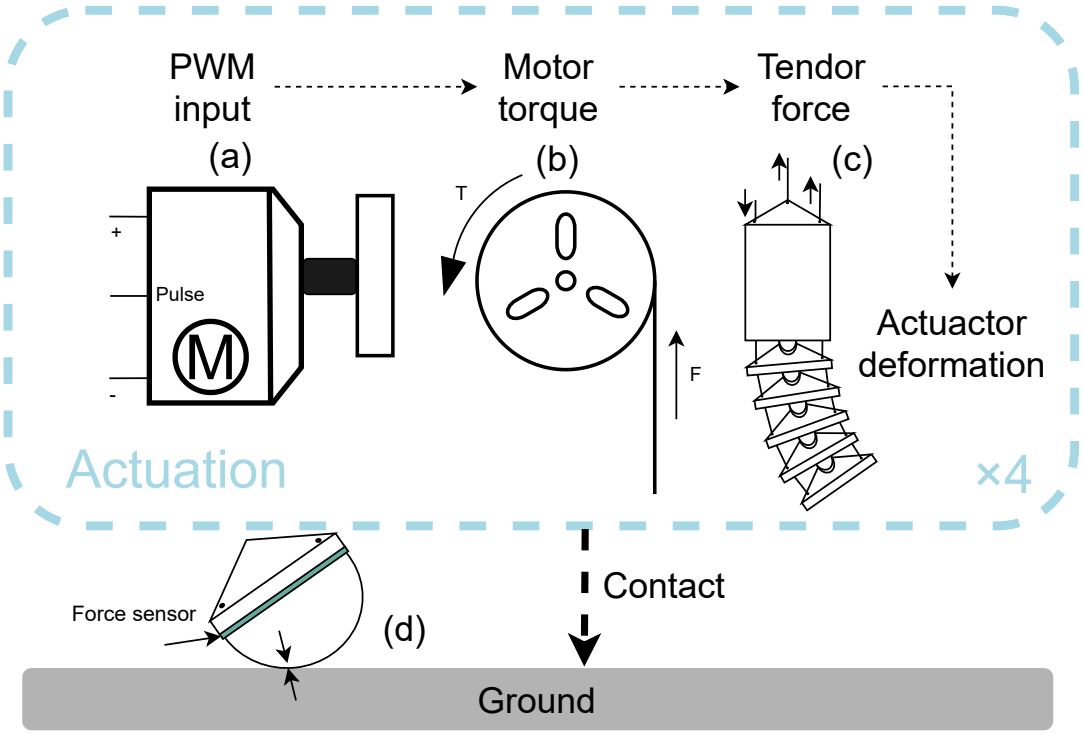


Figure 3.1.2: Overview of the quadruped robot's main subsystems enabled by tendon-driven soft actuators. (a) Servo motor converts PWM signal to rotational motions while applying torque exerted on the rotational spool. (b) The spool translates the torque from the servo motor to the traction force in the tendon. (c) The tendon force is applied on the actuator and effectuates its deformation. (d) The actuator's deformation introduces relative motion between the robot's feet and diverse ground conditions, thereby instigating the overall locomotion of the quadruped robot. Reproduced from Ji et al.[17, 38]

The SoftQ is a sophisticated robot for this project, and only part of sensors will be used to guide the optimal gait controller design. The servo motor interfaces with PWM signals, generating rotational torque that is subsequently propagated to downstream subsystems. The utilized servo motor is TS-411MG model from TrackStar, which receives PWM signals and rotates the shaft proportionally by a built-in position controller with corresponding sensors. In specific, the robot's feet are equipped with four force sensors each, labelled as Front Right (FR), Front Left (FL), Rear Right (RR), and Rear Left (RL). Constituting the sensing suite are five ToF sensors from the Adafruit VL53L1X series strategically positioned to measure distances from various sides, including the front, back, left, and right sides, as well as the bottom of the robot. Additionally, an IMU module sourced from Adafruit BNO055 is centrally mounted to facilitate the detection of the robot's pose and localization. For precise localization, the outputs of the ToF sensors are fused with the robot's rotational and translational accelerations, as quantified by the IMU. This sophisticated data fusion strategy culminates is developed in this thesis for the robot to accurately discern its position and orientation. This augmentation proves particularly advantageous for the successful implementation of MBRL. Particularly, certain sensors such as optical cameras and current sensors dedicated to motor control have been mounted on the robot but have not been integrated into the system. This deliberate omission stems from the

focused scope of this thesis, which primarily centers on lower-level control aspects and gait coordination.

In order to facilitate the gait control of SoftQ, the robot's local coordinate system has been established utilizing the conventions of the right-hand rule, which is shown in Fig.3.1.1(a). In this frame of reference, the x direction denotes the frontal orientation, while the anti-gravity direction aligns with the z direction. For capturing the robot's movements, the roll, pitch, and yaw motions are depicted through the rotational angles of the body center encompassing the three orthogonal axes, designated as θ_x, θ_y, and θ_z. The translational moving velocities are also represented by the movement of the body center in the three directions and are noted as v_x, v_y and v_z, respectively.

3.1.2 Modeling in MathWorks Simulink®

The Simulink model of SoftQ comprises multiple interconnected subsystems, each serving a distinct purpose. These include the motor-driven units, the tendon transmission mechanism, the flexible structure deformation system, and the ground contact subsystem. To model servo behavior in simulation, a PWM signals was employed to control the motor, attaching a spool and varying loads and adjusting servo parameters until the deviation between the experimental and the simulation data is minimal.

As visually depicted in Fig. 3.1.1, the CTSA is constructed through a sequential combination of flexible cores and rigid disks with triangle shapes to retain a high stability and fast fabrication. The three driven tendons are equally distributed at a radial formation around the rigid disks and passed through respective guiding holes placed on the rigid disks, and through the cooperation of three tendons deformation, the legs could realize omnidirectional movement. To model the above soft actuator deformations, the lumped parameter method was applied, where rigid components are applied to build the shape of the cores and the disks while using spring and damper joints for modeling the connections between the cores and disks and enabling the target bending and compression actuator motions. These joints use equivalent stiffness and damping to provide equal response of the entire actuator compared with the flexible segments, only needs to estimate the rigid connecting joint motions with stiffness k and damping d properties in the bending and longitude directions, denoted as k_B, d_B, k_L and d_L respectively. In the MATLAB Simscape Multibody environment, the continuum structure was modelled using rigid components with telescope joints. Through a series of simulation trials, the equivalent bending properties were estimated to $k_B = 0.0005Nm/deg$ and $d_B = 0.00002Nm/(deg/s)$, and the longitude stiffness and damping were estimated as $k_L = 15000N/m$ and $d_L = 800N/(m/s)$.

The tendons simulation was modelled using a belt-cable function within the MATLAB Simscape Multibody environment. This block facilitated a direct connection with the spools

and foot, omitting pulleys between the holes on the rigid disks for simulation efficiency. However, recognizing the variance introduced by this omission, a corrective coefficient, $\alpha_c = 1.16$, was introduced. This coefficient adjusts the servo motor's commanded rotation angle (θ) by multiplying it with α_c, aiming to bring the actuator's behavior in closer alignment with experimental observations. Furthermore, the use of continuum actuators in the robot, particularly the inclusion of the thigh segment, without the use of pulleys to secure tendon positions, introduces increased model errors due to the simulated detachment of tendons from the thighs. To overcome this issue, adjustments were made by relocating the spools on the servo motor shafts to the lower section of the thigh. This repositioning effectively prevents the tendons from exiting the thigh structure and consequently reduces the model error associated with the lack of pulley configurations.

The robot is placed on the ground with gravitational forces in the simulation environment. The diverse motions of the four legs result in relative movement with the ground, generating static or dynamic friction that propels the robot's overall movement. It is important to acknowledge that the distinct frictional properties of different ground conditions can lead to varied robot motions. To model the interaction between the quadruped robot's feet and the ground, static and dynamic friction coefficients were specifically determined as $f_s = 0.315$ and $f_d = 0.3$ respectively.

The update period of the RL algorithm is $T_s = 0.05s$ and the simulation model of the soft robot is a continuous-time model. The robot's state representation consists of ten key parameters: the roll, pitch and yaw rotations (θ_x, θ_y, θ_z), translational moving velocities in three axis (v_x, v_y, v_z) and contact force on each feet (f_{nFL}, f_{nFR}, f_{nRR}, f_{nRL}). To prevent the robot from falling down or entering unstable states in which the simulation results would be unreliable, termination boundaries are set in the simulation environment. When the robot states meet these termination conditions, the simulation ends before the desired simulation time. The roll and pitch angles of the body center θ_x and θ_y, are set to rotate less than 0.15 rad, while the bending angle $alph_b$ of all the four legs are set to less than $\pi/3$ rad.

3.2 Experiment Setup

The experimental setup for MBRL of the SoftQ quadruped robot encompasses a comprehensive integration of hardware and software components to facilitate the training and evaluation of the robot's locomotion control policies. The SoftQ robot, equipped with its sophisticated subsystems as described earlier, serves as the physical platform for conducting the experiments.

In this thesis, a practical methodology for autonomously learning and transferring agile and dynamic motor skills for complex and large legged systems were developed. This experimental

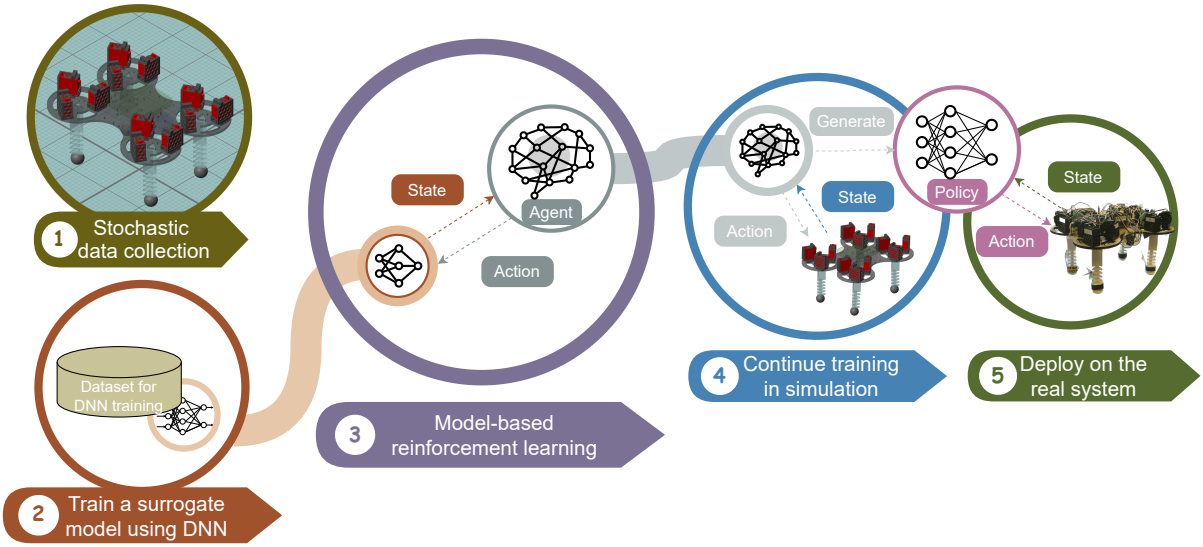


Figure 3.2.1: Creating a gait control policy. In the initial step, the physical parameters of the robot was identified and the stochastic actions were simulated in the identification to collect the data. In the subsequent step, an action-observation net was train that models complex robot model dynamics. The third step capitalized on the surrogate models produced in the previous two steps to train a control policy. In the fourth step, the trained control policy was further refined in simulation before deploying on the physical system.

framework constitutes a pivotal contribution to the field, encapsulated in the schematic representation shown in Figure 3.2.1. The central premise of this study revolves around the utilization of RL to train the gait controller through trial and error, which involves creating a virtual model of the robot and its environment to simulate possible walking behaviors and test the efficacy of various control policies. This approach is motivated by the challenges associated with collecting data in the physical world, which can be time-consuming, costly, and hazardous. By leveraging the benefits of simulation, the training process can generate a large volume of data quickly and inexpensively, allowing for the exploration of a broad range of behaviors and control policies without exposing the physical robot to potential risks.

Central to our approach is the utilization of a surrogate model to facilitate the training of controllers via the principles of deep neural network. This surrogate model serves as a vital intermediary in the training pipeline, mediating the interaction between the RL algorithm and the complex robotic system models. The architecture of surrogate model is embodied in a multi-layer perceptron, which operates as a neural network capable of capturing intricate relationships within the data. This multi-layer perceptron is engineered to process the potential sequence of the robot's states as input and subsequently generate the corresponding actions to be executed by legs as output. This architectural configuration enables the controller with the ability to encapsulate a wide array of state-action dependencies, thereby fostering expedited controller training. Preceding the training of the surrogate model, the collection of stochastic actions and their associated observations is requisite. This data collection process can be time-intensive, but its advantages extend to its reusability for the training of other controllers. This data-driven

approach allows for efficient iteration and optimization of the training process, resulting in a control policy that can be effectively applied to the robot.

The trained controller, obtained from the MBRL paradigm, would then proceed to a subsequent phase of refinement within the simulation environment. This phase entails simulation-based continuous training, which augments the controller's robustness and efficacy, thereby reducing the reality gap and enhancing its adaptability. Subsequently, the controller that has been refined in the virtual domain is seamlessly transitioned to the physical robotic system. This direct deployment showcases the transferability and generalization capabilities of the acquired motor skills, as the controller is able to translate its learned behaviors from simulation to the tangible real-world setting.

This methodology facilitates the application of deep learning and reinforcement learning techniques to develop a robust and efficient gait controller for the soft quadruped robot. The MATLAB functions developed and Simulink blocks constructed primarily utilize the CPU. The details of control architecture and the model-based reinforcement learning framework will be shown in the next chapter. During experimentation, the robot undergoes iterative training episodes, where it interacts with its environment to learn locomotion policies. The development environment is listed in Table 3.2.1. Thanks to efficient software implementations, no special computing hardware was needed, such as multi-CPU or multi-GPU servers, for training. All training sessions presented in this thesis were done on a personal computer with one CPU and one GPU, and none lasted more than twenty hours.

Component	Description
Operating System	Windows 10
Integrated Development Environment (IDE)	MATLAB R2021b
Central Processing Unit (CPU)	Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
Memory	16 GB
Graphics Processing Unit (GPU)	NVIDIA GeForce GT630

Table 3.2.1: Development environment for simulation and training

The collaboration of these hardware elements significantly enhances both the robot's operational capabilities and its potential for learning. A concise depiction of these hardware constituents is aptly captured in Table 3.2.2, highlighting pivotal components including servo motors, motor control boards, high-level control boards, interface boards, IMU units, ToF sensors, step-down regulators, and batteries. These components collectively contribute to the robot's ability to perceive its surroundings, execute precise movements, manage power, and establish connections with the external world. It gives rise to a comprehensive robotic system, primed to dynamically interact with its environment and gradually enhance its locomotion capabilities through the exploration of model-based reinforcement learning techniques. In

the next chapter, which includes details about control architecture and learning frameworks, the alignment between these hardware components and the research objectives becomes increasingly evident.

Component	Description
Servo motor$\times 12$	TrackStar TS-411MG (Information from: HobbyKing.com)
Motor control board	STM32G431KB (Information from: st.com)
High level control board	Raspberry Pi4 Model-B standard (Information from: raspberrypi.com)
Interface board	Designed by Muralidharan et al. [13] and manufactured by Shenzhen JLC Technology Group Co., Ltd
IMU	Adafruit BNO055 9-DoF@I2C (Information from: Adafruit.com)
ToF $\times 5$	Adafruit VL53L1X Breakout (Information from: mouse.com)
Step down regulator	LTC3780 DC-DC 5-32V to 1V-30V 10A (Information from: analog.com)
Battery $\times 2$	Li-PO 2S 7.4V 30C 1300mAh (Information from: batteriexperten.com)

Table 3.2.2: Hardware components used in the physical robot

3.3 Methodologies

Concerning research question 1, the state space restriction methods discussed should be confident to restrict the state space and increase the learning efficiency, since the algorithm is constructed directly from the state space, but the effectiveness of the surrogate model of SoftQ is undetermined. Therefore, an assessment of the effectiveness of the surrogate model is required, taking into account key metrics such as the Root Mean Squared Error (RMSE) of prediction on the gaits, correlation coefficient(R) between simulations and predictions on the reference, and Normalized Root Mean Squared Error (NRMSE) of observations from the sensors on the robot. Then, the independent variables that will be used are settings of parameterization parameters, rotational angle α_r, bending angle α_b and compressed length of the actuator z_l, which was determined by previous parameterization on the continuum actuators[38]. Thus, dependent variables are simulation time, prediction accuracy and long-term prediction accuracy, where long-term prediction was examined by correlation coefficient R.

- Independent variables:
  - Compressed length of the actuator limit(z_l (mm)), Type: Continuous
  - Bending angle limit (α_b (rad)), Type: Continuous

- Dependent variables:
  - Model accuracy, Type: Continuous, Metrics: Validation RMSE, Validation Loss, R and NRMSE of single step prediction
  - Long-term prediction accuracy, Type: Continuous, Metrics: R, NRMSE
  - Simulation time without failure, Type: Continuous, Metrics: time steps before failure

After the data was collected, the data should be preprocessed to conduct a correlation analysis to determine if there is a linear relationship between the independent variables and the dependent variable. Moreover, a general analysis will be employed to explore the trade-off between these multiple objectives. This analysis aims to identify parameter combinations or settings that achieve a desirable balance between model accuracy, long-term prediction effectiveness, and simulation efficiency. In addition, a best performance model should be determined from this test and proceed it to answer research question 2.

Regarding to research question 2, a one-way Analysis of Covariance (ANCOVA) test could be conducted to evaluate the performance of MBRL agent in comparison to MFRL agent in terms of stability, walking speed and cost-of-transport. Firstly, the independent variables are defined as the desired walking speed and the type of RL method, with two levels: MBRL and MFRL. The dependent variables will be stability, walking speed, and cost of transport, measured as performance metrics during the gait. Stability metrics should be high to indicate stable walking performance. The stability metric combines information about the time duration of the gait, angular velocity motion on z axis, and velocity in y axis to assess the stability of the gait, the equation is denoted by:

$$Stability = w_{time} \cdot (\Delta t) - w_{\dot{\theta}} \cdot max(|\dot{\theta}_y|) - w_v \cdot max(|v_y|) \tag{3.3}$$

where $[w_{time}, w_{\dot{\theta}}, w_v] = [0.2, 1, 1]$ stands for weights of different metrics, Δt is the time duration of the gait, and $\dot{\theta}_y$ represents the angular velocity on z axis. The specific values of the weights in the stability equation are chosen to balance the contributions of these three components. These values reflect the relative importance of each component in assessing stability. While walking speed and Cost Of Transport (COT) are direct performance metrics during gait, the COT is defined as

$$COT = \frac{E}{md} \tag{3.4}$$

where E is the energy consumed to walk, m is the mass of the robot and d represents the distance travelled by the robot. Additionally, learning efficiency can be measured by the training time required for the MBRL algorithm to converge to a satisfactory policy with the cumulative reward obtained by the policy reached a significant level. Data will be collected by simulating

the gait controller using both RL methods, and measuring the three performance metrics for each simulation run, resulting in three data sets for each RL method.

- Independent variables:
  - Type of RL method, Type: Categorical, Categories: Model-based, Model-free
- Covariance:
  - Desired walking speed (v_x (m/s)), Type: Continuous
- Dependent variables:
  - Stability, Type: Continuous, Metrics: Maximum angular velocity on z axis,
  - Resultant walking speed (m/s), Type: Continuous, Metrics: Average speed in validation
  - Cost-of-transport (J/kg/m), Type: Continuous, Metrics: Energy used by a weight to travel a distance
  - Learning efficiency (hrs), Type: Continuous, Metrics: Training time required for cumulative reward reach a significant level

A one-way ANCOVA test will be conducted with RL method type as one factor, and the desired walking speed as a covariance. The null hypothesis is that there is significant difference in the means of the performance metrics between the two RL methods with different desired walking speed. If the null hypothesis is rejected, it indicates that there is no significant difference in at least one of the performance metrics between the two RL methods. Furthermore, to validate the findings and evaluate the agent's ability in a real-world context, a field test will be employed. This real-world implementation will provide practical insights into how the RL agent's performance translates to real-world scenarios, thereby enhancing the robustness and applicability of the research.

Chapter 4

Model-based Reinforcement Learning

This chapter comprehensively examines the application of data-driven methods to implement MBRL for optimal gait control in soft quadruped robots, while also presenting the introduced robot's control architecture.

4.1 Method of Modelling in MBRL

4.1.1 Notation

The goal of reinforcement learning is to learn a policy that maximizes the cumulative rewards obtained over a sequence of time steps. At each discrete time step denoted by t, an agent interacts with its environment. The agent finds itself in a specific state $\mathbf{s}_t$ belonging to the state space $\mathcal{S}$, then performs an action $\mathbf{a}_t$ from the set of possible actions $\mathcal{A}$. Following the action, the agent receives an associated reward r_t, which depends on the chosen state and action. Afterward, the agent transitions to a new state $\mathbf{s}_{t+1}$ in a discrete time system. These transitions are determined by the underlying, yet unknown, dynamics function f, defined as a mapping from pairs of states and actions to subsequent states $f : (\mathbf{s}_t, \mathbf{a}_t) \rightarrow \mathbf{s}_{t+\Delta t}$.

In the context of MBRL, a surrogate model is developed to approximate the dynamics of the environment. This model is employed to predict the future outcomes of actions and states, aiding the agent in making informed decisions. The learned dynamics function is denoted as $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$, and it is parameterized by θ. This learned function accepts the current state $\mathbf{s}_t$ and the action $\mathbf{a}_t$ as inputs and produces an estimate of the future state $\mathbf{s}_{t+1}$ that the agent will encounter after taking the specified action. The parameter θ captures the characteristics of the learned function, determining its behavior. In this thesis, the chosen approach involves representing the learned dynamics function $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$ using a deep neural network. This neural network is designed to capture intricate relationships between the input state and action, allowing it to approximate the complex and often nonlinear dynamics of the environment.

Before initiating the training process, a dataset $\mathcal{D}$ was assembled through the collection of training samples. These samples were generated by initializing the system with various starting configurations denoted as $\mathbf{s}_0$. Subsequently, random actions $\mathbf{a}_t$ were executed at each time step.

These actions were drawn from a probability distribution $p(A)$. The outcomes of these actions were recorded, resulting in a sequence of states $(\mathbf{s}_0, ..., \mathbf{s}_{t-1}, \mathbf{s}_t, \mathbf{s}_{t+1}, ...)$. To ensure equitable treatment of distinct state components encompassing orientations, velocities, and forces, a preprocessing step is enacted. This step encompasses mean subtraction and standard deviation division of the dataset, ultimately leading to data distribution normalization.

Normalizing the data serves several vital purposes, such as stabilizing training, preventing sensitivity to input variations, mitigating gradient issues, expediting convergence, enhancing generalization, ensuring uniform feature treatment, and acting as a regularization technique. This normalization process is particularly beneficial in cases where state components exhibit significant scale variations, contributing to a more robust and generalized model.

4.1.2 Validation

The surrogate model $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$ was subsequently subjected to a training process employing the dataset $\mathcal{D}$. The goal of training was to minimize a specific loss function, specifically half of the Mean Squared Error (MSE). is a convention that simplifies the mathematical calculations involved in the training process. This convention simplifies mathematical calculations during the training process without altering the fundamental optimization problem. It streamlines gradient calculations and scales the loss appropriately. The formulation of the loss function is:

$$Loss = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}}^{\mathcal{D}} \frac{1}{2} \|\mathbf{s}_{t+1} - \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)\|^2 \tag{4.1}$$

where $\|\cdot\|$ denotes the Euclidean norm. Moreover, the RMSE was employed to quantify the accuracy of the predictions made by the surrogate model. This metric provides a comprehensive assessment of the residual errors between the predictions and the true values. The RMSE is computed as:

$$RMSE = \sqrt{\frac{\sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}}^{\mathcal{D}} \frac{1}{2} \|\mathbf{s}_{t+1} - \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)\|^2}{|\mathcal{D}|}} \tag{4.2}$$

During the training process utilizing the dataset $\mathcal{D}$, both the MSE and RMSE in the aforementioned expressions were computed not only on the training dataset but also on a distinct validation dataset $\mathcal{D}_{val}$, which was not part of the training dataset. Additionally, the surrogate model's performance was evaluated on a real walking gait dataset designed human expert, denoted as $\mathcal{D}_{real}$, employing the NRMSE:

$$NRMSE = \sqrt{\frac{1}{\mathcal{D}_{val}} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}_{val}}^{\mathcal{D}_{val}} \| \frac{\mathbf{s}_{t+1} - \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)}{max(\mathbf{s}_t) - min(\mathbf{s}_t)} \|^2} \tag{4.3}$$

While these error metrics offer an estimate of the surrogate model's predictive capability for

the next state, it is essential to assess its performance in predicting multi time step model behavior. This is crucial, as the model will be used for long-horizon control. To this end, the validation errors over a span of T steps were computed. This was achieved by employing the learned surrogate model to make multi-step open-loop predictions. For each given sequence $(a_t, ..., a_{t+T}) \in \mathcal{D}_{real}$ on the initial starting state s_0, a comparison was drawn between the corresponding ground-truth states $(s_t, ..., s_{t+T}) \in \mathcal{D}_{real}$ and the multi-step state predictions $(\hat{s}_t, ..., \hat{s}_T)$ generated by the learned surrogate model. This comparison was formulated as:

$$NRMSE^T = \sqrt{\frac{1}{\mathcal{D}_{val}} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}_{val}}^{\mathcal{D}_{val}} \frac{1}{T} \sum_{i=1}^{T} \| \frac{\mathbf{s}_{t+i} - \hat{\mathbf{s}}_{t+i}}{max(\mathbf{s}_t) - min(\mathbf{s}_t)} \|^2}, \hat{\mathbf{s}}_{t+i} = \begin{cases} s_t & i = 0 \\ \hat{f}_\theta(s_t, a_t) & i > 0 \end{cases} \tag{4.4}$$

Furthermore, a further comparison was carried out, involving the ground-truth state $(\mathbf{s}_{t+T})$ and the state prediction derived from the last computed step using the learned surrogate model $(\hat{\mathbf{s}}_{t+T})$. It is used to assess how closely the values of predictions and ground truth are correlated to each other. This comparison incorporated data from a total of N observations and was assessed using the correlation coefficient (R), as defined by the following equation:

$$R = \rho_{val}^N = \frac{\sum_{i=1}^{N} (\mathbf{s}_i - \bar{\mathbf{s}}_i)(\hat{\mathbf{s}}_i - \bar{\mathbf{s}}_{pred,i})}{\sqrt{\sum_{i=1}^{N} (\mathbf{s}_i - \bar{\mathbf{s}}_i)^2 \sum_{i=1}^{N} (\hat{\mathbf{s}}_i - \bar{\mathbf{s}}_{pred,i})^2}} \tag{4.5}$$

where $\bar{\mathbf{s}}_{pred,i}$ represents the mean prediction of the surrogate model for state $\mathbf{s}_i$ based on the actions $\mathbf{a}_i$ and is calculated as: $\bar{\mathbf{s}}_{pred,i} = \frac{1}{n} \sum_{i=1}^{n} \hat{f}_\theta(\mathbf{s}_i, \mathbf{a}_i)$

Importantly, the T-step validation procedure was exclusively utilized for result analysis and was not employed during the actual training process.

4.2 Neural Networks Design

When considering the creation of a surrogate model to expedite the RL training process, as discussed in Chapter 2, ANN comes to the forefront of the thoughts due to their inherent capacity to capture complex relationships within data and provide an efficient approximation of underlying dynamics[79]. As highlighted, there exist multiple viable approaches within the realm of ANNs, such as typical DNN, RNN and CNN. Although CNN are a subset of feedforward neural networks, their convolutional layers are specifically tailored to discern spatial patterns and features within multidimensional and local receptive fields. This attribute renders them particularly suited for analyzing the influence of actions on subsequent states in a structured manner. However, given that the model at hand pertains to complex temporal relationships, the utilization of CNN is not suitable within the scope of this thesis.

Belongs to ANN, a typical DNN consists of interconnected layers of artificial neurons, including the input layer, hidden layers, and output layer, which builds up the world of artificial intelligence and offers a straightforward design approach. The input layer receives raw data and transfers it to subsequent layers. The hidden layers undertake intermediate computations, employing activation functions to extract intricate features from the data. The output layer produces the final network output based on the assigned task, such as classification or regression. The design of the hidden layers can be meticulously tailored to accommodate various tasks, ensuring adaptability and versatility in tackling diverse computational challenges.

Emerging as a distinct architecture within the realm of ANN, RNNs present a distinctive architecture. Setting them apart from the conventional feedforward neural networks, RNNs incorporate loops within their structure, thereby enabling the assimilation of sequential and temporal information. This unique characteristic renders RNNs especially adept at handling tasks characterized by sequences, time series data, and other dynamic patterns[87]. Notably, each neuron within an RNN not only processes the immediate input but also taps into information from preceding time steps, resulting in the creation of an internal memory mechanism. This inherent recurrence empowers RNNs to effectively capture dependencies evolving over time and to manage sequences of varying lengths[89]. Given these attributes, RNNs are considered a choice for capturing the complex dynamics between actions and observations in time series data.

4.2.1 Neural Network Architecture

Continuing with the discussion of neural network architecture, it is important to underscore the significance of selecting an appropriate model for the specific task at hand. DNN serve as a foundational architecture in the realm of artificial neural networks. They are characterized by an intricate interconnection of layers consisting of artificial neurons, including input, hidden, and output layers.

In the context of DNNs, raw data is initially received at the input layer and subsequently propagated through the network's layers. The intermediate computations occur within the hidden layers, which are strategically positioned between the input and output layers. These computations involve the application of activation functions at each pair of hidden layers. A commonly used activation function is Rectified Linear Unit (ReLU)[90], known for introducing non-linearity by transforming negative values to zero while preserving positive values through element-wise activation.

A fundamental component within DNNs is the fully connected layer, also referred to as the dense layer[91]. This layer's distinguishing feature is that every neuron within it is interconnected with each neuron in the preceding and succeeding layers. This interconnectedness allows for comprehensive feature extraction and transformation. Neurons in the fully connected

layer apply a linear transformation to the input vector using weight matrices. Subsequently, a non-linear activation function, denoted as f, is applied to the product: $y_{jk}(x) = h\left(\sum_{i=1}^{n_H} w_{jk}x_i + w_{j0}\right)$. Finally, the output layer generates the network's ultimate output, tailored to the specific regression task at hand. The structural arrangement of a DNN is visually depicted in Fig. 4.2.1.

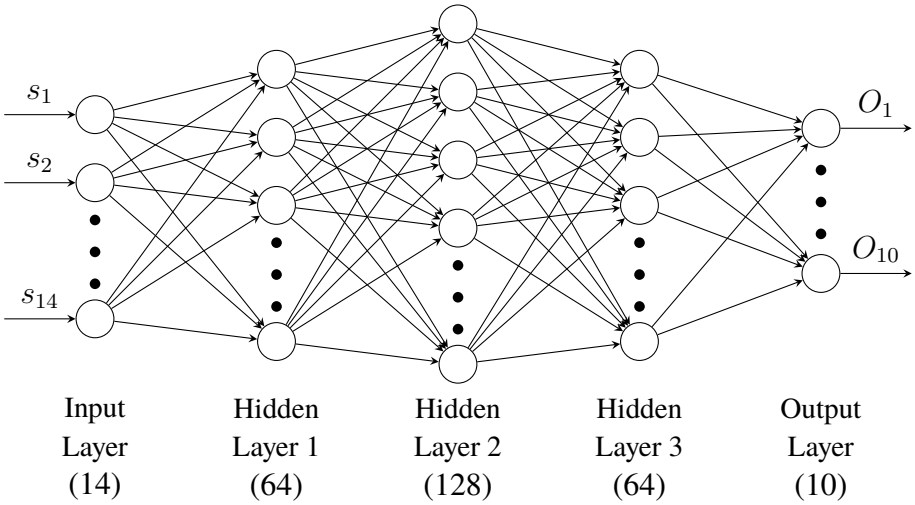


Figure 4.2.1: DNN Architecture. This diagram depicts a DNN structure characterized by three hidden layers. The initial layer comprised the input data, also followed by a fully connected layer featuring 64 neurons. Subsequently, a second fully connected layer with 128 neurons establishes connections, followed by another fully connected layer with 64 neurons. It's worth highlighting that each of these fully connected layers is ignited by a Rectified Linear Unit (ReLU) function, which adds non-linearity to the network's computations.

DNNs are particularly adept at capturing intricate and complex relationships within data[91], making them well-suited for a wide range of applications, including image recognition, natural language processing, and more. In contrast, RNN, characterized by loops in their structure, are ideal for handling sequential and temporal data, incorporating information from previous time steps to model temporal dynamics[87]. Each neuron within an RNN not only processes the current input but also incorporates information from previous time steps. This inherent recurrence equips RNNs to capture and model temporal dynamics, which is essential for scenarios where actions influence subsequent states over time[89]. RNNs employ loops to create a feedback mechanism where the output from one step becomes the input for the next. This cyclic connectivity enables RNNs to maintain a form of memory across different time steps[87]. As an evolution of the RNN architecture, Long Short-Term Memory (LSTM) layers incorporate specialized memory cells and gating mechanisms that enable them to selectively retain or forget information over extended sequences[92]. These mechanisms regulate the flow of information through the network, allowing it to selectively remember or forget information over longer sequences.

In the context of the SoftQ surrogate model, a bidirectional LSTM architecture is also adopted,

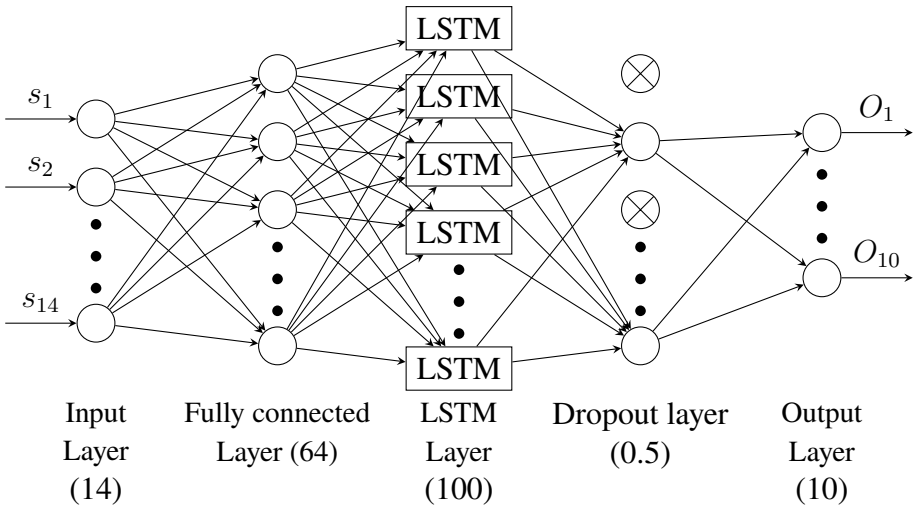


Figure 4.2.2: LSTM RNN Architecture. The figure illustrates the architecture of an LSTM-based RNN. The initial layer comprises the input data, followed by a fully connected layer featuring 64 neurons. Subsequently, the fully connected layer is seamlessly activated by a ReLU function before connected to a bidirectional LSTM layer with 100 hidden units. A Dropout layer with 0.5 probability is strategically incorporated before the output layers.

involving two distinct LSTM layers. The first layer processes the sequence in the traditional forward direction, while the second layer processes it in reverse. This bidirectional approach enhances the model's ability to capture bidirectional dependencies within the data. The architecture of the bidirectional LSTM RNN is visually depicted in Fig. 4.2.2. In addition to the LSTM layer, a dropout layer with a dropout rate of 50% is incorporated into the architecture to enhance robustness and generalization capabilities. The dropout layer is a regularization technique that helps prevent overfitting by randomly setting a portion of the input neurons to zero during training.

With the architectural groundwork laid, 15% of the training data was reserved for validation purposes, and the loss and RMSE of the training is shown in Fig. 4.2.4(a) and (c). Following this validation stage, the network was evaluated with some manual trot gait test data, the evaluation results are shown in Fig. 4.2.5.

4.2.2 Optimization Algorithms

Furthermore, it is essential to emphasize that alongside architectural considerations, the choice of optimization algorithms holds significant importance before the commencement of the training process. These algorithms steer the iterative parameter updates that guide the model towards convergence. Popular optimization algorithms include Stochastic Gradient Descent with Momentum (SGDM)[93], Adaptive Moment estimation (Adam)[94], and Root Mean Square Propagation (RMSProp)[95]. These algorithms adjust the weights of the model based on the gradients of the loss function with respect to the parameters. They help to guide the

model towards finding optimal parameter values that minimize the loss function.

SGDM augments the standard Stochastic Gradient Descent (SGD) by introducing a momentum term. This term incorporates the weighted average of the previous gradients to determine the direction of the parameter updates. This addition imparts momentum to the optimization process, allowing for smoother convergence by reducing the oscillations that can occur during gradient updates[96]. The momentum term helps to accelerate the movement along shallow directions while dampening oscillations in steep directions.

Adam is an optimization algorithm that combines the advantages of both momentum-based methods and adaptive learning rates. It maintains exponentially decaying moving averages of both past gradients and past squared gradients[97]. These moving averages are used to compute adaptive learning rates for individual parameters. Additionally, Adam employs bias correction mechanisms to counteract potential bias towards zero at the beginning of training. By adjusting the learning rates dynamically based on the historical gradients, Adam enhances convergence speed and adaptability across varying gradients[97].

RMSProp is an optimization technique designed to address the challenges of learning rate selection in SGDM. It maintains an exponentially decaying average of squared gradients for each parameter. The learning rate is then divided by the square root of this average, which normalizes the learning rate by the historical gradient magnitudes. This adaptive learning rate adjustment helps to mitigate the problem of diminishing or exploding gradients[98].

By implementing all three optimization algorithms in two networks, a comprehensive exploration was undertaken to discern the most suitable optimization strategy for the training of the surrogate model. This comprehensive approach facilitated a comprehensive assessment of each algorithm's impact on the training process and subsequent model performance. Specifically, the training was performed on both the bidirectional LSTM-RNN and DNN architectures, maintaining a consistent learning rate of 0.001. The training data was shuffled at the end of each epoch, and a mini-batch size of 512 was utilized. The training trajectories of the bidirectional LSTM-RNN and DNN networks, encompassing the utilization of the three optimization algorithms, are visually depicted in Figure 4.2.3.

For the bidirectional LSTM-based RNN architecture, SGDM is the emerged as the optimal choice due to its remarkable ability to achieve the lowest training loss and converge rapidly. It exhibited both the fastest convergence and the least training loss among the evaluated optimization algorithms. In contrast, for the DNN architecture, Adam optimization took the lead. Adam demonstrated superior performance by learning rapidly and converging to the lowest loss values, both in validation and training. Its ability to efficiently navigate the DNN's training dynamics underscores its suitability for this architecture. Notably, all three optimization algorithms proved to be stable options for training this architecture effectively.

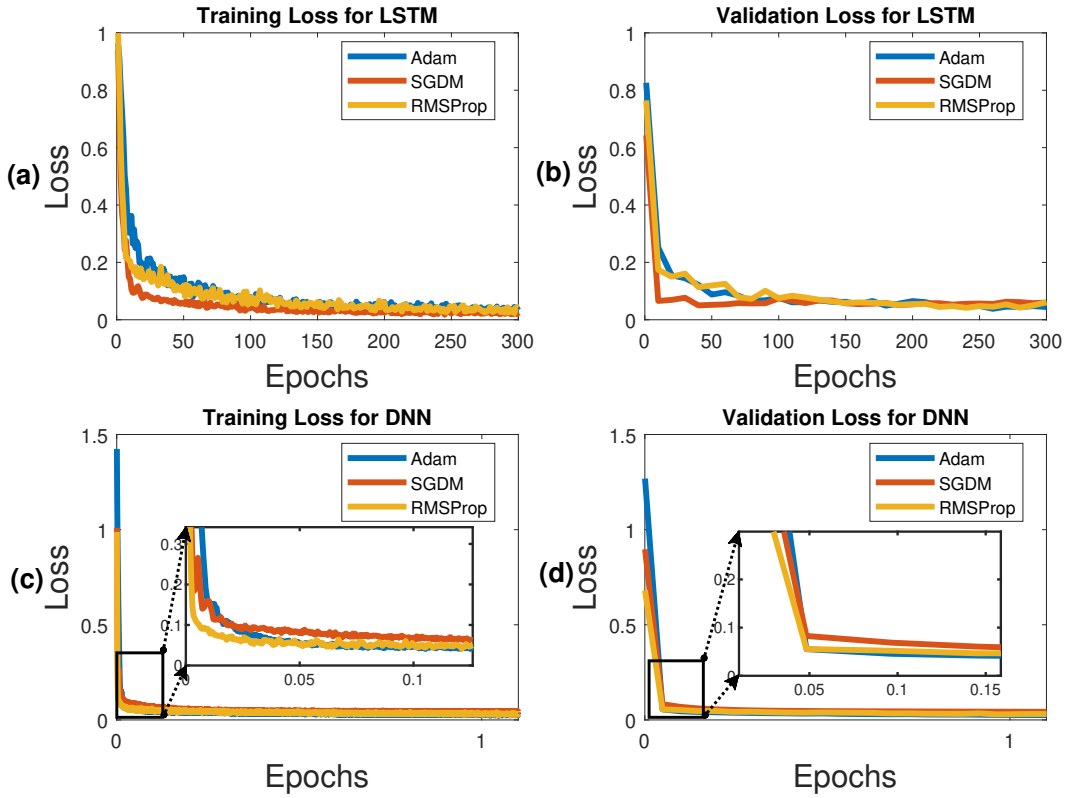


Figure 4.2.3: Comparison of Training Trajectories with Different Optimization Algorithms. The figure illustrates the training trajectories of two applied architectures while employing three different optimization algorithms: SGDM, Adam, RMSProp. (a) Training loss and (b) validation loss for the bidirectional LSTM-based RNN architecture, (c) training loss (d) validation loss for the DNN architecture.

4.2.3 Network Validation

Nonetheless, when comparing the training performance of the bidirectional LSTM-based RNN architecture and the DNN architecture, as illustrated in Figure 4.2.4, it becomes evident that the DNN architecture exhibits greater efficiency in terms of data utilization. Notably, the DNN showcases the capability to converge to a lower loss while achieving a reduced NRMSE. This underscores the DNN's efficiency in leveraging the provided data and achieving enhanced predictive accuracy with a more compact loss function. Furthermore, evaluated against the real expert trot gait simulation data, the bidirectional LSTM-based RNN architecture achieves an NRMSE of approximately 0.61 for single step prediction, whereas the DNN attains a significantly lower NRMSE of around 0.26. This substantial divergence in NRMSE values highlights the DNN's proficiency in making more accurate predictions on these test cases. A more comprehensive review of the single-step predictions, as compared to the expert trot simulation data, can be observed in Figure A.4.1 and Figure A.4.2 in the Appendix.

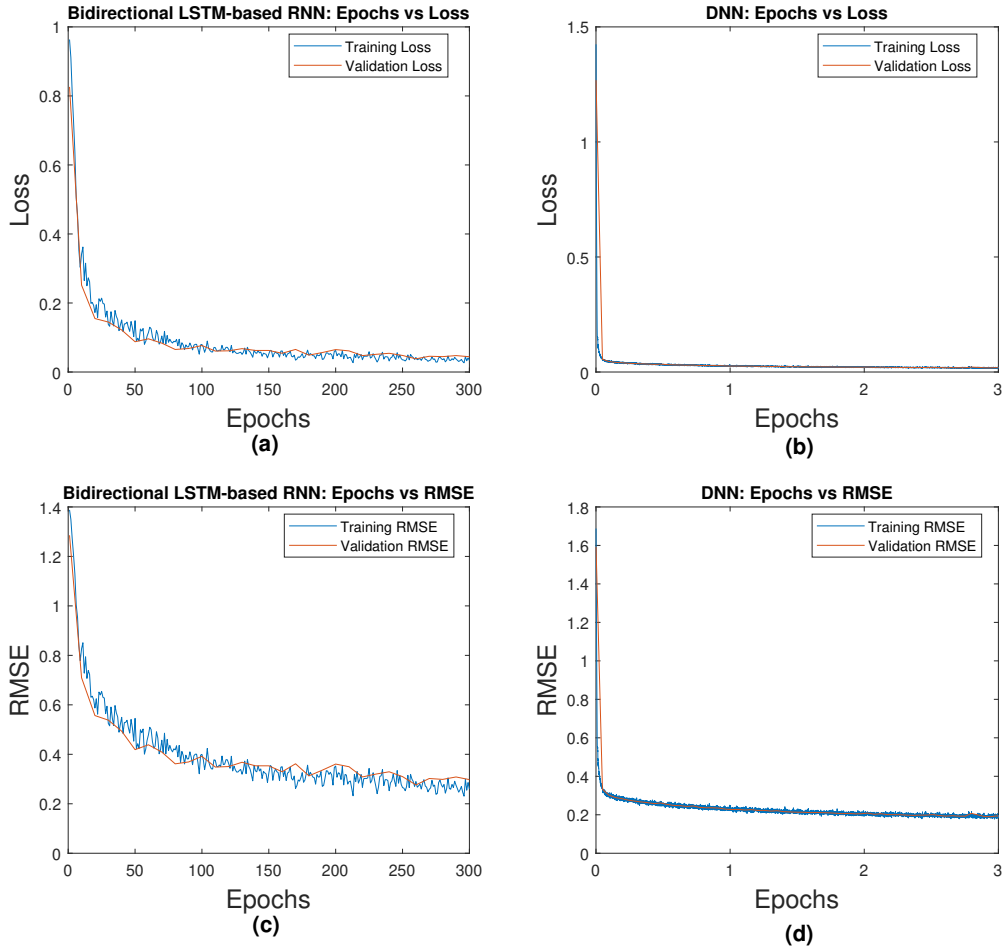


Figure 4.2.4: Training of bidirectional LSTM-RNN architecture and DNN architecture. (a)-(b) Loss vs epoch for model training with bidirectional LSTM-based and DNN model; (c)-(d) RMSE vs epoch for model training with bidirectional LSTM-based and DNN model.

In addition, an evaluation of the long-term predictions produced by both architectures, as illustrated in Figure 4.2.5, reveals valuable insights. The average correlation coefficient (R) for the DNN architecture's single-step predictions exceeded 0.9, maintaining a commendable level of above 0.76 even for predictions spanning 10 seconds. While there was a slight increase in the NRMSE for the DNN architecture from 0.26 to 0.35, the NRMSE remained relatively stable in the case of the bidirectional LSTM-based RNN. Additionally, it was also observed from the correlation coefficient (R) of different prediction steps that the bidirectional LSTM-based RNN architecture did not exhibit substantial drops, demonstrating its stability in predicting long-term and time-series data with consistency. On the other hand, the bidirectional LSTM-based RNN architecture demonstrated a single-step correlation coefficient (R) of around 0.64 and an NRMSE of approximately 0.62. For long-term predictions, these metrics do not surpass those achieved by the DNN architecture, indicating its limited ability to predict future states with high accuracy.

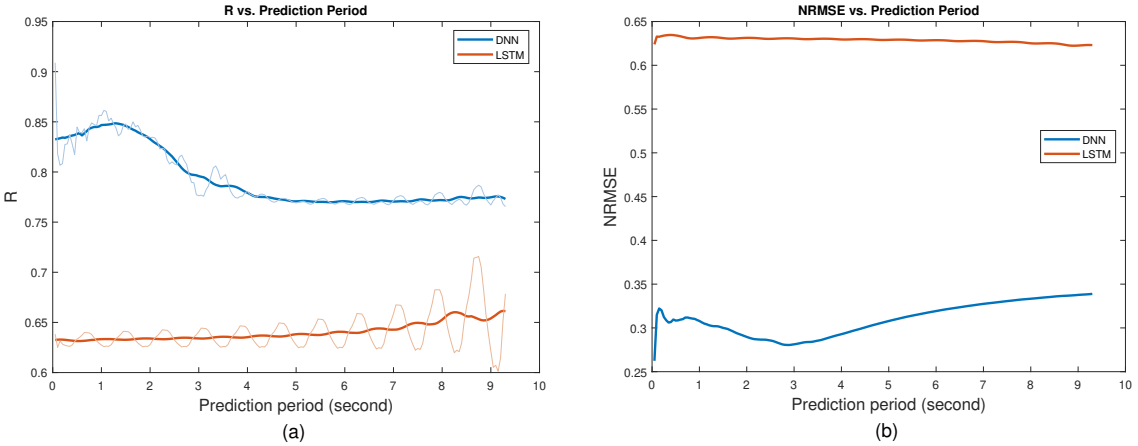


Figure 4.2.5: Comparison of Long-Term Predictions by Some Expert Gait Data $\mathcal{D}_{real}$. (a) This figure illustrates the average correlation coefficient (R) of long-term predictions between the DNN architecture and the bidirectional LSTM-based RNN architecture. (b) This figure shows the NRMSE of long-term predictions between the DNN architecture and the bidirectional LSTM-based RNN architecture.

LSTM is designed to capture long-term dependencies in sequential data, which can introduce unnecessary complexity when applied to problems with less pronounced sequential patterns. In contrast, DNNs are simpler and may be better suited for situations where data relationships are not strongly sequential. The underlying dynamics of the environment may involve complex and nonlinear relationships. DNNs are well-suited for capturing such intricate patterns, as they can adapt their internal representations to accommodate nonlinearities. While LSTM is a powerful choice for tasks with strong sequential dependencies, it may not be the best fit for problems where data relationships are less sequential. In this particular case, the characteristics of the problem favored the DNN architecture. LSTM, while proficient in handling sequential dependencies, may not excel in capturing these nonlinear relationships. The DNN's efficiency, simplicity, ability to capture nonlinear patterns, and stability in long-term predictions made it the preferred architecture for this particular task of surrogate model training. Therefore, the DNN architecture optimized with the Adam optimization algorithm emerges as a promising choice for surrogate model training.

4.2.4 Dataset Size

The neural network's architecture, as discussed earlier, provides the framework for representing the complex relationships within the soft quadruped robot's behavior. However, the size of dataset remains a critical factor in enabling the neural network to learn effectively and achieve high estimation accuracy. The dataset's size is defined as a sequence of data, where each sequence data refers to the data collected during a sampling session that occurs without any failures. This definition is established to guarantee the consistency and reliability of the data employed in the neural network's training.

A sufficiently large dataset is essential as it provides the neural network with a diverse set of examples that encompass various scenarios and conditions. This diversity allows the network to generalize well and effectively capture the underlying relationships inherent in the dynamics of the soft quadruped robot. The consequence of having a small dataset is the risk of overfitting, where the network memorizes the training examples but does not truly comprehend the underlying patterns. Conversely, a larger dataset should aid the network in learning these underlying patterns more comprehensively and reduce the risk of overfitting.

In Figure 4.2.6, the average results of 5 long-prediction tests with different dataset size are shown. The correlation coefficient (R) remains similar at the single-step prediction level, but for long-term prediction, R increases significantly as the dataset size grows from 20 to around 220 sequences of data. It is worth noting that our analysis identifies oscillations in the correlation coefficient (R). These oscillations can be attributed to the presence of an oscillation gait pattern within the data. This gait pattern introduces fluctuations in the predictions at different time steps. Furthermore, we observe that such oscillations are more pronounced when using LSTM networks. This behavior can be attributed to the LSTM's focus on capturing long-term dynamics, potentially overlooking instantaneous dynamics.

In addition to the correlation coefficient, the prediction accuracy was also assessed using NRMSE. The findings reveal a consistent trend: the NRMSE decreases as the dataset size grows from 20 sequences to around 280 sequences. These observations indicate that a larger dataset contributes to improved generalization, enhancing the network's ability to accurately estimate the behavior of the soft quadruped robot across a broader range of situations, including those has not encountered during sampling. This becomes particularly crucial when deploying the surrogate model in real-world scenarios where conditions may vary.

However, it is noteworthy that the trend reverses after 300 sequences of data and worsens further with 3000 sequences of data. The reason for the decline in performance with more data could be attributed to the fact that the random action data collected may not adequately characterize the useful information about the robot's dynamics. This could be due to the imbalanced distribution of data, with fewer samples representing high walking velocity, leading to the diminishing returns of adding more data in this form.

It is crucial to strike a balance between dataset size and the associated computational resources and time required for training. Collecting and curating a substantial dataset demands significant effort and resources, and training a neural network on a large dataset can be computationally intensive. Therefore, careful consideration is required to determine an appropriate dataset size that aligns with the available resources while still achieving the desired level of accuracy and generalization. Based on these observations, selecting a dataset size approximately 20 times the dimensions of the observation data seems to strike a reasonable balance for the surrogate model in this thesis, which means about 200 sequences of data in this thesis.

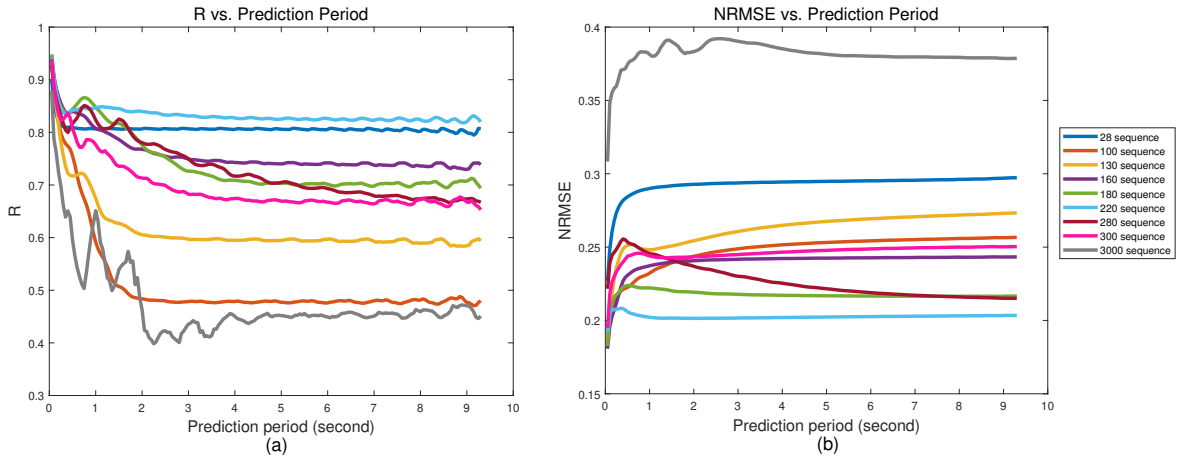


Figure 4.2.6: Comparison of Different Data Size of Dataset $\mathcal{D}$ in Long-Term Predictions using some expert trot gait data. (a) This figure illustrates the average correlation coefficient (R) of long-term predictions between the DNN architecture and the bidirectional LSTM-based RNN architecture. (b) This figure shows the NRMSE of long-term predictions between the DNN architecture and the bidirectional LSTM-based RNN architecture.

4.3 Model-based RL Algorithm

After successfully training the surrogate model, the next phase of this thesis involves the application of MBRL algorithms to further advance the research objectives. As previously discussed in Chapter 2, the SAC algorithm has been selected due to its advantages in promoting efficient exploration and its compatibility with continuous action spaces.

SAC addresses the challenge of dealing with continuous action spaces by incorporating an actor network that explicitly represents the policy function. This actor network is parameterized to output a probability distribution over actions, typically modeled as a Gaussian distribution with mean and standard deviation. SAC learns both the Q-function and the policy (actor) simultaneously through a combination of entropy-regularized policy optimization and Q-value estimation. In Figure 4.3.1, an overview of this approach to learn a walking gait controller for a quadruped robot using the SAC algorithm is presented.

4.3.1 SAC Algorithm

The SAC algorithm builds upon the principles of Actor-Critic but introduces a significant enhancement. To find an optimal gait controller through RL, the surrogate model is employed as an environment model within the framework of a Markov Decision Process $(\mathcal{S}, \mathcal{A}, P, r)$. In this construct:

- $\mathcal{S}$ and $\mathcal{A}$ characterize the continous state and action spaces, respectively.
- The transition function $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow [0, +\infty)$ encapsulates the probability density governing an agent's transition from the current state $s_t \in S$ to a subsequent state

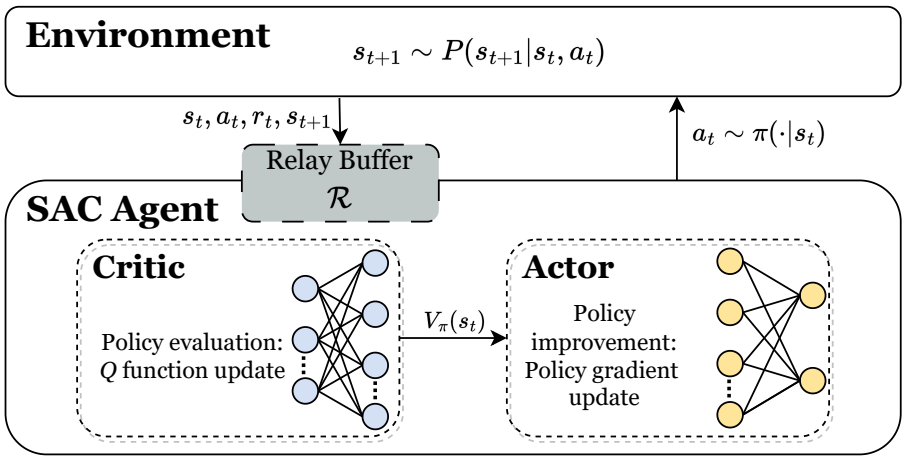


Figure 4.3.1: Schematic of the Soft Actor Critic (SAC) method. Reproduced from Ji et al.[17]

$\mathbf{s}_{t+1} \in \mathcal{S}$ upon executing action $\mathbf{a}_t \in \mathcal{A}$. This function encapsulates the dynamics of the environment.

- Following the execution of action $\mathbf{a}_t$ within state $\mathbf{s}_t$, the agent is rewarded with an immediate feedback $r_t := r(\mathbf{s}_t, \mathbf{a}_t)$. The reward function $r : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ quantifies the desirability of an action within a given state.

The central objective of SAC is to derive a policy that maximizes the cumulative expected reward while also taking into account the entropy of the policy distribution. Specifically, if X is a random variable and its probability density function is p, then its entropy $H(X)$ is defined as $H(X) = \mathbb{E}_{x\sim p}[-\ln p(x)]$, then the degree of stochasticity of a policy π in a state $\mathbf{s}_t$ could be represented as $H(\pi(\cdot|\mathbf{s}_t)) = \mathbb{E}_\pi[-\ln \pi(\mathbf{a}_t|\mathbf{s}_t)]$. Policy entropy measures the randomness of the actions available to the agent, with higher entropy values indicating greater uncertainty. By including the entropy value in the objective function, exploration during training is encouraged, mitigating the risk of converging to a locally optimal policy. A non-deterministic control policy π provides the conditional probability density $\pi(\mathbf{a}_t|\mathbf{s}_t)$ of taking action $\mathbf{a}_t$ in a known state $\mathbf{s}_t$. The goal is to find a stochastic policy that maximizes the sum of rewards and entropy.

According to this Soft Bellman equation[25], the process of Soft Policy Iteration can ultimately converge to the Soft Q function associated with policy π. Then in the policy improvement step, the policy is updated towards the exponential of the new soft Q function. The iterative process of alternating between Soft policy evaluation and Soft policy improvement allows the final policy to converge to the optimal policy, as dictated by the objectives of maximum entropy reinforcement learning. In cases involving continuous spaces, it becomes necessary to approximate these iterations by parameterizing the Q function and the policy π. In the maximum entropy RL, the problem can be defined to find the optimal policy π^* each visited

state, with the form of

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}[\sum_{t} r(\mathbf{s}_t, \mathbf{a}_t) + \alpha H(\pi(\cdot|\mathbf{s}_t))] \tag{4.6}$$

where α is the entropy regularization coefficient that explicitly controls the trade-off between exploration and exploitation. It determines the relative importance between the entropy term against the reward, and a higher α corresponds to higher exploration.

In Soft Policy Iteration, the Soft Bellman equation plays a pivotal role in evaluating and improving the control policy. This equation calculates the expected reward for taking an action $\mathbf{a}_t$ in a state $\mathbf{s}_t$ and then following the policy π with a discount factor γ. This Q function, which assesses the quality of a control policy, can be expressed as:

$$Q_{\pi}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\pi}[V(\mathbf{s}_{t+1})] \tag{4.7}$$

Next, the value function $V_{\pi}(\mathbf{s}_t)$ is defined to represent the expected cumulative reward when following policy π from state $\mathbf{s}_t$. It takes into account the Soft Q function and incorporates an entropy term with a temerature parameter α:

$$V_{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi}[Q_{\pi}(\mathbf{s}_t, \mathbf{a}_t)] + H(\pi(\cdot|\mathbf{s}_t)) = \mathbb{E}_{\mathbf{a}_t \sim \pi}[Q_{\pi}(\mathbf{s}_t, \mathbf{a}_t) - \alpha \ln \pi(\mathbf{a}_t|\mathbf{s}_t)] \tag{4.8}$$

For environments in continuous state and action spaces, the SAC algorithm employs Gaussian mixtures with mean and variance through the neural networks to model both the soft Q-function critic and the policy actor. Specifically, two action-value Q functions are modeled, each parameterized by ω_1 and ω_2 and a policy function π, parameterized by ϕ. SAC follows the concept of Double Deep Q-Network (DQN), using two networks and selecting the one with the smaller value to mitigate value overestimation issues. Both networks provide independent estimations of the value function through $Q_1(\mathbf{s}_t, \mathbf{a}_t)$ and $Q_2(\mathbf{s}_t, \mathbf{a}_t)$. Additionally, corresponding target networks, $Q_{\omega_1}(\mathbf{s}_t, \mathbf{a}_t)$ and $Q_{\omega_2}(\mathbf{s}_t, \mathbf{a}_t)$, are created to enhance optimization stability. During the update of both the Q value function and the policy function, the minimum of the two target Q functions is used for temporal difference target calculation. The loss function for the Q functions, denoted as $L_Q(\omega)$, aims to minimize the soft Bellman residual:

$$\begin{aligned} L_Q(\omega) &= \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) \sim R}\left[\frac{1}{2}\left(Q_{\omega}(\mathbf{s}_t, \mathbf{a}_t) - (r_t + \gamma V_{\omega}^{-}(\mathbf{s}_{t+1}))\right)^2\right] \\ &= \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) \sim R,\ \mathbf{a}_{t+1} \sim \pi_{\phi}(\cdot|\mathbf{s}_{t+1})}\left[\frac{1}{2}\left(Q_{\omega}(\mathbf{s}_t, \mathbf{a}_t) - (r_t + \gamma(\min_{j=1,2} Q_{\omega_j}(\mathbf{s}_t, \mathbf{a}_t) - \alpha \ln \pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}))\right)^2\right] \end{aligned} \tag{4.9}$$

This loss function incorporates the temporal difference error, accounting for the difference between the predicted Q-value and the target value, which is a combination of the immediate

reward and the estimated value of the next state. Additionally, the policy entropy term is included in the loss function. The loss function of the policy π is obtained from the Kullback-Leibler divergence, which is simplified as:

$$L_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim R, \mathbf{a}_t \sim \pi_\phi}[\alpha \ln(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)) - Q_\omega(\mathbf{s}_t, \mathbf{a}_t)]$$

The convergence and optimality of this approach have been rigorously established in previous works[25, 27].

To adaptively adjust the entropy regularization term, SAC reformulates the RL objective as an optimization problem with constraints. The aim is to maximize the expected return while ensuring that the mean entropy remains above a specified threshold H_0:

$$\max_\pi \mathbb{E}_\pi \left[\sum_t (\mathbf{s}_t, \mathbf{a}_t) \right] \; s.t. \; \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi}[-\ln(\pi_t(\mathbf{a}_t|\mathbf{s}_t))] \geq H_0 \; \forall t$$

During the training process, the temperature parameter α is dynamically updated based on the entropy of the current policy, using past experiences. Initially set to 1, α gradually decreases to 0 as the training progresses. That is, maximising the expected return while constraining the mean value of entropy to be greater than H_0, the α will be updated by the loss function:

$$L_\pi(\alpha) = \mathbb{E}_{\mathbf{s}_t \sim R, \mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)}[-\alpha \ln \pi(\mathbf{a}_t|\mathbf{s}_t) - \alpha H_0]$$

This adjustment reduces the emphasis on exploration when evaluating the value function. When the policy's entropy falls below the target value H_0, the training objective increases α, giving more weight to the policy entropy term and encouraging exploration. Conversely, when the policy's entropy exceeds H_0, α decreases, focusing the policy training more on value enhancement.

In Figure 4.3.1, an overview of this approach to learn a walking gait controller for a quadruped robot using the SAC algorithm is presented. The process begins with initialization, followed by the actor selecting an action $\mathbf{a}_t$ at each training step based on the observation $\mathbf{s}_t$, guided by the policy $\pi(\cdot|\mathbf{s}_t)$. The selected action is executed, allowing the robot to interact with the environment and receive a reward r_t along with the next observation $\mathbf{s}_{t+1}$. These experiences, denoted as $(\mathbf{a}_t, \mathbf{s}_t, r_t, \mathbf{s}_{t+1})$, are stored in the replay buffer R and serve as training data for updating the neural network parameters. The agent periodically updates the parameters in the Q-value network, alternating between the two networks to mitigate value overestimation. Note that the actor–critic connection is represented in the update of the actor policy network, where its update utilizes the minimum of soft state value from target Q networks based on Equation 4.8. The policy actor network and temperature parameter are also updated through stochastic gradient descent, minimizing the residual across batch samples from the replay buffer $\mathcal{R}$. The

update process considers the minimum of soft state values from target Q networks, as described earlier. The learning rate λ guides the parameter updates in the descent direction. This iterative process of interaction and learning continues until the gait controller converges to an optimal policy or reaches a predefined maximum number of episodes. To obtain a more comprehensive and in-depth mathematical elucidation, one is advised to consult the primary research articles authored by Haarnoja et al.[25, 27]. This algorithm effectively addresses the challenges of continuous state and action spaces, providing a framework for maximum entropy reinforcement learning in such environments.

4.3.2 Agent Specifications and Reward

In the context of reinforcement learning, the definition of state, action, and reward constitutes a critical phase. This section provides an in-depth exposition of these fundamental aspects.

State space

The first consideration revolves around the definition of the state space, which is defined as the available sensor measurements from the robot. As discussed before, it includes the robot's orientation $\boldsymbol{\theta}(t) = [\theta_x(t), \theta_y(t), \theta_z(t)] \in \mathbb{R}^3$ with regards to three dimensions, specifically roll, pitch, yaw; the moving velocities $\mathbf{v}(t) = [v_x(t), v_y(t), v_z(t)] \in \mathbb{R}^3$ along three principal axes, x, y and z; and the normalized contact force $\mathbf{f}_n(t) = [f_{nFL}(t), f_{nFR}(t), f_{nRR}(t), f_{nRL}(t)] \in \mathbb{R}^4$ between each foot and ground. The above 10 dimensional measurements provide a comprehensive perspective on the robot's surroundings, enabling us to infer its internal state accurately. Additional insights into the nature and characteristics of these sensor measurements are available in Chapter 3. In addition, due to the computation and communication delays in real-world implementations, these temporal disparities introduce asynchrony between the observation of the environment and the execution of control actions. Notably, this asynchrony challenges the fundamental assumptions underpinning the Markov Decision Process, potentially resulting in substantial performance degradation, particularly in robot locomotion tasks. To address this challenge and ensure the effective modeling of the system's dynamics, a holistic approach is adopted. Specifically, the state representation is augmented by incorporating the action executed during the previous time step, denoted as $\mathbf{a}_{t-1} \in \mathbb{R}^4$. This augmentation empowers the agent to consider the temporal dependencies of its actions and their influence on the current state. Consequently, the state vector, denoted as $\mathbf{s}_t$, assumes an expanded form:

$$\mathbf{s}_t = [\boldsymbol{\theta}(t), \mathbf{v}(t), \mathbf{f}_n(t), \mathbf{a}_{t-1}] \in \mathbb{R}^{14}$$

This improved state representation is expected to provide the agent with a better ability to model the dynamics of its environment effectively. Furthermore, it enables the agent to grapple with the complexities arising from hardware latency in real-time robotic applications, ultimately

advancing the pursuit of optimal gait control.

Action space

As previously elucidated, the locomotion of the robot is intricately governed by an inverse kinematics model, elegantly captured by Equation 3.1. In this context, the gait could be meticulously designed to propel the robot forward, with a particular parameterization on diagonal leg pairs. Consequently, the action space is framed around the articulation of these diagonal leg pairs. In precise terms, the action space is defined by two key components, desired bending angles (α_{b_1} , α_{b_2}) of each diagonal leg pairs and compressed leg length (z_{l_1} , z_{l_2}) of each diagonal leg pairs, that is $\mathbf{a}_t = [\alpha_{b_1}, z_{l_1}, \alpha_{b_2}, z_{l_2}]$. For faster gradient descent, the actor output is normalized between 0 and 1 based on the action spaces derived from the simulation dataset. Thus, the action space is defined as a continuous 4-dimensional space, and a single action $\mathbf{a}_t$ is a vector containing 4 desired motor positions after normalization, i.e., $\mathbf{a}_t \in [0, 1]^4$ Subsequently, the policy network output is transformed using a mapping to reference leg locomotion: $\alpha_{b_{ref}} = 2\alpha_{b_{gain}}(\alpha_b(t) - 0.5)$ for bending angle, and $z_{l_{ref}} = z_{l_{gain}} z_l(t)$ for compressed leg length. These transformations allow for the conversion of the actor's output into meaningful references for leg locomotion, thereby enabling the robot to execute the desired gait effectively.

Reward

The primary objective in this study is to encourage the robot maintain a straight and stable gait over extended periods. Hence, the objective reward function is formulated and defined as:

$$r(\mathbf{s}_t, \mathbf{a}_t) = \epsilon_1 \frac{T_s}{T_f} + (1 - \epsilon_2 |v_x(t) - v_{ref}|) - \epsilon_3 \|\ddot{\mathbf{a}}_t\| - \epsilon_4 \|\mathbf{a}_t - \sigma_{threshold}\| - \epsilon_5 (\mathbf{a}_t - \frac{\sum_{i=1}^{H} \mathbf{a}_i}{H})^2 \quad (4.10)$$

In each episode, the control steps continue until the robot falls or the simulation fails. The criteria of failing are identical to that of simulation termination as previously defined in Section 3.1.2. At each training step, the robot is rewarded with a constant value $\epsilon_1 \frac{T_s}{T_f}$ for maintaining balance without triggering any terminal conditions. Here T_s denotes the sampling time of the agent's interaction with the simulation environment, and T_f is the final time of this training in one episode. Instead of applying significant penalties upon episode termination due to undesired states, continuous rewards at each step for the robot's longevity in the task are provided. The accumulated rewards for this term are proportional to the sequence length, effectively motivating the agent to remain upright for extended duration.

As a walking gait controller, the agent is trained to encourage the robot to consistently move in close alignment with a reference speed v_{ref} along the x-axis. To achieve this, a reward component that regulates the forward velocity $v_x(t)$. This component utilizes a shifted absolute

value function, with ϵ_2 governing the shape of the reward function. The essential idea is to stimulate the robot to maintain forward motion with a positive speed while discouraging backward movement (negative speed). Therefore, the choice of ϵ_2 is dependent on the reference speed v_{ref}, with a larger magnitude of ϵ_2 favored when v_{ref} decreases. This sharpens the reward decay near v_{ref} and penalizes negative velocities. For simplicity, we set $\epsilon_2 = -1/v_{ref}$ to ensure that the maximum reward is unity, and negative speeds are penalized.

To promote stable action on motors, a penalty on the large action acceleration $\ddot{\mathbf{a}}_t$ quantified by ϵ_3 is introduced. This penalty discourages abrupt changes in motor actions, aiming to deliver a stable actuation signal to the motors. The penalty acceleration value is estimated using finite differences of actions from the last three time steps:

$$\ddot{\mathbf{a}}_t = \frac{\mathbf{a}_t + \mathbf{a}_{t-2} - 2\mathbf{a}_{t-1}}{T_s^2}$$

Note that $\mathbf{a}_{t-1}$ and $\mathbf{a}_{t-2}$ are not included in the state space at the time step t, but is recorded in the robot's memory.

In addition, to ensure stable walking gait control, the excessive bending angles α_b of the robot's leg pairs are penalized, where negative rewards are assigned as $-\epsilon_4\|\mathbf{a}_t - \sigma_{threshold}\|$, serving as a discouragement for excessive leg bending. $\sigma_{threshold}$ is the threshold for bending angle to prevent simultaneous bending of all four legs. However, it was observed that penalizing leg pairs for excessive bending alone was insufficient, as the robot consistently tended to bend its legs in one direction. Therefore, an additional penalty term is introduced, which is $-\epsilon_5(\mathbf{a}_t - \frac{\sum_{i=1}^{H}\mathbf{a}_i}{H})^2$. This term penalizes leg pairs that consistently bend in the same direction over a horizon H. In this study, the horizon H is set to 16 time steps based on experimental results. The weighting factor $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5]$ objectively defines the emphasis placed on each aspect of the reward function. These values have been thoughtfully selected as $[5, -1/v_{ref}, 0.25, 10, 3]$, aligning with the objective of training the robot effectively.

4.3.3 Training Setup

As previously mentioned, two Q-value critic networks are employed, each complemented by its corresponding target critic network. Their primary purpose is to estimate the Q function as defined in Eq. 4.7. These critic networks share a uniform architecture, characterized by two distinct input components: observations and actions. The observation section undergoes processing through two fully-connected layers, each comprising 128 units. Meanwhile, the action component is managed by a single fully-connected layer with 128 units. The outputs from both sections are concatenated and subsequently channeled through another fully-connected layer featuring 32 units, with the activation function being ReLU. The ultimate output of the Q network serves as an estimation of the Q function.

Concurrently, the actor network utilizes the environment observation as its input. This input is processed through two fully-connected layers, each containing 256 units, and employs ReLU as the activation function. The core responsibility of the actor network is to predict the mean and variance for each action through Gaussian distributions. To ensure that the generated actions fall within the [0, 1] range, the hyperbolic tangent function (tanh) is employed as a squashing function.

To mitigate the risk of overfitting, the learning rates for both the critic and actor networks are set at 0.002 and 0.001, respectively. The learning rate for the temperature parameter α is kept consistent with the network learning rates, fixed at 0.001. For the sake of stability during updates, the update frequency of the policy network is reduced, with parameters being updated every 3 steps in the simulation. The discount factor γ is firmly set at 0.96, as the number of complete steps in a training episode is 50. Furthermore, the target entropy is defined based on the number of actions, with a specific value of $H' = -4$. The batch size is set to 4096 (2^{12}), and the size of the replay buffer is 16384 (2^{14}). To ensure uniformity in environment initialization, all episodes begin from the same initial state $\mathbf{s}_0$.

4.4 Control Architecture Design

The control architecture design of the soft quadruped robot is a critical component in achieving stable and efficient locomotion. Unlike the previous study lacking organized control structures, this study has introduced a structured control system. In essence, the control architecture of the soft quadruped robot represents a harmonious synthesis of hardware, software, and sensor integration. This design enables stable and efficient locomotion, with servo motors, the STM32 microcontroller, and seamless communication underpinning precise control. Additionally, sensory observations gathered through a diverse array of sensors empower the robot with heightened environmental perception and adaptability. The Simulink controller, driven by observations and control commands, ensures the robot's actions align with its intended behaviors, ultimately culminating in stable and efficient locomotion. The control architecture of SoftQ is visually depicted in Figure 4.4.1.

The locomotion of SoftQ is profoundly governed by servo motors, which receive PWM signals from the STM32 microcontroller. These signals translate into specific rotational torques applied to the actuators, leading to controlled limb motions for different gaits. The servo motors feature built-in position controllers that respond to PWM signals, ensuring precise motor positioning.

The STM32 microcontroller manages contact forces and controls the servo motors through General-Purpose Input/Output (GPIO) pins. It efficiently processes commands from the Raspberry Pi, generating PWM signals in response to Simulink controller commands. This

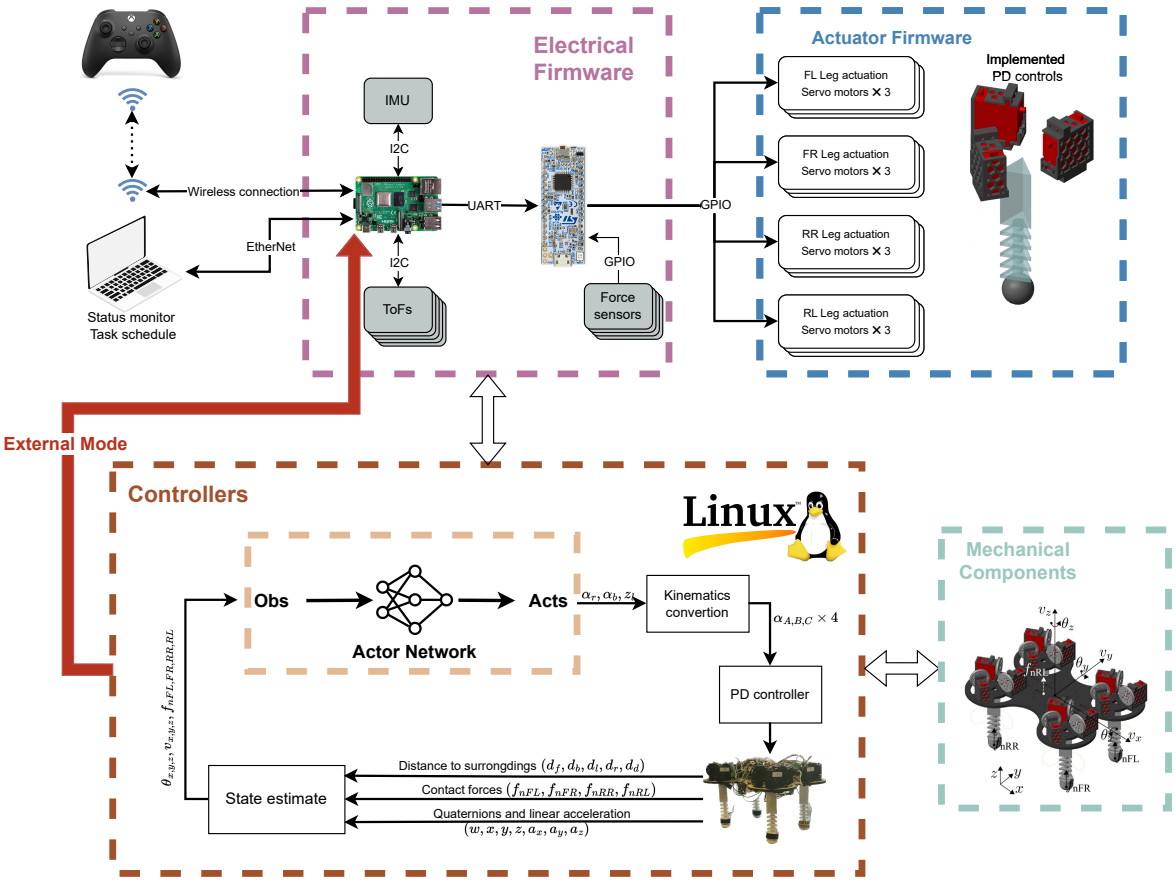


Figure 4.4.1: Graphical overview of control architecture of SoftQ.

fine-grained control guarantees accurate execution of the robot's intended actions. Additionally, the STM32 interfaces with force sensors, reading voltage values across them. These voltage changes help the controller determine the applied forces' magnitude, serving as feedback for the robot to adapt its movements to external forces. Communication between the Raspberry Pi and STM32 utilizes Universal Asynchronous Receiver-Transmitter (UART) for command and data exchange.

The Raspberry Pi directly interfaces with a variety of sensors and devices to gather supplementary information for guiding the robot's control. These vital observations encompass quaternion angles, linear accelerations, and distance data, all of which are acquired through the Inter-Integrated Circuit (I²C) interface. The quaternion angles, originating from the IMU, provide a representation of the robot's orientation in three-dimensional space. For a detailed description of the conversion process from quaternion to Euler angles, please refer to Section A.1. To enhance the situational awareness, these quaternion angles are converted into Euler angles, specifically roll, pitch, and yaw. The IMU itself is a sophisticated 9-DOF sensor, featuring integrated sensor fusion capabilities via its onboard microcontroller. This functionality allows the IMU to provide additional fused data, including linear accelerations in three axis. The linear accelerations is subsequently integrated with distance data from ToF

sensors to estimate the robot's actual velocity. ToF sensors employ the time-of-flight principle by emitting light pulses that bounce off objects, returning to the sensor to calculate the mean distance within a square area in front of it, measuring the distance between the sensor and the reflecting object.

In the context described, the Simulink controller plays a crucial role by assuming the task of generating the robot's actions through external mode operation on the Raspberry Pi. This entails the generation of control commands, which are determined based on the assessment of states inferred from direct observations. To facilitate this process, raw data such as orientation, linear accelerations, and contact forces of the robot are subjected to preprocessing, transforming them into states ($\mathbf{s}_t$) that can serve as inputs for the actor network. The actor network, which is derived from a converged SAC algorithm, serves as the core component in this control scheme. The actor network takes on the critical task of planning an optimal gait, generating actions that enable the robot to move forward effectively. However, it is essential to note that these actions, produced by the actor network, must undergo a further transformation before they can be effectively utilized by the lower-level controllers. This transformation step involves the conversion of the actions into motor commands. This conversion process relies on inverse kinematic relations, as outlined in Equation 3.1. This ensures that the actions generated by the actor network are translated into commands that directly influence the robot's motors, thereby enabling the desired movements and control of the robot's behavior.

Furthermore, the Raspberry Pi establishes wireless communication with a PC to enhance its functionality. The PC serves as a conduit for transferring MATLAB Simulink programs to the Raspberry Pi via external mode. It also acts as a Graphical User Interface (GUI), enabling seamless human-robot interaction. The GUI offers real-time status updates, providing immediate insights into the robot's performance and operation. Moreover, it facilitates the reception of tasks and instructions from the operator, allowing real-time directive issuance. This bidirectional communication between the PC and the robot enhances versatility and adaptability. Additionally, the control system accommodates user input, offering various interfaces like joysticks or command line velocity inputs for real-time robot guidance and adjustments.

4.4.1 Task Planning and Execution

The control architecture of the soft quadruped robot encompasses a multifaceted orchestration of task planning and execution, which plays a pivotal role in the robot's ability to achieve its goals with precision and adaptability. This section delves into the specific architecture of task planning and execution at software side within the control framework.

The core of the robot's control architecture is centered around the efficient utilization of multi-threading. Each thread is responsible for a specific aspect of the robot's operation, and these threads run concurrently, allowing for parallel processing and efficient resource management.

This concurrent execution is particularly advantageous in enhancing the robot's ability to perform multiple tasks simultaneously. Notably, the Raspberry Pi serves as the primary processing unit responsible for coordinating the robot's actions. Within this Raspberry Pi-based control unit, four distinct threads operate in unison to orchestrate the robot's functions:

- UDP_Transmit Thread (20 Hz): This thread is responsible for reading, composing and transmitting observational data to the Simulink environment via User Datagram Protocol (UDP). It facilitates the seamless exchange of sensory information gathered by the robot with the Simulink simulation. The frequency is bounded by the I²C communication of 5 ToFs.

- UDP_Receive Thread (50 Hz): The UDP_Receive thread specializes in receiving motor control commands from the Simulink environment via UDP communication. It ensures that the robot can receive real-time instructions from the simulation, allowing for dynamic control adjustments. Importantly, this thread operates conditionally, only receiving commands when a specific counter condition is met, contributing to safety and controlled execution.

- com2STM32 Thread (20 Hz): This thread is responsible for bidirectional communication with the STM32 microcontroller, which interfaces with the robot's hardware components. It sends relevant control data to the STM32 unit, enabling the robot to perform physical actions. Simultaneously, it receives sensor readings and feedback from the STM32, providing crucial information about the robot's state and surroundings. The frequency is bounded by the computation load from Simulink.

- Countdown Thread (0.5 Hz): The Internal Countdown thread operates at a lower frequency of 0.5 Hz. Its primary role is to manage a countdown mechanism that regulates the activation of flags to the STM32, indicating when motor commands received from Simulink can be executed.

One fundamental aspect of this architecture is its decentralized nature. Instead of relying on a single central controller, the robot employs a distributed control strategy. Each thread is designed to handle a specific subset of tasks or functionalities. For instance, the STM32 process operates on an event-triggered basis, only executing motor commands when received from the Raspberry Pi. It also sends back contact force readings, adding a feedback loop to the control process. Additionally, the presence of flags controlling the switch of power sources ensures that only reliable motor commands obtained from Simulink are executed, further enhancing the safety and reliability of the robot's actions.

To facilitate cooperation among these threads and processes, a well-defined mechanism for multi-core communication and data sharing is established. This mechanism allows threads to exchange information and synchronize their actions. The schematic representation

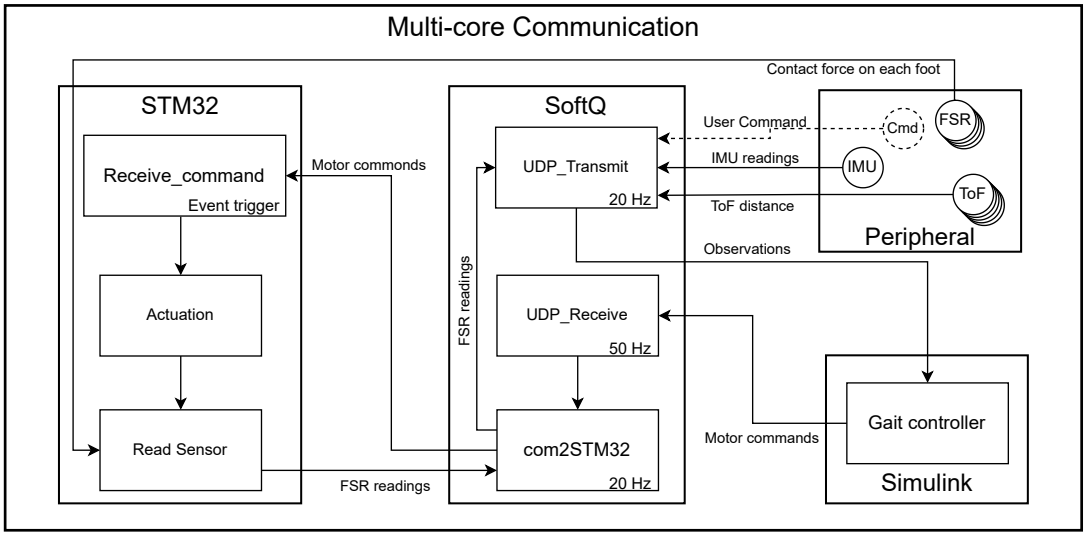


Figure 4.4.2: Schematics of multi-core communication of SoftQ.

of this multi-core communication is illustrated in Figure 4.4.2. To exemplify this inter-thread communication, consider the SoftQ process running on the Raspberry Pi. Within this context, a ”Thread-Safe Data Access” mechanism is meticulously implemented. This involves encapsulating shared data within a dataclass, a structured approach that helps prevent data corruption or conflicts in a multi-threaded environment. Furthermore, synchronization mechanisms such as locks or semaphores are judiciously employed. These synchronization tools serve as safeguards, ensuring that only one thread can access and modify shared data at any given time, thus mitigating the risk of data inconsistency or errors. In addition, the control architecture benefits from pre-designed Direct Memory Access (DMA) and Interrupts Muralidharan et al.[13] and Danelia et al.[14]. These advanced techniques enhance data transfer efficiency and response times, further augmenting the control system’s capabilities.

4.4.2 Sensor Fusion by Kalman Filter

In this section, the focus shifts to the implementation of a Kalman filter for sensor fusion, a critical element of the research aimed at acquiring precise velocity information. The quest for accurate velocity data is driven by the core objective of achieving precise control and adaptability in the deployment of MBRL on physical robots. Velocity estimation is approached by differentiating the position component over time, with particular attention given to forward walking scenarios, where the angular velocity component can be safely disregarded. The instant velocity calculated based on the distance data from the ToF sensors is plotted in Figure 4.4.3. However, this velocity exhibits a sharp changes and may not effectively capture dynamic changes in velocity. Furthermore, the computed velocity may result from the integration of linear acceleration over time, as plotted in Figure 4.4.3. This approach, unfortunately, is susceptible to inaccuracies, with the calculated velocity exhibiting constant fluctuations even

in the absence of motion. The source of this inaccuracy is attributed to the drift present in the acceleration data acquired from the IMU. Consequently, the velocity estimation from acceleration reaches a constant value at -0.8 m/s after walk.

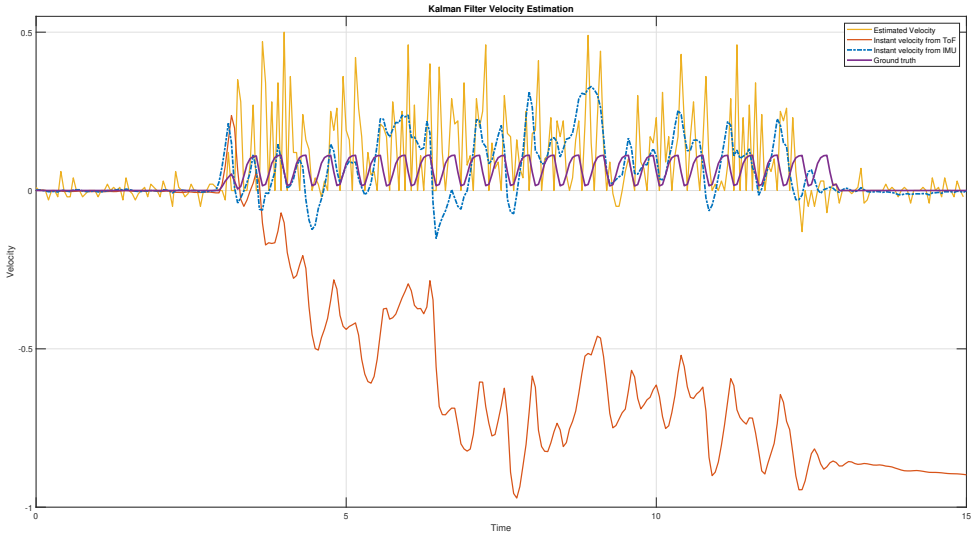


Figure 4.4.3: Velocity estimation in x-axis (v_x) through Kalman filtering of ToF and IMU data.

To achieve a more accurate and precise velocity estimation, a Kalman filter has been specifically designed and integrated into the control architecture. The Kalman filter assumes a crucial role in reducing sensor electronic noise, addressing measurement inaccuracies, mitigating sensor noise, and minimizing quantization errors when predicting velocity from acceleration and position measurements. However, it is important to note that obtaining ground truth velocity data from the real robot is complex. Therefore, we used velocity data from simulations of the tested expert gait as a reference, as shown in Figure 4.4.3. The Kalman filter's estimation appears to capture most of the movement periods, although some inconsistencies remain. This is attributed to the reality gap, as the robot faces challenges in maintaining a relatively constant speed due to differences in ground conditions and other factors in the real-world environment.

Kalman filter is a linear optimal states estimation method, which is known as one of the most famous Bayesian filter theories. State equation is a linear representation of a state vector x_{k-1}, a process noise matrix w_k and a control input vector u_k. Observation equation is a linear representation of a state vector x_k and a measurement noise vector v_k. A dynamic model is presented with state equation and observation equation through the reliable estimation corrected by measurements, which is shown in Equation 4.11.

$$\begin{aligned} \text{State Equation:} \quad & x_k = Ax_{k-1} + Bu_{k-1} + w_k \\ \text{Observation Equation:} \quad & z_k = Hx_k + \nu_k \end{aligned} \tag{4.11}$$

In the above formulas: A, H, w_k, ν_k is the state transition matrix, the observation matrix, the system noise vector, the observation noise vector, respectively. w_k and ν_k are assumed to satisfy positive definite, symmetric and uncorrelated, zero mean Gaussian white noise vector. w_k is a random variable with a multivariate normal distribution, where the covariance matrix is Q. ν_k is also assumed to be a random variable with a multivariate normal distribution, where the covariance matrix is R. In addition, The state covariance matrix, denoted as P, is a symmetric matrix that provides information about the uncertainty in the state variables being estimated by the Kalman filter. P undergoes continuous updates as the Kalman filter processes new measurements and predictions. In specific, the Kalman filter is appled to a linear system, and the system dynamics can be approximated as a linear relation between the acceleration a, velocity$v$ and position p, namely

$$\begin{cases} p = p + T_s \cdot v + \frac{T_s^2}{2} \cdot a \\ v = v + T_s \cdot a \end{cases}$$

Defining the states as $x_1 = p$, $x_2 = v$ and the input $u = a$, the model could be transformed into state space form

$$\begin{cases} \mathbf{x}_{k+1} = \begin{bmatrix} p \\ v \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}}_{A} \mathbf{x}_k + \underbrace{\begin{bmatrix} \frac{T_s^2}{2} \\ T_s \end{bmatrix}}_{B} u_k \\ y_k = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{H} \mathbf{x}_k \end{cases} \tag{4.12}$$

The Kalman gain K determines the weight given to incoming measurement data when updating the estimated state of a dynamic system. In each step, K_k is updated by P_k, $K_k = \frac{P_k H}{H P_k H^T + R}$. The Kalman gain K_k emerges as a critical factor in adjusting the state estimate in response to incoming measurements, as outlined in the state estimation equation:

$$\hat{\mathbf{x}}_{k+1} = A\hat{\mathbf{x}}_{k-1} + Bu_k + K_k \cdot (z_k - H\hat{\mathbf{x}}_{k-1}) \tag{4.13}$$

In this equations, $\hat{\mathbf{x}}_{k+1}$ is the updated state estimate, u_k is the control input, z_k is the measurement in position. The observer is interpreted as a one-step predictor, which forecasts the system states at the next time instance $k+1$ by utilizing the estimated states at time k, along with additional feedback from the estimation error, weighted by the Kalman gains. Since the disturbance primarily affects the acceleration, we can update the structure of Q accordingly. The

process noise covariance matrix Q should reflect the covariance of disturbances in acceleration (σ_a^2) and how they propagate through the system. In a simplified form, it might look like: $Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$, and the observation noise covariance matrix is set as $R = 0.15$. Additionally, the observation noise covariance matrix is tuned to be $R = 0.1$, and initial state estimates ($\mathbf{x}_0$) and initial error covariance estimates (P_0) are provided as: $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

The detailed process of how the Kalman filter is updated and applied can be found in Section A.2. Before incorporating sensor measurements into the Kalman filter, a preprocessing step was performed to bound and average the measurements, ensuring their reliability and stability. The same Kalman filter for sensor fusion was employed for velocity estimation in both the x and y axes. This choice was driven by the presence of two Time-of-Flight (ToF) sensors dedicated to these axes. Conversely, in the z axis, which represents vertical motion, a single ToF sensor located at the bottom was used. The optimal state observer, known as the Kalman filter provides the optimal estimate of $\mathbf{x}$ minimizing the covariance matrix P of the estimation error $\hat{\mathbf{x}}$ using series of noisy measurements. As the Kalman filter iteratively processed sensor measurements, it progressively refined its estimates of velocity. The results of velocity estimation, facilitated by the Kalman filter application, are visualized and presented in Figure 4.4.3. These plots provide a clear representation of the refined velocity estimates and underscore the effectiveness of the Kalman filter in mitigating sensor noise and enhancing the overall reliability of the velocity sensing.

4.5 Potential Reality Gap

Transitioning to a different facet of consideration, it is crucial to address the concept of the ”Potential Reality Gap.” In the realm of reinforcement learning, the surrogate model serves as a bridge between simulation environments and real-world scenarios. However, due to the inherent complexity and uncertainty associated with real-world environments, a misalignment often arises between the training data collected from simulations and the actual performance of the trained model in real-world situations. This discrepancy is referred to as the ”Reality Gap.”

The Reality Gap poses a challenge because the surrogate model might perform exceptionally well during training in simulation environments but struggle to generalize effectively to real-world settings. This phenomenon can be attributed to various factors, including disparities in physical dynamics, sensor noise, and unmodeled environmental influences. To address the Reality Gap, certain strategies can be adopted. One approach involves introducing stochasticity during training in simulations, simulating uncertainties and disturbances that are characteristic of real-world scenarios, where zero mean Gaussian noise is introduced into all observation

signals. This additional noise is independently sampled from zero-mean Gaussian distributions in a random manner. Based on the sensor calibration results, the variances of the noise applied to the velocity, angular and force signal data are set to: $\sigma^2_{\boldsymbol{\theta}} = 0.002$, $\sigma^2_{\mathbf{v}} = 0.002$ and $\sigma^2_{\mathbf{f}_n} = 0.005$, respectively. By incorporating such noise into the training process, the model is better equipped to develop robustness and adaptability, allowing it to effectively handle unexpected variations and challenges encountered in real-world applications.

In an effort to mitigate the Reality Gap, it is worth noting that the SAC algorithm offers commendable exploration capabilities. This presents the opportunity to retrain a converged agent with the potential to adapt its policy for various tasks. Consequently, the converged policy guiding the robot's walking behavior, as determined by MBRL, can be repurposed as the initial policy for continued training with a more accurate model, facilitating the reduction of the Reality Gap, which is so-called continuous training.

The continuous training begins by extracting the policy that has been converged during the initial MBRL training. This policy is the result of the agent's learning process and encapsulates the knowledge it has gained about optimal gait control. To address the reality gap, the converged policy is integrated with a more accurate simulation model in Matlab. This simulation would strive to closely replicate the real-world dynamics and physics of the robot. The use of a more accurate model helps in reducing the discrepancies between simulation and reality. The training process begins with the converged policy as the starting point. This initial policy is influenced by the agent's prior learning, facilitating faster adaptation to the refined simulation environment. Depending on the specific goals and challenges of the new simulation setup, it is important to adjust the reward structure by fine-tuning reward functions to align with desired behavior or objectives in the updated environment. Once the continuous training process has yielded a sufficiently adapted policy, this policy can be transferred to the physical robot for real-world testing.

Chapter 5

Results

This chapter presents the results of the experiments and evaluations conducted in this research. It provides a detailed analysis of the performance of MBRL algorithms with continuous training and provide insights into the research questions posed at the beginning of this study.

5.1 Surrogate Model Performance

5.1.1 Performance Evaluation

To evaluate the surrogate model's performance, it is important to consider several critical metrics. These metrics serve as crucial indicators of how effectively the model approximates and forecasts the robot's behavior. Here, these metrics are elaborated without delving into details:

- Average Steps before Failure: This metric assesses the model's capacity to predict the duration of a simulation run without encountering failure, serving as an indicator of sample efficiency.
- Validation RMSE and Validation Loss: This RMSE quantifies the disparity between the surrogate model's predictions and the actual values present in the validation dataset, as defined in Equation 4.2. Lower RMSE values are indicative of a higher degree of precision and accuracy in forecasting future states (s_{t+1}). Concurrently, the Validation Loss serves as an overarching measure of the model's performance on the validation dataset, underscoring the significance of minimizing this metric to enhance predictive capabilities, as defined in Equation 4.1.
- NRMSE: NRMSE provides a normalized measure of error, enabling comparisons of prediction accuracy across expert pattern datasets. It proves particularly valuable in evaluating the surrogate model's ability to predict sensor observations on the robot. The single step prediction NRMSE is calculated by Equation 4.3, the long-term prediction NRMSE is calculated by Equation 4.4.
- Correlation Coefficient (R): The R quantifies the linear relationship between simulated

and predicted values, as specified in Equation 4.5. A higher R signifies a more robust correlation, highlighting the model's proficiency in capturing underlying data trends and patterns.

5.1.2 Control Variables Restriction

The investigation commences with a focused exploration of state-space restriction within the surrogate model. The state-space, encompassing both observations and actions, is defined as $\mathbf{s}_t = [\boldsymbol{\theta}(t), \mathbf{v}(t), \mathbf{f}n(t), \mathbf{a}_{t-1}]$ in Section 4.3.2. It is worth noting that the first three terms in this state-space originate from observations and are inherently challenging to restrict, but the action space can be subject to constraints. It is crucial to acknowledge that the first three terms in this state-space comes from observations and present inherent challenges for restriction, but the action space can indeed be subjected to constraints.

An important aspect to underscore is that the restriction of the robot's action space was achieved through the application of pattern-defined modeling techniques. Specifically, the robot's legs are interconnected in diagonal pairs, similar to a trotting pattern. This inherent structural arrangement significantly simplifies the action space compared to dealing with the complexity of controlling 12 individual motor actions[17], effectively reducing it from $\mathbf{a} \in \mathbb{R}^{12}$ to $\mathbf{a} \in \mathbb{R}^4$.

Furthermore, an alternate method of parameterization was applied, where the surrogate model's configuration incorporates specific control variables. Within this context, the actions are as defined in Section 4.3.2, taking the form of $\mathbf{a}_t = [\alpha_{b_1}, z_{l_1}, \alpha_{b_2}, z_{l_2}]$. These variables are subject to the bounds governed by the gains ($\alpha_{b_{gain}}, z_{l_{gain}}$). It is also crucial to emphasize that these gains associated with actions hold significance not only in the construction of the surrogate model but also play a vital role in shaping the behavior and adaptability of the SoftQ. Therefore, our focus centers on two key components within the action space: the gain of bending angle $\alpha_{b_{gain}}$ and the gain of compression length $z_{l_{gain}}$ for each of the diagonal leg pairs. To address this aspect, a series of simulations involving random actions within a bounded action space were carried out. The specific bounds for the control variables are α_{b_1}, z_{l_1}, α_{b_2}, and z_{l_2}. Each test included three training sessions of neural networks with the same training data, and each of these networks was evaluated by expert data $\mathcal{D}_{real}$. These simulations aimed to gather the necessary data required for training the surrogate model. Subsequently, the performance of the trained model was evaluated. To effectively assess the trained neural networks performance, a color heat map can be employed to provide a visual representation of various performance metrics, as shown in Figure 5.1.1. Each metric is associated with a specific color scale, allowing for a quick and intuitive evaluation.

In Figure 5.1.1(a), it is evident that reducing the lower restrictions within the action space leads to an increase in the average number of steps the system can take before encountering failure,

indicating improved data efficiency due to reduced locomotion risk with smaller actions. In Figure 5.1.1(c), all neural networks converge with a loss below 0.037 in the validation dataset, with minimal differences between them. In Figure 5.1.1(b), validated in the same dataset, RMSE converges to a minimum level. Notably, better validation RMSE is achieved when α_b ranges from 0.55 to 0.7 radians and z_l ranges from 3 to 9 millimeters, suggesting optimal model performance within these parameter ranges.

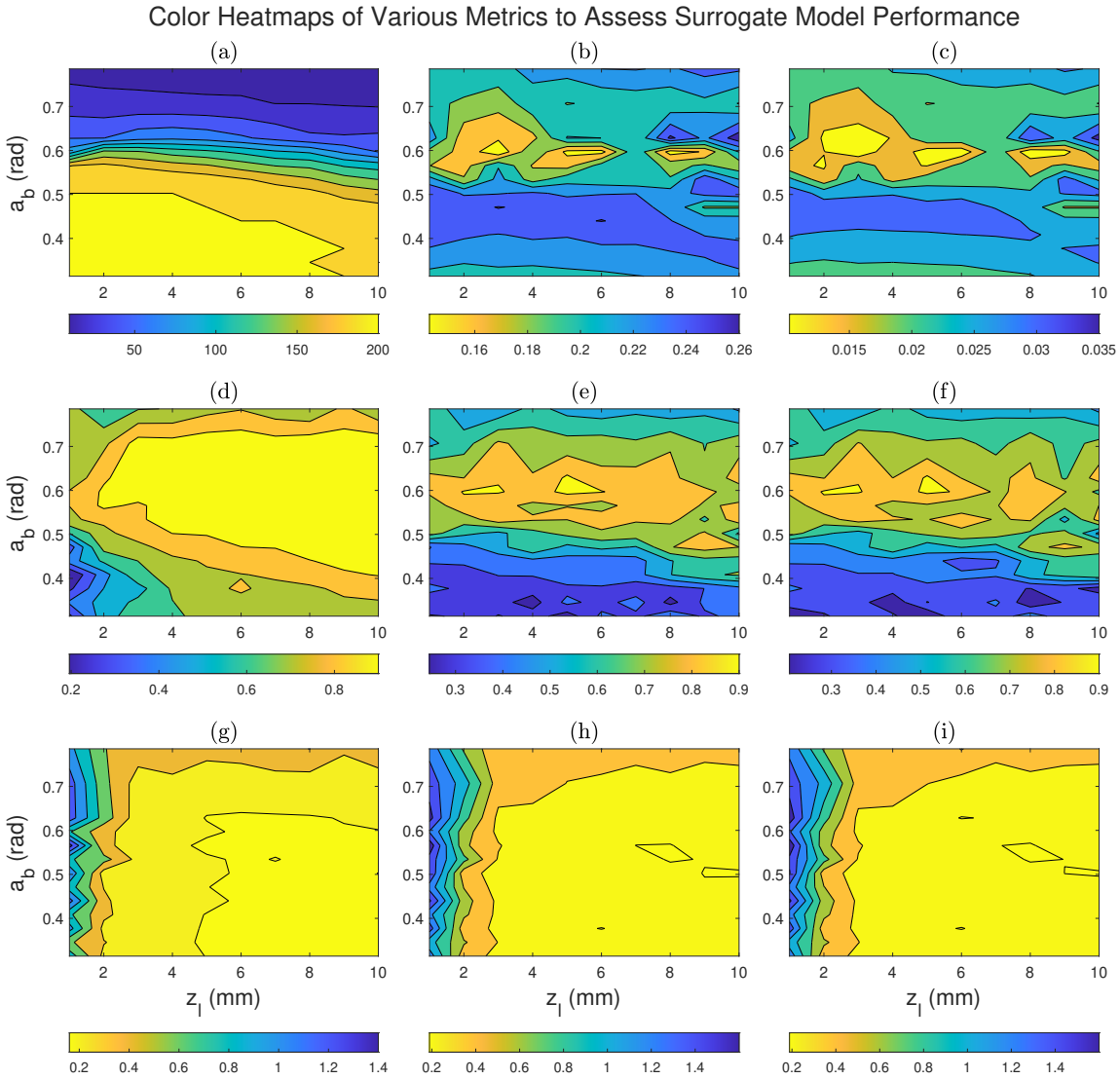


Figure 5.1.1: Visualization of Performance Metrics Through Color Heat Map for Evaluating Trained Surrogate Models. Lighter colors indicate better performance in metrics. The metrics include: (a) Average Steps before Failure; (b) RMSE on the Validation Dataset; (c) Loss on the Validation Dataset; (d) Single Step Prediction Coefficient R; (e) Average Coefficient R for Long-term Prediction; (f) Coefficient R for Full Horizon Prediction; (g) NRMSE for Single Step Prediction; (h) Average NRMSE for Long-term Prediction; (i) NRMSE for Full Horizon Prediction.

In assessing single-step predictions using various expert gait pattern datasets, the correlation coefficient (R) between predictions and ground truth was calculated, as depicted in Figure

5.1.1(d). The results show consistently high accuracy when either the compression length (z_l) or the bending angle (α_b) is high, or when both are elevated. This observation aligns with the expectation that for the robot to achieve higher speeds, it should exhibit more pronounced behaviors, which are characterized by larger values of these parameters. Consequently, the expert gait dataset used in this context emphasizes high values of α_b and z_l, resulting in smaller differences between predictions and ground truth, and consequently, lower values of R. A similar trend is observed when considering single-step predictions. Smaller NRMSE values are associated with high values of z_l and α_b, but interestingly, low α_b values also yield small NRMSE values. This implies that bending angle (α_b) exerts a more significant influence on robot locomotion than z_l. These results are visually represented in Figure 5.1.1(g). However, it is important to note that neither the NRMSE nor R reach their lowest points in the highest α_b region, suggesting poorer predictions when too small steps are taken before simulation failure, resulting in an insufficient amount of useful data.

For multi-step predictions on the same expert gait pattern datasets, the effects of α_b and z_l are similar in single step prediction, but the models with low α_b values perform less effectively in predicting over longer time horizons, suggesting a lack of robust generality in these models. However, as shown in Figure 5.1.1(e), the average correlation coefficient (R) for long-term predictions reveals that models with relatively high values of both α_b and z_l struggle to predict states over extended horizons. This challenge could be attributed to the larger action space, which introduces more unpredictability and subsequently leads to reduced performance compared to models with lower z_l values when trained on the same dataset size. This trend is also evident in the full horizon prediction R, as depicted in Figure 5.1.1(f), where predictions at relatively high values of both α_b and z_l deteriorate. In addition, the average NRMSE of long-term predictions exhibits a similar performance to what was observed in single-step predictions, which shows different patterns compared to R results, because the NMRMSE is normalised and large values like contact forces may have more effects on the metrics. Lower errors are associated with larger values of α_b and z_l, but NRMSE also decreases when α_b reaches its highest values. These results are presented in Figure 5.1.1(h). Interestingly, performance remains consistent when α_b is smaller than 0.72 radians and z_l is larger than 3 millimeters. Likewise, in the case of NRMSE for full horizon prediction, there is no significant change in performance, as indicated in Figure 5.1.1(i).

Hence, the optimal action state restrictions can be identified as approximately α_b around 0.6 rad and z_l around 7 mm. However, it is important to consider that these restrictions also impact MBRL training, as they limit the range of exploration within the robot's action space. Therefore, it is preferable to keep the action space as broad as possible. Another viable set of restrictions to consider is α_b around 0.63 rad and z_l around 8 mm for the state space during surrogate model training and MBRL training. This broader range allows for more exploration and potentially better overall performance in subsequent MBRL learning.

5.2 Model-based RL Training

After the surrogate model determined in Section 4.2, we can continue to training of MBRL for optimal gait controls, which will be used as a gait controller in robot to navigate with learned optimal gait. It is important to note that the training settings remained consistent across all experiments. The code snippet for training details can be found in Section B.1. To account for potential variations due to randomness, the training process was initiated with various random seeds. It was observed that training had a higher failure rate when the agent became trapped in local minima. Consequently, it can be intuitively inferred that mastering the policy for low-speed locomotion posed a more challenging task for the agent. Notably, in the experiments, v_{ref} was maintained at no smaller than 0.1 m/s to ensure meaningful training outcomes and the final time of the training in one episode T_f is 10 seconds.

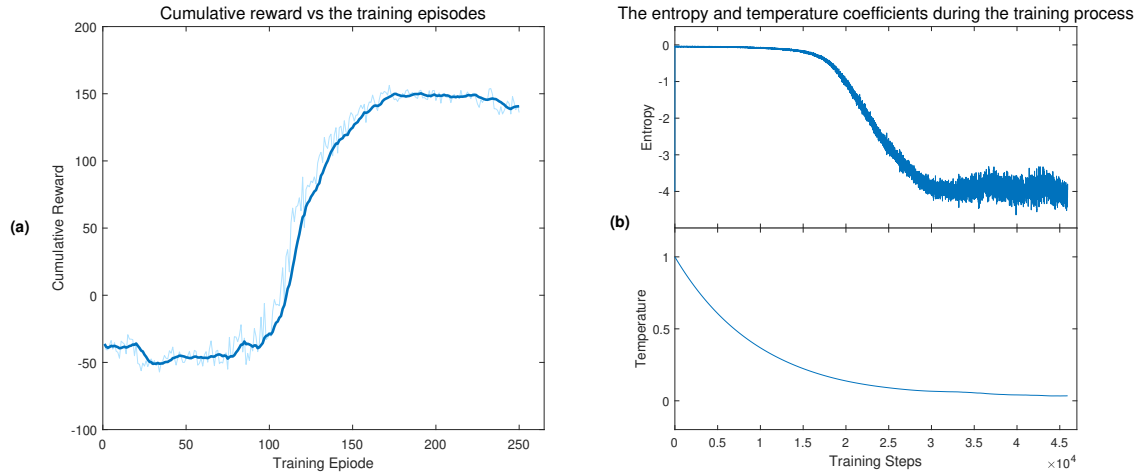


Figure 5.2.1: The Training Results of MBRL. (a) Cumulative reward in MBRL vs the training episodes; (b) The variation of entropy and temperature parameters during the training process.

The training progress, along with the entropy and temperature parameters, is visually represented in Figure 5.2.1. The plot illustrates the successful training of the agent using the SAC algorithm. The cumulative reward steadily increased from approximately -45 to 150, indicating the acquisition of the desired behavior by the agent. Additionally, the entropy decreased to around -4 (with the goal being the negative of the number of actions), and the temperature parameter also exhibited a decline from 1 to 0.05. These trends signify that the agent effectively learned the desired behavior during the training phase. Subsequent to the training process, the trained agent was implemented in simulation to evaluate its performance. The results of this implementation are elucidated in Figure 5.2.2, providing valuable insights into how well the trained MBRL model performs in executing optimal gaits for the SoftQ.

Post-training validation in the simulation environment, conducted for a duration of 5 seconds, yielded noteworthy observations. As depicted in the figure, the robot exhibited some velocity

along the x-axisbut was unable to maintain this velocity for an extended period. However, it remained virtually stationary at its initial position, effectively patrolling within a confined area. Despite generating speed, the robot failed to manifest any significant forward displacement. This phenomenon is notably characterized by a marked increase in the COT during the initial phases of robot motion. This surge in COT can be attributed to the imperative need for the robot to surmount inertia and initiate acceleration as it commences motion. Consequently, a substantial input of energy is necessitated. Additionally, heightened frictional forces encountered by mechanical components, such as wheels and joints, during the initial moments of motion contribute to this increased energy expenditure.

Figure 5.2.2: The Validation Results of MBRL. (a) COT of the robot while walking; (b) Distance traveled along the x, y, and z axis; (c) Speed of the robot along the x, y, and z axis; (d) Orientation of the robot along the x, y, and z axis.

To assess the operational capacity and explore the gait patterns of the soft robot, the same RL configurations were applied to learn optimal gaits under different reference speeds denoted as v_{ref}. Throughout the training experiments, an interesting observation emerged: the training performance of the simulated robot was notably influenced by the reference speed v_{ref}. When the target velocity was set higher, the learning agent was able to achieve forward motion in fewer training episodes, ultimately converging to an effective policy. Conversely, when v_{ref} was set to a lower value, the agent exhibited a tendency to pitch forward, resulting in episode termination. The training process was initiated with various random seeds, and it was observed that training had a higher failure rate when the agent became trapped in local minima. Consequently, it can be intuitively inferred that mastering the policy for low-speed locomotion posed a more challenging task for the agent.

Given the limitations observed in the direct application of the agent trained through MBRL, a decision was made to further train this agent in a model-free environment to bridge the reality gap effectively. It is noteworthy that adjustments were made to the reward structure, particularly in penalizing instances where all four legs bent in the same direction with a significant angle. The coefficient ϵ_4 governing this penalty was increased from 10 to 100. This modification was introduced due to the possibility that the model-based training might generate actions that are fatal in simulation but less perilous in the real-world scenario.

5.2.1 Continue Learning

To accelerate the training process, a predefined expert behavior, denoted as $A^e = [\mathbf{a}^e_{T_0}, ..., \mathbf{a}^e_T]$, is established[17]. This predefined behavior offers the robot an initial stable gait. Specifically, during the initial second of each training episode, the RL agent emulates human knowledge by executing the predefined behavior A^e. By this means, we resort to the framework of *imitate learning*[99]. Data generated from this process is collected and stored in a replay buffer for subsequent learning phases. This approach aligns with the principles of imitation learning, where the RL agent imitates pre-established behaviors as a form of guidance. While it is important to note that this predefined state-action trajectory does not guarantee the optimality, it serves as a valuable reference for the agent. Emulating these guided elementary movements enables the agent to swiftly implement and adapt its behavior in real-time scenarios, contributing to efficient online implementation.

To continue training the agent from MBRL, the same RL configurations were applied, with the exception of the penalty adjustment for the four legs bending. The goal for the robot was to learn an optimal gait for higher speed, with v_{ref} set at 0.3 m/s. The reward functions remained the same as defined in Equation 4.10, but the coefficients were adjusted to $[\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5] = [5, -1/v_{ref}, 0.25, 100, 3]$.

The results of continuous training are visualized in Figure 5.2.3 for comparison with Model-Free Reinforcement Learning (MFRL). Notably, it is observed that the continuous training policy exhibited rapid improvement, reaching convergence at around 200 episodes. However, it is important to note that this policy incurs higher penalties, aligning with the assumption that the MBRL policy may generate riskier actions. Additionally, the MFRL policy converges at approximately 120 episodes, which is slower than the continuous training agent. However, the reward at the beginning of training is higher for MFRL, indicating that the agent in early training may not actively promote forward motion but is associated with fewer high-risk actions. Furthermore, the MFRL approach failed to converge to a walking gait even after 350 episodes, and the MFRL agent did not learn to travel a significant distance. Notably, the MFRL method required about 24 hours of training, while MBRL and continuous training together took approximately 10.5 hours. This substantial reduction in training time, coupled with

the ability to achieve effective learning and improved gait control, underscores the efficiency and effectiveness of the continuous training approach compared to the benchmark MFRL method.

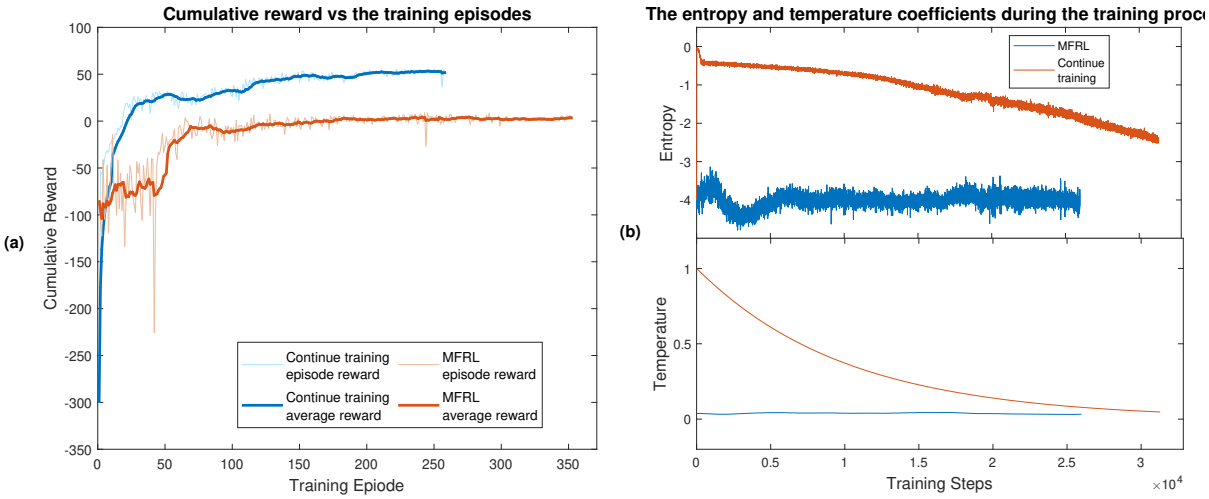


Figure 5.2.3: The Training Results of Continuous Training Compared to MFRL. (a) Cumulative reward in MFRL and continuous training vs the training episodes; (b) The variation of entropy and temperature coefficients during the training process of both MFRL and continuous training.

The outcomes of validations for continuous training are graphically presented in Figure 5.2.4, which serves as a overview of continuous training performance. The COT achieved a remarkable low value of approximately 73 J/kg/m, signifying the robot's ability to traverse a certain distance with efficient energy consumption. However, it is worth noting that at the beginning of movement, the COT exhibited a notable increase, reaching approximately 200 J/kg/m. This initial rise in COT can be attributed to several factors, such as static friction, initiate acceleration, and control and actuator-related challenges during the transition from rest to motion. Despite this initial increase in COT, the robot demonstrated its capability to efficiently attain and sustain a speed higher than the predefined benchmark $v_{x_{avg}}$ = 0.2 m/s, as attached in Appendix Figure A.5.1. Over a 5-second validation period, the robot covered a distance of 1.8 meters, achieving an average speed of 0.36 m/s, surpassing the reference speed v_{ref} of 0.3 m/s. These findings highlight the agent's capability to attain and sustain a speed higher than the predefined benchmark. Moreover, the robot demonstrated stability during locomotion, as indicated by the minor fluctuations in velocity along the x-axis and minimal deviations in the y and z axes. The orientation plots revealed controlled movement, with the largest rotation measuring approximately 0.055 radians in the initial stages, confirming a stable and well-maintained gait. Collectively, these validation results underscore the practical viability of the continuous training technique in enhancing the robot's performance in real-world scenarios.

Figure 5.2.5 (a) shows the examples of the four foot trajectories, and Figure 5.2.5 (b) with colored areas representing the contact regions demonstrates a typical gait patterns captured

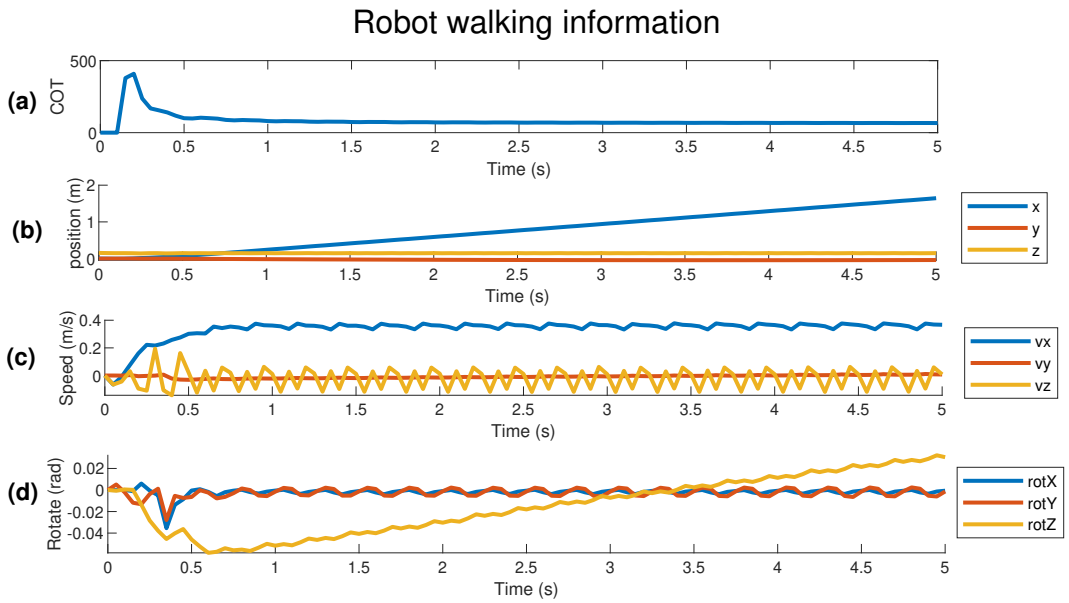


Figure 5.2.4: The Validation Results of Continuous Training. (a) COT of the robot while walking; (b) Distance traveled along the x, y, and z axis; (c) Speed of the robot along the x, y, and z axis; (d) Orientation of the robot along the x, y, and z axis.

from the validation simulation. Synchronized motions were observed between the FR and RL legs, as well as the FL with RR legs, as they are linked as pairs. Notably, the FL and RR leg pairs exhibit a smaller swing area compared to the FR and RL pairs. This difference in swing area may be attributed to corrective actions addressing yaw rotation errors. These yaw rotation errors could potentially be caused by slight uneven mass distribution on the robot, favoring the right side, as indicated by the orientation in the z axis shown in Figure 5.2.4 (d). Additionally, a delay of approximately 0.1 seconds is observed between each diagonal leg pair. The contact time for each leg is around 0.15 seconds, resulting in the formation of a periodic gait pattern. This pattern indicates that the robot successfully learned to coordinate its movements in a rhythmic manner, which is essential for stable and sustained locomotion.

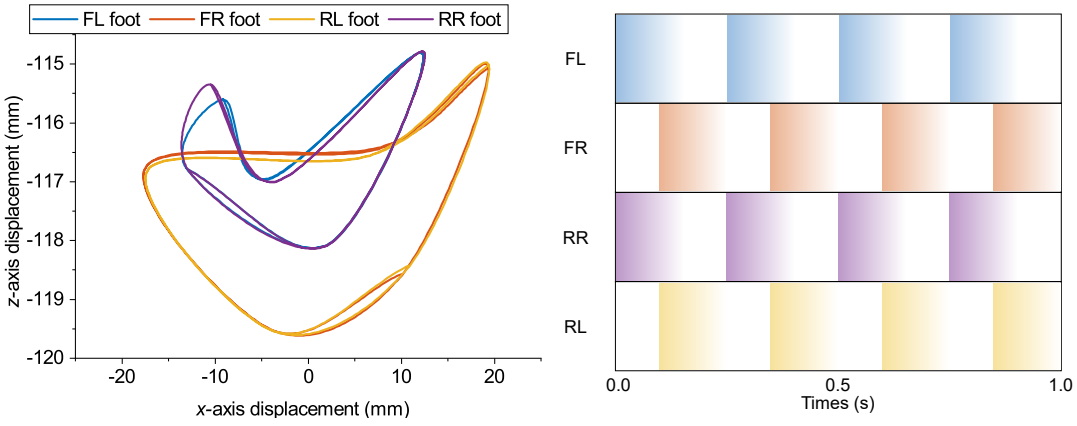


Figure 5.2.5: The Learned gait patterns. (a) Four foot moving trajectories. (b) Footfall patterns

Moreover, it is essential to highlight that the observed results go beyond demonstrating successful gait patterns and stability. They also reveal the practical significance of this achievement in terms of direction control within velocity control. As shown in Figure 5.2.4 (d), in the beginning of walking, the robot initially turns to the right but subsequently corrects itself, adjusting its orientation slightly to the left by the end of the gait cycle. This phenomenon was also observed in field tests, as presented in Section B.3. This capability for self-correction and precise direction control enhances the robot's suitability for applications that require controlled and accurate movement, further expanding its utility in real-world scenarios.

5.3 Model-based RL Comparison

Given that the MBRL with continuous learning reached a superior performance than MFRL in the same experimental setting; however, it is still required more examinations to statistically prove that MBRL with continuous learning suppress MFRL. Therefore, to establish the statistical significance of this observation and confirm that MBRL with continuous learning outperforms MFRL consistently, further comprehensive examinations are required. This section aims to systematically and statistically determine which RL method is more effective for optimally controlling the gait of a soft quadruped robot.

5.3.1 Performance Evaluation

In the performance evaluation of the trained agent, it is important to consider several critical metrics, which have been discussed in Section 3.3. These metrics serve as crucial indicators of proficiency of the agent, referred to as the controller. Here, these metrics are expounded again avoiding excessive details:

- Stability: This metric combines information about the time duration of the gait, rotational motion, and vertical displacement to assess the stability of the gait. It is calculated using Equation 3.3 and provides valuable insights into the agent's ability to maintain stability during operation.
- Resultant walking speed (m/s): This metric involves assessing the agent's walking speed, particularly by calculating the average speed achieved during validation. It offers a straightforward measure of the agent's mobility and effectiveness in movement.
- Cost-of-Transport (J/kg/m): The COT metric evaluates the energy efficiency of the agent. It is defined in Equation 3.4 and quantifies quantifies the amount of energy expended by the agent, normalized by its weight and the distance traveled. This metric is crucial for understanding the agent's energy consumption patterns.
- Learning efficiency: Learning efficiency gauges how quickly the agent converges to an

optimal or satisfactory performance level. It is quantified by measuring the training time required for the agent to reach convergence and the reward reach a significant level. This metric provides insights into the agent's adaptability and learning capabilities.

5.3.2 Compare with MFRL

This section will delve into the performance comparison between the MBRL with continuous training and MFRL based metrics outlined earlier. To ensure a robust evaluation, three identical training sessions were conducted for each reference velocity v_{ref}, which took values of 0.2 m/s, 0.3 m/s, and 0.5 m/s. Training was stopped upon the convergence of each policy to a significant reward level. Following the training phase, each trained agent was implemented and evaluated through simulation of 5s. The results of these evaluations are visually represented in Figure 5.3.1. It is evident from the figure that MBRL with continuous training consistently outperforms MFRL. This conclusion is drawn based on the observation that the mean values of key performance metrics, including stability, resultant walking speed, COT, and training time, are consistently higher for MBRL with continuous training compared to MFRL.

Specifically, in terms of stability, MBRL with continuous training achieved a mean stability level of approximately 0.62, significantly surpassing MFRL, which achieved a mean stability of approximately 0.05. Notably, the level of stability achieved by MBRL with continuous training closely approximated that of the predefined expert gait, indicating its capability to generate highly stable locomotion.

Regarding resultant walking speed, it was noted that MBRL with continuous training exhibited a higher mean walking speed. However, it also displayed a larger variance, partly due to variations in reference walking speeds (v_{ref}).Despite the variance, the mean walking speed achieved by MBRL with continuous training was significantly greater than that of MFRL. It is worth noting that MBRL, in its pursuit of efficient locomotion, tended to focus on walking further, resulting in a mean speed that was significantly higher than the expert gait's speed, even though it initially learned from the expert gait. In contrast, MFRL faced challenges in learning extended walking within 350 training episodes, which is reflected in the mean speed approaching zero. It primarily struggled to adapt and achieve higher speeds or longer distances. This disparity in performance between MBRL with continuous training and MFRL underscores the advantages of the former approach in facilitating both stable and efficient locomotion, even surpassing the performance of the expert gait.

Furthermore, the analysis revealed that MBRL with continuous training achieved an efficient COT level, similar to that of the predefined expert gait, indicating economical energy consumption during locomotion. In contrast, MFRL agents exhibited significantly higher COT values. This difference primarily stemmed from the fact that only a limited number of MFRL agents succeeded in overcoming the initial resistance associated with initiating longer-distance

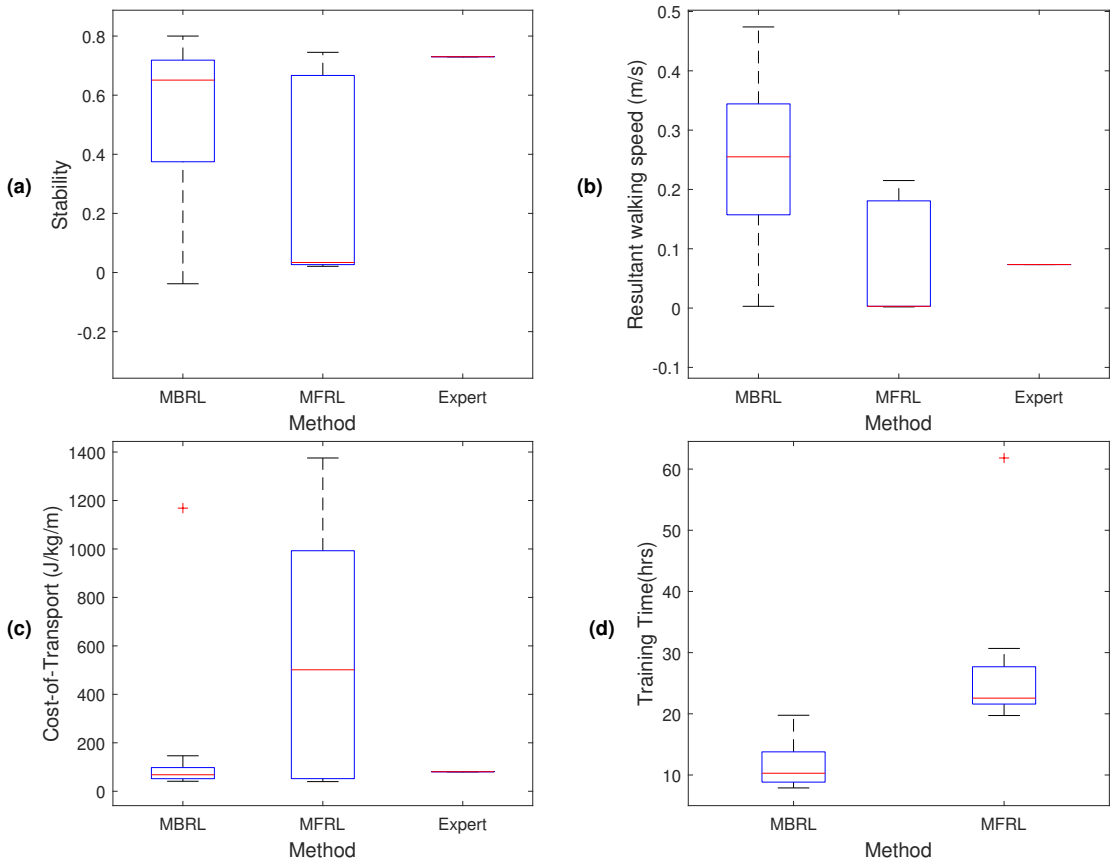


Figure 5.3.1: The Box Plot of MBRL with Continuous Training compared to MFRL. (a) Box plot of stability by method; (b) Box plot of resultant walking speed by method ; (c) Box plot of COT by method; (d) Box plot of simulation time by method.

walking.

Lastly, it was observed that MBRL with continuous training demonstrated superior training efficiency, aligning with the primary objective of this thesis. It exhibited a greater capacity to learn efficient locomotion, particularly for longer distances. Additionally, considering that a smaller fraction of MFRL agents achieved extended walking within 400 episodes, the actual simulation time required for MFRL to achieve comparable results would likely exceed what is presented in Figure 5.3.1(d).

In summary, the findings emphasized the consistent advantages of employing MBRL with continuous training over MFRL in terms of stability, walking speed, energy efficiency, and training effectiveness. Furthermore, MBRL with continuous training not only outperforms MFRL but also attains performance levels similar to those of the predefined expert gait, showcasing its potential for real-world applications where stable and efficient locomotion is of paramount importance.

5.3.3 ANCOVA Tests

However, it is essential to acknowledge that reference walking speed (v_{ref}) represents a covariate in this comparative analysis. To ensure a rigorous assessment that accounts for the potential influence of this covariate, an ANCOVA was conducted to compare MBRL with continue learning and MFRL. This statistical analysis enabled us to compare these two RL approaches while effectively controlling for the covariate v_{ref}. This analysis primarily focused on the following dependent variables, as previously defined in Section 3.3: stability, resultant walking speed, COT and learning efficiency. This ANCOVA analysis provides a robust framework for evaluating the relative performance of the two RL approaches, while considering the potential impact of reference walking speed (v_{ref}). The results of this analysis offer deeper insights into the effectiveness of each method in achieving stable and efficient gait control for our soft robot. The details of ANCOVA is discussed in A.3.

The hypothesis holds that MBRL with continuous training is significantly more effective and efficient in achieving stable and efficient gait control for our soft robot SoftQ compared to MFRL, across various reference velocities (v_{ref}).

Stability

The ANCOVA analysis revealed both the method of RL (MBRL or MFRL) and the covariate v_{ref} have a statistically significant effect on walking stability. The F-value of 5.9902 and the p-value of 0.0307 in the effect of method indicate that the method of RL significantly influences walking stability. Given that the p-value is less than 0.05, this effect is statistically significant, suggesting that one of these methods is likely better than the other in terms of enhancing walking stability. v_{ref} serves as a covariate in this study, and it too shows a statistically significant F-value of 4.3583 and a p-value of 0.0378. This implies that v_{ref} independently impacts walking stability, and its effect is statistically significant. The interaction between the method and v_{ref} is also significant, with an F-value of 5.4802 and a p-value of 0.0204. This suggests that the effect of the learning method on walking stability is not consistent across all levels of v_{ref}. In other words, the optimal choice between MBRL and MFRL may depend on the specific value of v_{ref}. This suggests that the RL approaches has a significant positive impact on the stability of walking.

Source	DoF	Sum Sq	Mean Sq	F	Prob>F
v_{ref}	2	0.4734	0.2367	4.3583	0.0378
Method	1	0.3254	0.3254	5.9902	0.0307
v_{ref}·Method	2	0.5953	0.2977	5.4802	0.0204
Error	12	0.6518	0.0543		

Table 5.3.1: ANCOVA test of stability metrics.

Resultant walking speed

The ANCOVA analysis concluded that the method of RL significantly impacts resultant walking speed, while v_{ref} does not. The DoF for v_{ref} is 2, and it has a Sum of Squares (Sum Sq) of 0.0056, yielding a Mean Square (Mean Sq) of 0.0028. The F-value is a low 0.2133, and the p-value is notably high at 0.8109. These metrics strongly suggest that v_{ref} does not have a statistically significant impact on the resultant walking speed in this case. The Method has a DoF of 1 with a Sum Sq of 0.1390, resulting in a Mean Sq of 0.1390. The F-value is considerably high at 10.6660, and the p-value is 0.0068, well below the 0.05 threshold. These values point to a statistically significant influence of the RL method on resultant walking speed. This implies that either MBRL or MFRL has a substantial impact on walking speed, warranting further investigation to determine the most effective approach. The interaction term shows a DoF of 2, a Sum Sq of 0.0991, and a Mean Sq of approximately 0.0496. With an F-value of 3.8011 and a p-value of 0.0526, the interaction term is borderline significant. This indicates that the effect of the RL method on resultant walking speed may vary depending on v_{ref}, although the evidence is not strong enough to make this conclusion definitively.

Source	DoF	Sum Sq	Mean Sq	F	Prob>F
v_{ref}	2	0.0056	0.0028	0.2133	0.8109
Method	1	0.1390	0.1390	10.6660	0.0068
v_{ref}·Method	2	0.0991	0.0496	3.8011	0.0526
Error	12	0.1564	0.0130		

Table 5.3.2: ANCOVA test of resultant walking speed.

Cost-of-Transport

The data indicates that the choice of RL method (MBRL or MFRL) has a statistically significant impact on COT. On the other hand, the influence of v_{ref} is borderline significant, and the interaction term between v_{ref} and method of RL is not statistically significant. In effect of v_{ref}, the F-value stands at 3.7479 with a p-value of 0.0544. While the F-value suggests a reasonable effect, the p-value marginally exceeds the typical 0.05 significance threshold, making the influence of v_{ref} on COT statistically borderline. In effect of RL method, the F-value is 5.5947 with a p-value of 0.0357. Given the p-value is less than 0.05, the influence of the chosen RL method on COT is statistically significant. In the effect of interaction term (v_{ref}·Method), the F-value is 2.9587, and the p-value is 0.0902. Although the F-value suggests some degree of effect, the p-value is greater than 0.05, suggesting that the interaction effect is not statistically significant. Therefore, while the RL method appears to be a crucial factor affecting COT, the role of v_{ref} and its interaction with the method may require additional research to be conclusively understood.

Source	DoF	Sum Sq	Mean Sq	F	Prob>F
v_{ref}	2	9.5553×10^5	4.7776×10^5	3.7479	0.0544
Method	1	7.1318×10^5	7.1318×10^5	5.5947	0.0357
v_{ref}·Method	2	7.5432×10^5	3.7716×10^5	2.9587	0.0902
Error	12	1.5297×10^6	1.747×10^5		

Table 5.3.3: ANCOVA test of COT.

Learning efficiency

The ANCOVA analysis underscores the statistical significance of the RL method in influencing learning efficiency. The v_{ref} term as a covariate, is characterized by a Degrees of Freedom (DoF) of 2, Sum of Squares (Sum Sq) of 208.0905, and Mean Square (Mean Sq) of 104.0452. The F-value is 1.2107 and the p-value is 0.3319. Given the p-value is well above the common 0.05 significance threshold, v_{ref} does not appear to have a statistically significant impact on learning efficiency. The Method term has a DoF of 1, a Sum Sq of 1.1997×10^3, and a Mean Sq of 1.1997×10^3. The F-value is remarkably high at 13.9597, and the p-value is notably low at 0.0028. These values confirm that the Method is statistically significant in affecting learning efficiency, suggesting a noteworthy influence of the chosen RL approach. The interaction term has a DoF of 2, a Sum Sq of 265.9588, and a Mean Sq of 132.9794. The F-value is 1.5473, and the p-value is 0.2524. These numbers indicate that the interaction term does not meet the threshold for statistical significance, which implies that the effect of the RL method on learning efficiency is generally consistent across different v_{ref} levels.

Source	DoF	Sum Sq	Mean Sq	F	Prob>F
v_{ref}	2	208.0905	104.0452	1.2107	0.3319
Method	1	1.1997×10^3	1.1997×10^3	13.9597	0.0028
v_{ref}·Method	2	265.9588	132.9794	1.5473	0.2524
Error	12	1.0313×10^3	85.9405		

Table 5.3.4: ANCOVA test of learning efficiency.

In summary, the statistical analyses not only support but also extend the project's earlier findings. The results indicated that the choice of RL method had a statistically significant impact on stability, resultant walking speed, COT, and learning efficiency, while the influence of v_{ref} varied across these metrics and was not always statistically significant. In addition, MBRL has proven itself not only effective in providing superior gait control but also adaptable across different walking speeds. It shows promise in task-specific skill development, which could be manipulated through different reward functions in the RL framework.

5.4 Field Test

The culmination of the MBRL and continuous training efforts is the deployment of the trained controller onto the physical SoftQ robot. This section presents the results of the field tests, providing insights into the controller's transferability, generalization capabilities, and its performance in real-world scenarios.

The primary objective of the field tests is to evaluate the extent to which the controller's learned policy can be transferred from the simulated environment to the physical robot. To facilitate these tests, two room dividers are placed at the front and side of the robot to form the test field and reflect the light signals from the ToF sensors as well as the localization references for the robot. The task assigned to the robot is to walk from its initial position towards the front divider and stop at a distance of 0.3 m to the front divider, the total walking distance is approximately 1.5 m.

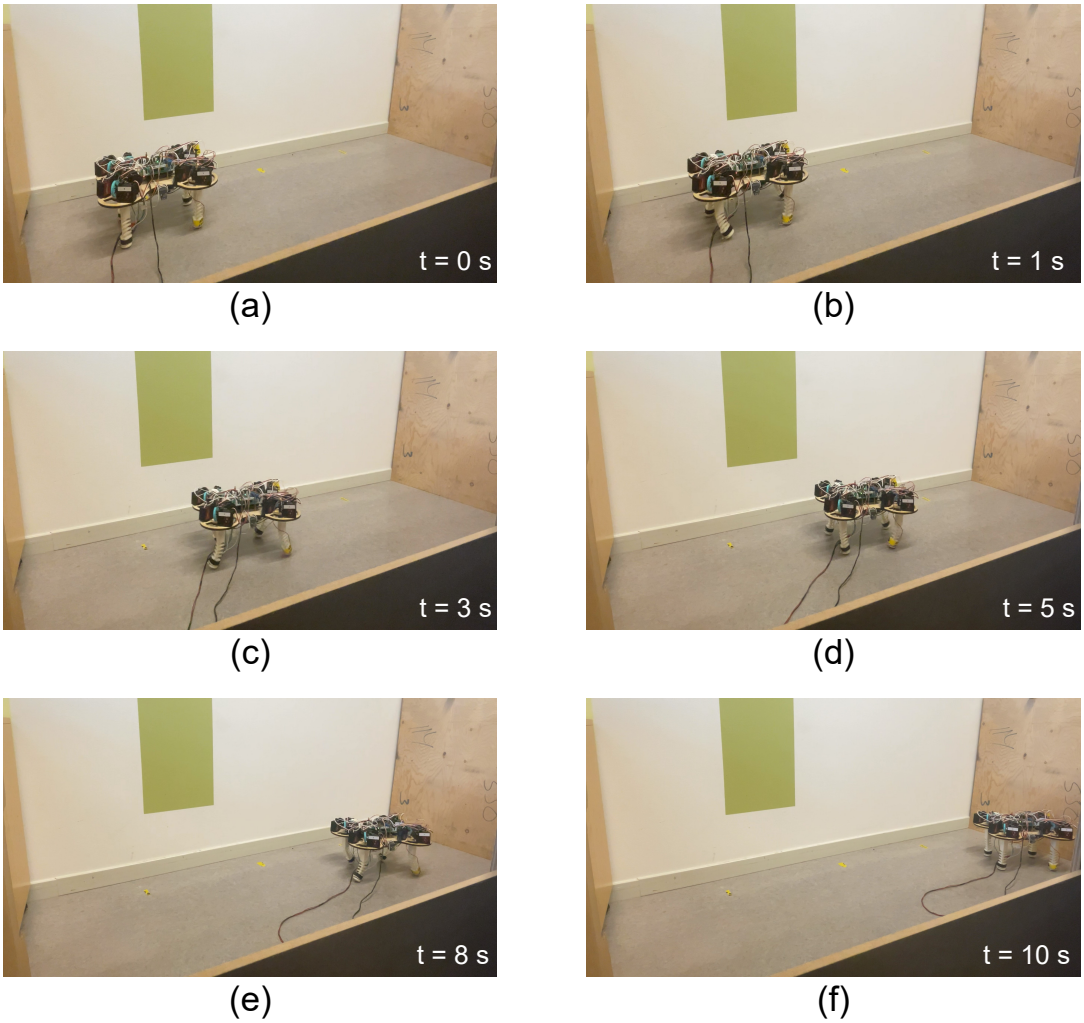


Figure 5.4.1: Snapshots at Different Time Stamps of the Field Test of SoftQ Walking. (a) The initial start point of the robot SoftQ. (b - e) The robot walked forward with the trained control policy at different time while walking. (f) , the robot finished walking at the designated endpoint.

The directly learned walking gait for $v_{ref} = 0.3m/s$ was tested on the real robot of SoftQ. Figure 5.4.1 provides a photographic stills of the field test at various time stamps. It begins with the robot's initial starting point (a) and progresses through its forward movement with the trained control policy at different time intervals (b - e). Finally, the robot successfully completes the 1.5-meter walk in 10 seconds and stands at the designated endpoint (f). These field tests demonstrate the exceptional transferability and adaptability of the controller, showcasing its ability to seamlessly apply learned behaviors in real-world scenarios. The successful execution of the task underscores the practical viability of our approach, emphasizing the potential for real-world applications where precise locomotion and control are essential.

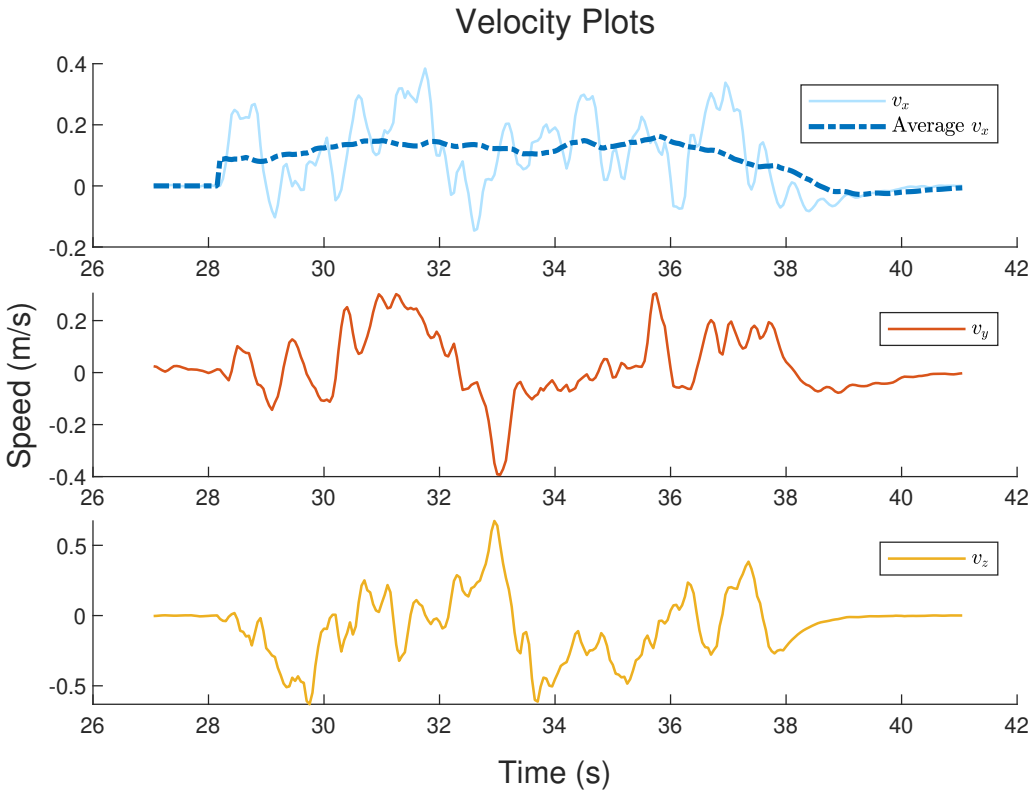


Figure 5.4.2: The field test results of measurement of velocity at x, y and z axis.

Figure 5.4.2 shows the recorded speed change in the three directions of the robot center. The speed in the forward direction starts around 0.2 m/s, represented by the filtered curve of the forward speeds. This speed then decreases to around -0.1 m/s before gradually increasing to approximately 0.4 m/s. Subsequently, the speed fluctuates between 0 m/s and 0.3 m/s until it reaches the endpoint. The resultant walking speed, calculated based on the total distance traveled, averages around 0.15 m/s. It is noteworthy that this achieved speed is somewhat lower than the target speed of 0.3 m/s in the simulation. This discrepancy can be attributed to the reality gap caused by inaccuracies in the model's actuators, sensors, and foot-ground contact dynamics. In the lateral directions (y and z), observable speed perturbations are present, particularly in the z axis. These fluctuations indicate the robot's rapid vertical motion, ranging

from -0.6 m/s to 0.6 m/s. The y-axis velocities range from -0.4 m/s to 0.3 m/s, which may be attributed to self-corrections and lateral movements during walking.

These results underscore the exceptional transferability and generalization capabilities of the trained controller, highlighting its ability to adapt learned behaviors to real-world settings despite disparities between the simulation and the physical environment. The field tests further emphasize the controller's impressive locomotion performance across various terrains and conditions, providing evidence of its adaptability and robustness in practical scenarios. The outstanding performance observed in the field tests holds significant implications for the power of this methodology enhanced by MBRL. The robot prototype successfully completes the preliminary walking test under real-time execution by utilizing the SAC trained agents.

Chapter 6

Discussions and Conclusions

This chapter summarizes the key findings and contributions of this thesis, providing insights into the achievements and implications of the research conducted. It also outlines the limitations of this thesis and suggests potential avenues for future research.

6.1 Discussion

In summary, this research endeavor has sought to unravel the complexities surrounding gait control in soft quadruped robots, with the goal of enhancing learning efficiency by Model Based Reinforcement Learning (MBRL) for more effective control strategies. A systematic exploration of the research questions outlined in this study has yielded valuable insights and findings that contribute significantly to the field of robotics and autonomous locomotion. Additionally, this section will conduct a comprehensive discussion of the research questions outlined in the introduction, with the aim of providing answers and insights based on the research conducted in this thesis.

6.1.1 Answer to Research Questions 1

The investigation into restricting the state space and designing a surrogate model with high estimation accuracy was instrumental in the pursuit of efficient and accurate simulations and training for soft quadruped robots. This research sought to address this question through two distinct approaches: pattern-defined reinforcement learning and parameterization. These methods provide a solid foundation for efficient and accurate simulations and training, ultimately leading to improved gait control strategies, as shown in Section 5.3.

The concept of pattern-defined reinforcement learning showed promise as a means to restrict the state space efficiently. By selecting a subset of features based on specific patterns, such as the trot, the study aimed to simplify the state space while retaining essential information. This approach represents a significant step towards efficient simulations and training.

Parameterization introduced a higher-level abstraction into the state-action space by using phases between gait and real motors to parameterize the gaits of quadruped robots. This

approach, while complex, holds the potential to simplify the representation of the state space. It has the advantage of capturing critical information about locomotion while streamlining the learning process.

Additionally, it is important to acknowledge the inherent challenges posed by the interconnected nature of the robot's legs. This unique characteristic of soft quadruped robots introduces constraints on the diversity of gaits that can be studied. Unlike robots with independently movable legs, soft quadrupeds exhibit a more synchronized form of locomotion due to their interconnected structure. This limitation restricts the robot's capability to perform certain types of locomotion that may be relevant in real-world scenarios, where adaptability is key.

Given these constraints, our research approach pivoted towards a practical evaluation methodology that prioritized the tangible impact of controllers on the robot's performance in real-world applications. Rather than delving into the intricacies of algorithms in isolation, we directed our focus towards assessing how these controllers influence the robot's stability, walking speed, and cost-of-transport. These metrics serve as practical indicators of a controller's effectiveness in real-world contexts, aligning our research with the practical needs of the field.

This shift in emphasis allowed us to bridge the gap between theoretical developments and real-world deployment, ensuring that our research contributes directly to the enhancement of soft quadruped robots' capabilities in practical scenarios. In essence, our approach recognized the correlation of algorithm design and real-world performance, striving for a holistic understanding of gait control in these unique robotic systems.

6.1.2 Answer to Research Questions 2

Furthermore, the comparative analysis between model-based and model-free reinforcement learning approaches has shed light on the potential of the former in generating superior control agents for soft quadruped robots.In conclusion, the comparative analysis conducted in response to Research Question 2 has highlighted the potential superiority of model-based reinforcement learning over its model-free counterpart in the realm of soft quadruped robot control. The observed improvements in terms of stability, walking speed, and cost-of-transport underscore the significant role played by model-based reinforcement learning as a benchmark within the scope of this project.

Regarding stability, our findings indicate that both the reinforcement learning method and the parameter v_{ref} have a statistically significant impact on the robot's walking stability. Moreover, the interaction between these factors also proved to be significant, suggesting that the choice of the most effective reinforcement learning method may vary depending on the specific value of v_{ref}. Notably, the reinforcement learning method exhibited a statistically significant influence

on learning efficiency, while v_{ref} did not. Furthermore, the interaction between these two factors was found to be borderline significant, emphasizing the consistent effect of the reinforcement learning method across different v_{ref} levels.

Additionally, our research has brought to light the importance of carefully considering the trade-offs among learning efficiency, simulation accuracy, and long-term planning accuracy when striving to train an optimal gait control policy for soft quadruped robots. This comprehensive examination of trade-offs has provided invaluable insights that contribute to the quest for optimal gait control policies.

One noteworthy aspect of our approach lies in its adaptability, allowing for the customization of the controller's behavior to suit various task objectives by manipulating distinct reward functions within the reinforcement learning framework. As the reinforcement learning algorithm learns to optimize the controller for different tasks based on specific reward functions, this adaptability renders our methodology versatile and applicable to a wide range of scenarios. Consequently, the robotic system can acquire a diverse set of task-specific skills, enhancing its overall versatility and capability.

6.2 Conclusions

6.2.1 Key Findings

In this research, a multi-faceted approach was employed to address the challenges in soft quadruped robotic control, leveraging the potential of reinforcement learning techniques. The key findings are summarized as follows:

- **Optimal Control Strategies**: The study focused on enhancing the control mechanisms of soft quadruped robots. By leveraging model-based reinforcement learning, the research made strides in achieving real-time applicability of learned control strategies.
- **Training and Robustness**: The research highlighted that employing reinforcement learning in real-world scenarios requires meticulous attention to training and robustness. The developed methodologies were designed to bridge the gap between simulated and actual environments.
- **Efficiency Gains**: The implemented algorithms showed a notable improvement in training efficiency and autonomous behavior of the soft quadruped robots. This underscores the efficacy of the methodologies undertaken in the study.
- **Surrogate Model Architecture**: The research emphasized the critical role played by selecting an appropriate deep neural network architecture for the task at hand. This selection was essential in optimizing the control strategies effectively.

Overall, the study contributes to advancing the field of robotics by addressing challenges such as time efficiency and dimensionality. It showcases the feasibility of applying advanced machine learning techniques to practical robotic control issues.

6.2.2 Conclusion

In conclusion, this thesis embarked on an exploration of enhancing learning efficiency in the context of optimal gait control for soft quadruped robots, with a primary emphasis on addressing the challenges through a MBRL approach. The overall objective was to develop robust gait control policies capable of accommodating the continuous and deformable morphology of the robot while contributing to the advancement of RL control strategies for real-world applications. To substantiate the theoretical framework and proposed methodologies, a series of experimental tests were conducted, ensuring the practical applicability of the research findings. Additionally, a tailored software architecture designed specifically for the soft quadruped robot, SoftQ, was introduced to facilitate the research's objectives.

Nonetheless, it is imperative to acknowledge certain limitations and delimitation that influenced the thesis's scope. The "sim-to-real gap" constraint underscored the challenges associated with translating simulation results to real-world scenarios, reinforcing the need for a practical evaluation of the proposed gait control strategies. Moreover, the interconnected nature of the robot's legs imposed constraints on the variety of gaits that could be studied, limiting the exploration of certain locomotion strategies relevant to real-world contexts. These limitations necessitated an evaluation focused on the practical impact of controllers on the robot's stability, walking speed, and cost-of-transport, rather than delving into algorithm intricacies.

Delimitations were intentionally set to maintain a concentrated focus on soft quadruped robots, excluding other robotic categories, external environmental factors, hardware design variations, and alternative RL algorithms. This deliberate narrowing of scope aimed to foster an in-depth analysis of the specific challenges related to gait control in the chosen context.

The research methodology encompassed stages of data collection, MBRL algorithm development, evaluation, analysis, and validation, culminating in the comparison of model-based and model-free RL approaches. Practical validation through physical implementation on SoftQ ensured the relevance of the findings in real-world scenarios. The project also acknowledged ethical and sustainability considerations, emphasizing the responsible use of robotics technology and environmentally-conscious design choices.

In essence, this thesis contributes to the advancement of soft quadruped robotics by proposing an MBRL approach to gait control, with practical validation underscoring its potential for real-world applications. It is anticipated that the insights gained from this research will pave the way for more efficient and effective RL control strategies, ultimately enhancing the capabilities of

soft quadruped robots in various domains, while remaining mindful of ethical and sustainability concerns.

6.3 Future Work

This thesis has explored the context of optimal gait control for soft quadruped robots, uncovering valuable insights and paving the way for further investigations. As with any scientific endeavor, there are promising avenues for future research that can expand upon the knowledge and limitations identified in this study. In this section, the potential directions of future research were outlined for the advancement of soft quadruped robot system.

Hardware advancements

The field of soft robotics is continually evolving, with ongoing developments in soft actuators, materials, and sensors. Future research can explore the integration of state-of-the-art hardware components to enhance the physical capabilities and adaptability of quadruped robots. The continuous evolution of soft robotic materials and actuators presents an opportunity for future research. Investigating the integration of cutting-edge soft robotics hardware, such as novel actuators and sensors[100, 101], can enhance the physical capabilities and adaptability of quadruped robots, thereby advancing their locomotion control.

Multi-terrain adaptation

Soft quadruped robots should be able to navigate a wide range of terrains, from rugged outdoor environments to indoor spaces. Future research should prioritize the development of control algorithms that enable seamless adaptation to various surfaces. This includes rough, uneven terrains, dynamically changing landscapes, and transitions between different terrain types. Achieving robust multi-terrain adaptation will significantly expand the practical utility of soft quadruped robots across diverse applications.

Online learning and adaptation

The ability of robots to adapt to changing environmental conditions and unforeseen challenges in real-time is crucial. Future research can focus also on developing online learning and adaptation mechanisms that allow soft quadruped robots to continuously update their control policies during operation. This real-time adaptability will be essential for improving performance, responsiveness, and autonomy in dynamic environments.

Feedback control to achieve position control

Incorporating feedback control mechanisms to achieve precise position control is an important area of exploration. Future research can delve into advanced control strategies that enable soft quadruped robots to achieve and maintain specific positions. The significance of this lies in tasks that demand meticulous positioning, such as precise robotic manipulation, obstacle avoidance, and navigating through confined spaces. By developing control techniques that seamlessly incorporate precise position control into the skill set of soft quadruped robots, we can open up new possibilities for their application in a wider array of scenarios.

Robustness and fault tolerance

Ensuring the robustness and fault tolerance of soft quadruped robots is critical for real-world applications. Future research can investigate robust control strategies capable of withstanding disturbances, uncertainties, and unexpected environmental changes. Additionally, the development of fault-tolerant control mechanisms will be essential for handling hardware failures or damage, ensuring the reliability and safety of these robots in challenging scenarios.

6.3.1 Sustainability and Ethical Considerations

In a world increasingly concerned with environmental impact, the notion of environmental sustainability is more relevant than ever. In the context of this thesis, significant consideration was given to the COT metric as a guiding factor in the design of the controller. Emphasis was placed on the role of energy efficiency in the performance of soft quadruped robots, with the objective of creating controllers that enhance the robot's endurance while minimizing its environmental impact.

The choice of materials and manufacturing techniques applied in this research proved to be highly advantageous in achieving these goals. The utilization of compliant and soft materials was aligned with the inherent characteristics of quadruped locomotion and contributed to improved energy efficiency. These materials facilitated better energy absorption and recovery during the robot's gait, leading to reduced energy losses and enhanced overall efficiency.

Furthermore, the incorporation of advanced 3D printing methods enabled the creation of intricate and customized components for the soft quadruped robot. This level of manufacturing precision not only minimized material wastage but also bolstered structural integrity, further supporting the overarching objectives of sustainability and energy efficiency. These considerations are not mere addenda but foundational principles that should guide every stage of robotic development.

Crucially, regulatory frameworks serve as the scaffolding upon which sustainability and ethics

are built. Collaborative efforts among governments, industry bodies, and researchers are essential to craft standards and guidelines that embody these principles. Such frameworks are not restrictive but liberating, providing a clear roadmap for the responsible development and deployment of soft quadruped robots.

As soft quadruped robots become integrated into society, ethical considerations, including privacy, safety, and their impact on employment, require careful examination. Future research should also delve into these ethical dimensions to ensure the responsible deployment of these robots.

6.3.2 Final Words

In these final words, it is crucial to acknowledge the collaborative efforts of researchers, engineers and innovators who continue to push the boundaries of robotics. The journey to master gait control in soft quadruped robots is ongoing, with numerous challenges and opportunities awaiting exploration.

In the grand scheme of scientific exploration, this research represents a small but significant stride toward unlocking the full potential of soft quadruped robots. It is our hope that the knowledge gained here will inspire further research and innovation, ultimately leading to the realization of highly capable and versatile soft quadruped robots that can navigate the complexities of our ever-changing world.

In closing, this thesis marks not an end, but rather a new beginning in the journey to harness the untapped potential of soft quadruped robots. May our collective efforts continue to drive progress, pushing the boundaries of what these remarkable machines can achieve.

Bibliography

[1] Billard, Aude and Kragic, Danica. “Trends and challenges in robot manipulation”. In: *Science* 364.6446 (June 2019), eaat8414.

[2] Wang, Tian-Miao, Tao, Yong, and Liu, Hui. “Current Researches and Future Development Trend of Intelligent Robot: A Review”. In: *International Journal of Automation and Computing* 15.5 (Oct. 2018), pp. 525–546. ISSN: 1751-8520.

[3] Li, Yatao, Wu, Shunkai, He, Leiying, Tong, Junhua, Zhao, Runmao, Jia, Jiangming, Chen, Jianneng, and Wu, Chuanyu. “Development and field evaluation of a robotic harvesting system for plucking high-quality tea”. In: *Computers and Electronics in Agriculture* 206 (Mar. 2023), p. 107659. ISSN: 0168-1699.

[4] Zhang, Xinhai, Tao, Jianbo, Tan, Kaige, Törngren, Martin, Sánchez, José Manuel Gaspar, Ramli, Muhammad Rusyadi, Tao, Xin, Gyllenhammar, Magnus, Wotawa, Franz, Mohan, Naveen, Nica, Mihai, and Felbinger, Hermann. “Finding Critical Scenarios for Automated Driving Systems: A Systematic Mapping Study”. In: *IEEE Transactions on Software Engineering* 49.3 (Mar. 2023), pp. 991–1026. ISSN: 1939-3520.

[5] Riedo, Fanny, Chevalier, Morgane, Magnenat, Stéphane, and Mondada, Francesco. “Thymio II, a robot that grows wiser with children”. In: *2013 IEEE Workshop on Advanced Robotics and its Social Impacts*. Nov. 2013, pp. 187–193.

[6] OpenAI. *GPT-4 Technical Report*. Mar. 2023.

[7] Li, Yibin, Li, Bin, Ruan, Jiuhong, and Rong, Xuewen. “Research of mammal bionic quadruped robots: A review”. In: *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*. Sept. 2011, pp. 166–171.

[8] Hawkes, Elliot W., Blumenschein, Laura H., Greer, Joseph D., and Okamura, Allison M. “A soft robot that navigates its environment through growth”. In: *Science Robotics* 2.8 (July 2017), eaan3028. ISSN: 2470-9476.

[9] Hu, Xuran, He, Feng, Xiao, Peng, Wang, Tao, Zhang, Decai, Zhou, Xingfu, and Fan, Yan. “Design of a Quadruped inspection Robot Used in Substation”. In: *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. Vol. 4. June 2021, pp. 766–769.

[10] Hewing, Lukas, Wabersich, Kim P., Menner, Marcel, and Zeilinger, Melanie N. "Learning-based model predictive control: Toward safe learning in control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), pp. 269–296.

[11] Muralidharan, Seshagopalan Thorapalli, Zhu, Ruihao, Ji, Qinglei, Feng, Lei, Wang, Xi Vincent, and Wang, Lihui. "A soft quadruped robot enabled by continuum actuators". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. Aug. 2021, pp. 834–840.

[12] Wang, Jue and Chortos, Alex. "Control Strategies for Soft Robot Systems". In: *Advanced Intelligent Systems* 4.5 (2022), p. 2100165. ISSN: 2640-4567.

[13] Thorapalli Muralidharan, Seshagopalan and Zhu, Ruihao. "Continuum Actuator Based Soft Quadruped Robot". Thesis. Stockholm, Sweden: KTH Royal Institute of Technology, 2020.

[14] Danelia, David and Fu, Shuo. "Structure and Gait Optimizationof a Soft Quadrupedal Robot". Thesis. KTH Royal Institute of Technology, 2021.

[15] Akhtaruzzaman, Md., Shafie, Amir Akramin, and Khan, Md. Raisuddin. "Gait analysis: systems, technologies, and importance". In: *Journal of Mechanics in Medicine and Biology* 16.07 (Nov. 2016), p. 1630003. ISSN: 0219-5194.

[16] Xu, Kun, Zi, Peijin, and Ding, Xilun. "Gait Analysis of Quadruped Robot Using the Equivalent Mechanism Concept Based on Metamorphosis". In: *Chinese Journal of Mechanical Engineering* 32.1 (Feb. 2019), p. 8. ISSN: 2192-8258.

[17] Ji, Qinglei, Fu, Shuo, Tan, Kaige, Thorapalli Muralidharan, Seshagopalan, Lagrelius, Karin, Danelia, David, Andrikopoulos, Georgios, Wang, Xi Vincent, Wang, Lihui, and Feng, Lei. "Synthesizing the optimal gait of a quadruped robot with soft actuators using deep reinforcement learning". In: *Robotics and Computer-Integrated Manufacturing* 78 (Dec. 2022), p. 102382. ISSN: 0736-5845.

[18] Recht, Benjamin. "A Tour of Reinforcement Learning: The View from Continuous Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019), pp. 253–279.

[19] Zhang, Haochong, Cao, Rongyun, Zilberstein, Shlomo, Wu, Feng, and Chen, Xiaoping. "Toward Effective Soft Robot Control via Reinforcement Learning". In: *Intelligent Robotics and Applications*. Ed. by YongAn Huang, Hao Wu, Honghai Liu, and Zhouping Yin. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 173–184. ISBN: 978-3-319-65289-4.

[20] Ji, Qinglei. "Learning-based Control for 4D Printing and Soft Robotics". Thesis. Stockholm, Sweden: KTH Royal Institute of Technology, 2022.

[21] Cebe, Oguzhan, Tiseo, Carlo, Xin, Guiyang, Lin, Hsiu-chin, Smith, Joshua, and Mistry, Michael. "Online Dynamic Trajectory Optimization and Control for a Quadruped Robot". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. May 2021, pp. 12773–12779.

[22] Chignoli, Matthew and Kim, Sangbae. "Online Trajectory Optimization for Dynamic Aerial Motions of a Quadruped Robot". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. May 2021, pp. 7693–7699.

[23] Chignoli, Matthew, Morozov, Savva, and Kim, Sangbae. "Rapid and Reliable Quadruped Motion Planning with Omnidirectional Jumping". In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 6621–6627.

[24] Wang, Zhicheng, Li, Anqiao, Zheng, Yixiao, Xie, Anhuan, Li, Zhibin, Wu, Jun, and Zhu, Qiuguo. "Efficient learning of robust quadruped bounding using pretrained neural networks". In: *IET Cyber-Systems and Robotics* 4.4 (2022), pp. 331–338. ISSN: 2631-6315.

[25] Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, July 2018, pp. 1861–1870.

[26] Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. "Understanding deep learning (still) requires rethinking generalization". In: *Communications of the ACM* 64.3 (2021), pp. 107–115. ISSN: 0001-0782.

[27] Haarnoja, Tuomas, Zhou, Aurick, Hartikainen, Kristian, Tucker, George, Ha, Sehoon, Tan, Jie, Kumar, Vikash, Zhu, Henry, Gupta, Abhishek, Abbeel, Pieter, and Levine, Sergey. *Soft Actor-Critic Algorithms and Applications*. Jan. 2019.

[28] Annaswamy, Anuradha M. "Adaptive Control and Intersections with Reinforcement Learning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 6.1 (2023), null.

[29] Ray, Soumya and Tadepalli, Prasad. "Model-Based Reinforcement Learning". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 690–693. ISBN: 978-0-387-30164-8.

[30] Polydoros, Athanasios S. and Nalpantidis, Lazaros. "Survey of Model-Based Reinforcement Learning: Applications on Robotics". In: *Journal of Intelligent & Robotic Systems* 86.2 (May 2017), pp. 153–173. ISSN: 1573-0409.

[31] Arulkumaran, Kai, Deisenroth, Marc Peter, Brundage, Miles, and Bharath, Anil Anthony. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (Nov. 2017), pp. 26–38. ISSN: 1558-0792.

[32] Wang, Tingwu, Bao, Xuchan, Clavera, Ignasi, Hoang, Jerrick, Wen, Yeming, Langlois, Eric, Zhang, Shunshi, Zhang, Guodong, Abbeel, Pieter, and Ba, Jimmy. *Benchmarking Model-Based Reinforcement Learning*. July 2019.

[33] Zhong, Yuhai, Wang, Runxiao, Feng, Huashan, and Chen, Yasheng. "Analysis and research of quadruped robot's legs: A comprehensive review". In: *International Journal of Advanced Robotic Systems* 16.3 (May 2019), p. 1729881419844148. ISSN: 1729-8806.

[34] Owaki, Dai and Ishiguro, Akio. "A Quadruped Robot Exhibiting Spontaneous Gait Transitions from Walking to Trotting to Galloping". In: *Scientific Reports* 7.1 (Mar. 2017), p. 277. ISSN: 2045-2322.

[35] Allen, K., DeCamp, C. E., Braden, T. D., and Balms, Michelle. "Kinematic Gait Analysis of the Trot in Healthy Mixed Breed Dogs". In: *Veterinary and Comparative Orthopaedics and Traumatology* 07.4 (1994), pp. 148–153. ISSN: 0932-0814, 2567-6911.

[36] Fletcher, Thomas F. and Datt (nee Johnson), Vicki L. *Trot*. Aug. 2012.

[37] Shao, Yecheng, Jin, Yongbin, Liu, Xianwei, He, Weiyan, Wang, Hongtao, and Yang, Wei. "Learning Free Gait Transition for Quadruped Robots Via Phase-Guided Controller". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 1230–1237. ISSN: 2377-3766.

[38] Ji, Qinglei, Fu, Shuo, Feng, Lei, Andrikopoulos, George, Wang, Xi Vincent, and Wang, Lihui. "Omnidirectional walking of a quadruped robot enabled by compressible tendon-driven soft actuators". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2022, pp. 11015–11022.

[39] Biswal, Priyaranjan and Mohanty, Prases K. "Development of quadruped walking robots: A review". In: *Ain Shams Engineering Journal* 12.2 (June 2021), pp. 2017–2031. ISSN: 2090-4479.

[40] Fazeli, N., Oller, M., Wu, J., Wu, Z., Tenenbaum, J. B., and Rodriguez, A. "See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion". In: *Science Robotics* 4.26 (Jan. 2019), eaav3123.

[41] Yang, Zhengxin and Zhang, Li. "Magnetic Actuation Systems for Miniature Robots: A Review". In: *Advanced Intelligent Systems* 2.9 (2020), p. 2000082. ISSN: 2640-4567.

[42] Ding, Chao, Zhou, Lelai, Li, Yibin, and Rong, Xuewen. "A Novel Dynamic Locomotion Control Method for Quadruped Robots Running on Rough Terrains". In: *IEEE Access* 8 (2020), pp. 150435–150446. ISSN: 2169-3536.

[43] Lee, Joonho, Hwangbo, Jemin, Wellhausen, Lorenz, Koltun, Vladlen, and Hutter, Marco. “Learning quadrupedal locomotion over challenging terrain”. In: *Science Robotics* 5.47 (Oct. 2020), eabc5986.

[44] Choi, Suyoung, Ji, Gwanghyeon, Park, Jeongsoo, Kim, Hyeongjun, Mun, Juhyeok, Lee, Jeong Hyun, and Hwangbo, Jemin. “Learning quadrupedal locomotion on deformable terrain”. In: *Science Robotics* 8.74 (Jan. 2023), eade2256.

[45] Lee, Young Hun, Lee, Yoon Haeng, Lee, Hyunyong, Phan, Luong Tin, Kang, Hansol, Kim, Uikyum, Jeon, Jeongmin, and Choi, Hyouk Ryeol. “Trajectory design and control of quadruped robot for trotting over obstacles”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 4897–4902.

[46] Sun, Wenkai, Tian, Xiaojie, Song, Yong, Pang, Bao, Yuan, Xianfeng, and Xu, Qingyang. “Balance Control of a Quadruped Robot Based on Foot Fall Adjustment”. In: *Applied Sciences* 12.5 (Jan. 2022), p. 2521. ISSN: 2076-3417.

[47] Li, Zhi, Li, Bin, Liang, Qixing, Liu, Weilong, Hou, Landong, and Rong, Xuewen. “A quadruped robot obstacle avoidance and personnel following strategy based on ultra-wideband and three-dimensional laser radar”. In: *International Journal of Advanced Robotic Systems* 19.4 (July 2022), p. 17298806221114705. ISSN: 1729-8806.

[48] Miki, Takahiro, Lee, Joonho, Hwangbo, Jemin, Wellhausen, Lorenz, Koltun, Vladlen, and Hutter, Marco. “Learning robust perceptive locomotion for quadrupedal robots in the wild”. In: *Science Robotics* 7.62 (Jan. 2022), eabk2822. ISSN: 2470-9476.

[49] Fang, Guoxin, Matte, Christopher-Denny, Scharff, Rob B. N., Kwok, Tsz-Ho, and Wang, Charlie C. L. “Kinematics of Soft Robots by Geometric Computing”. In: *IEEE Transactions on Robotics* 36.4 (Aug. 2020), pp. 1272–1286. ISSN: 1941-0468.

[50] Lee, Kit-Hang, Leong, Martin C. W., Chow, Marco C. K., Fu, Hing-Choi, Luk, Wayne, Sze, Kam-Yim, Yeung, Chung-Kwong, and Kwok, Ka-Wai. “FEM-based soft robotic control framework for intracavitary navigation”. In: *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. July 2017, pp. 11–16.

[51] Giorelli, Michele, Renda, Federico, Calisti, Marcello, Arienti, Andrea, Ferri, Gabriele, and Laschi, Cecilia. “Neural Network and Jacobian Method for Solving the Inverse Statics of a Cable-Driven Soft Arm With Nonconstant Curvature”. In: *IEEE Transactions on Robotics* 31.4 (Aug. 2015), pp. 823–834. ISSN: 1941-0468.

[52] Poulakakis, Ioannis and Grizzle, Jessy W. “The Spring Loaded Inverted Pendulum as the Hybrid Zero Dynamics of an Asymmetric Hopper”. In: *IEEE Transactions on Automatic Control* 54.8 (Aug. 2009), pp. 1779–1793. ISSN: 1558-2523.

[53] Hwangbo, Je Min. “Simulation to Real World: Learn to Control Legged Robots”. Doctoral Thesis. ETH Zurich, 2018.

[54] Lagrelius, Karin. “Comparing Four Modelling Methods for the Simulation of a Soft Quadruped Robot”. Thesis. KTH Royal Institute of Technology, 2022.

[55] Hutter, Marco, Gehring, Christian, Jud, Dominic, Lauber, Andreas, Bellicoso, C. Dario, Tsounis, Vassilios, Hwangbo, Jemin, Bodie, Karen, Fankhauser, Peter, Bloesch, Michael, Diethelm, Remo, Bachmann, Samuel, Melzer, Amir, and Hoepflinger, Mark. “ANYmal - a highly mobile and dynamic quadrupedal robot”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016, pp. 38–44.

[56] Bledt, Gerardo, Powell, Matthew J., Katz, Benjamin, Di Carlo, Jared, Wensing, Patrick M., and Kim, Sangbae. “MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 2245–2252.

[57] Bledt, Gerardo, Wensing, Patrick M., Ingersoll, Sam, and Kim, Sangbae. “Contact Model Fusion for Event-Based Locomotion in Unstructured Terrains”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 4399–4406.

[58] Kolda, Tamara G., Lewis, Robert Michael, and Torczon, Virginia. “Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods”. In: *SIAM Review* 45.3 (Jan. 2003), pp. 385–482. ISSN: 0036-1445.

[59] Dai, Hongkai, Valenzuela, Andrés, and Tedrake, Russ. “Whole-body motion planning with centroidal dynamics and full kinematics”. In: *2014 IEEE-RAS International Conference on Humanoid Robots*. Nov. 2014, pp. 295–302.

[60] Gehring, Christian, Coros, Stelian, Hutter, Marco, Dario Bellicoso, Carmine, Heijnen, Huub, Diethelm, Remo, Bloesch, Michael, Fankhauser, Peter, Hwangbo, Jemin, Hoepflinger, Mark, and Siegwart, Roland. “Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot”. In: *IEEE Robotics & Automation Magazine* 23.1 (Mar. 2016), pp. 34–43. ISSN: 1558-223X.

[61] Farshidian, Farbod, Jelavic, Edo, Satapathy, Asutosh, Giftthaler, Markus, and Buchli, Jonas. “Real-time motion planning of legged robots: A model predictive control approach”. In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. Nov. 2017, pp. 577–584.

[62] Koch, Kai Henning, Mombaur, Katja, and Soueres, Philippe. “Optimization-based walking generation for humanoid robot”. In: *IFAC Proceedings Volumes*. 10th IFAC Symposium on Robot Control 45.22 (Jan. 2012), pp. 498–504. ISSN: 1474-6670.

[63] Kalakrishnan, Mrinal, Buchli, Jonas, Pastor, Peter, Mistry, Michael, and Schaal, Stefan. “Fast, robust quadruped locomotion over challenging terrain”. In: *2010 IEEE International Conference on Robotics and Automation*. May 2010, pp. 2665–2670.

[64] Neuhaus, Peter D, Pratt, Jerry E, and Johnson, Matthew J. “Comprehensive summary of the Institute for Human and Machine Cognition’ s experience with LittleDog”. In: *The International Journal of Robotics Research* 30.2 (Feb. 2011), pp. 216–235. ISSN: 0278-3649.

[65] Gong, Daoxiong, Yan, Jie, and Zuo, Guoyu. “A Review of Gait Optimization Based on Evolutionary Computation”. In: *Applied Computational Intelligence and Soft Computing* 2010 (June 2010), e413179. ISSN: 1687-9724.

[66] Whitehead, Steven D. and Ballard, Dana H. “Learning to perceive and act by trial and error”. In: *Machine Learning* 7.1 (July 1991), pp. 45–83. ISSN: 1573-0565.

[67] Nakamura, Yutaka, Mori, Takeshi, Sato, Masa-aki, and Ishii, Shin. “Reinforcement learning for a biped robot based on a CPG-actor-critic method”. In: *Neural Networks* 20.6 (Aug. 2007), pp. 723–735. ISSN: 0893-6080.

[68] Wang, Jiayu, Hu, Chuxiong, and Zhu, Yu. “Hierarchical Gait Generation for Modular Robots Using Deep Reinforcement Learning”. In: *2021 IEEE International Conference on Mechatronics (ICM)*. Mar. 2021, pp. 1–6.

[69] Lee, Honglak, Shen, Yirong, Yu, Chih-Han, Singh, G., and Ng, A.Y. “Quadruped robot obstacle negotiation via reinforcement learning”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* May 2006, pp. 3003–3010.

[70] LI, CAI, Lowe, Robert, and Ziemke, Tom. “Humanoids Learning to Walk: A Natural CPG-Actor-Critic Architecture”. In: *Frontiers in Neurorobotics* 7 (2013). ISSN: 1662-5218.

[71] Spröwitz, Alexander, Tuleu, Alexandre, Vespignani, Massimo, Ajallooeian, Mostafa, Badri, Emilie, and Ijspeert, Auke Jan. “Towards dynamic trot gait locomotion: Design, control, and experiments with Cheetah-cub, a compliant quadruped robot”. In: *The International Journal of Robotics Research* 32.8 (July 2013), pp. 932–950. ISSN: 0278-3649.

[72] Di Carlo, Jared, Wensing, Patrick M., Katz, Benjamin, Bledt, Gerardo, and Kim, Sangbae. “Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 1–9.

[73] Ji, Gwanghyeon, Mun, Juhyeok, Kim, Hyeongjun, and Hwangbo, Jemin. "Concurrent Training of a Control Policy and a State Estimator for Dynamic and Robust Legged Locomotion". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 4630–4637. ISSN: 2377-3766.

[74] Tang, Zhi Qiang, Heung, Ho Lam, Tong, Kai Yu, and Li, Zheng. "Model-based online learning and adaptive control for a "human-wearable soft robot" integrated system". In: *The International Journal of Robotics Research* 40.1 (Jan. 2021), pp. 256–276. ISSN: 0278-3649.

[75] Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on Machine Learning*. PMLR, Jan. 2014, pp. 387–395.

[76] Haarnoja, Tuomas, Ha, Sehoon, Zhou, Aurick, Tan, Jie, Tucker, George, and Levine, Sergey. *Learning to Walk via Deep Reinforcement Learning*. June 2019.

[77] Peng, Xue Bin, Andrychowicz, Marcin, Zaremba, Wojciech, and Abbeel, Pieter. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 3803–3810.

[78] Zhang, Xiang-Sun. "Introduction to Artificial Neural Network". In: *Neural Networks in Optimization*. Ed. by Xiang-Sun Zhang. Nonconvex Optimization and Its Applications. Boston, MA: Springer US, 2000, pp. 83–93. ISBN: 978-1-4757-3167-5.

[79] Zou, Jinming, Han, Yi, and So, Sung-Sau. "Overview of Artificial Neural Networks". In: *Artificial Neural Networks: Methods and Applications*. Ed. by David J. Livingstone. Methods in Molecular Biology™. Totowa, NJ: Humana Press, 2009, pp. 14–22. ISBN: 978-1-60327-101-1.

[80] Chen, Shaochuan, Zhang, Teng, Tappertzhofen, Stefan, Yang, Yuchao, and Valov, Ilia. "Electrochemical-Memristor-Based Artificial Neurons and Synapses—Fundamentals, Applications, and Challenges". In: *Advanced Materials* n/a.n/a (), p. 2301924. ISSN: 1521-4095.

[81] Hopfield, J J. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the National Academy of Sciences* 79.8 (Apr. 1982), pp. 2554–2558.

[82] Ebert, Frederik, Dasari, Sudeep, Lee, Alex X., Levine, Sergey, and Finn, Chelsea. "Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning". In: *Proceedings of The 2nd Conference on Robot Learning*. PMLR, Oct. 2018, pp. 983–993.

[83] Koenig, S. and Simmons, R.G. “Unsupervised learning of probabilistic models for robot navigation”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 3. Apr. 1996, 2301–2308 vol.3.

[84] Ganguly, Shashwat, Ahmed, Afaq, and Wang, Fan. “Optimised building energy and indoor microclimatic predictions using knowledge-based system identification in a historical art gallery”. In: *Neural Computing and Applications* 32.8 (Apr. 2020), pp. 3349–3366. ISSN: 1433-3058.

[85] Hwangbo, Jemin, Lee, Joonho, Dosovitskiy, Alexey, Bellicoso, Dario, Tsounis, Vassilios, Koltun, Vladlen, and Hutter, Marco. “Learning agile and dynamic motor skills for legged robots”. In: *Science Robotics* 4.26 (Jan. 2019), eaau5872. ISSN: 2470-9476.

[86] Albawi, Saad, Mohammed, Tareq Abed, and Al-Zawi, Saad. “Understanding of a convolutional neural network”. In: *2017 International Conference on Engineering and Technology (ICET)*. Aug. 2017, pp. 1–6.

[87] Lipton, Zachary C., Berkowitz, John, and Elkan, Charles. *A Critical Review of Recurrent Neural Networks for Sequence Learning*. Oct. 2015.

[88] Song, Ge, Hong, Seong Hyeon, Kyzer, Tristan, and Wang, Yi. “Energy consumption auditing based on a generative adversarial network for anomaly detection of robotic manipulators”. In: *Future Generation Computer Systems* 149 (Dec. 2023), pp. 376–389. ISSN: 0167-739X.

[89] Sherstinsky, Alex. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. ISSN: 0167-2789.

[90] Sun, Shiqi, Zhang, Yan, Luo, Xusheng, Vlantis, Panagiotis, Pajic, Miroslav, and Zavlanos, Michael M. “Formal Verification of Stochastic Systems with ReLU Neural Network Controllers”. In: *2022 International Conference on Robotics and Automation (ICRA)*. May 2022, pp. 6800–6806.

[91] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT Press, 2016.

[92] Yu, Yong, Si, Xiaosheng, Hu, Changhua, and Zhang, Jianxun. “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures”. In: *Neural Computation* 31.7 (July 2019), pp. 1235–1270. ISSN: 0899-7667.

[93] Liu, Yanli, Gao, Yuan, and Yin, Wotao. “An Improved Analysis of Stochastic Gradient Descent with Momentum”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 18261–18271.

[94] Zhang, Zijun. “Improved Adam Optimizer for Deep Neural Networks”. In: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. June 2018, pp. 1–2.

[95] Li, Ziheng, Zuo, Jiankai, Wu, Jiehong, Song, Yifan, Geng, Shenglin, and Li, Junyi. “Deep Convolution and Gated Recurrent Unit Network for Robot Perceptual Intelligent Recognition”. In: *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*. Apr. 2021, pp. 792–795.

[96] Zhang, Xin, Karkee, Manoj, Zhang, Qin, and Whiting, Matthew D. “Computer vision-based tree trunk and branch identification and shaking points detection in Dense-Foliage canopy for automated harvesting of apples”. In: *Journal of Field Robotics* 38.3 (2021), pp. 476–493. ISSN: 1556-4967.

[97] Zhang, Hongxu, Wang, Fei, Wang, Jianhui, and Cui, Ben. “Robot grasping method optimization using improved deep deterministic policy gradient algorithm of deep reinforcement learning”. In: *Review of Scientific Instruments* 92.2 (Feb. 2021), p. 025114. ISSN: 0034-6748.

[98] Escobar-Naranjo, Juan and Garcia, Marcelo V. “Self-supervised Learning Approach to Local Trajectory Planning for Mobile Robots Using Optimization of Trajectories”. In: *Intelligent Sustainable Systems*. Ed. by Atulya K. Nagar, Dharm Singh Jat, Durgesh Kumar Mishra, and Amit Joshi. Lecture Notes in Networks and Systems. Singapore: Springer Nature, 2023, pp. 741–748. ISBN: 978-981-19766-0-5.

[99] Kober, Jens and Peters, Jan. “Imitation and Reinforcement Learning”. In: *IEEE Robotics & Automation Magazine* 17.2 (June 2010), pp. 55–62. ISSN: 1558-223X.

[100] Tan, Kaige, Ji, Qinglei, Feng, Lei, and Törngren, Martin. “Shape Estimation of a 3D Printed Soft Sensor Using Multi-Hypothesis Extended Kalman Filter”. In: *IEEE Robotics and Automation Letters* 7.3 (July 2022), pp. 8383–8390. ISSN: 2377-3766.

[101] Tan, Kaige, Ji, Qinglei, Feng, Lei, and Törngren, Martin. “Edge-Enabled Adaptive Shape Estimation of 3-D Printed Soft Actuators With Gaussian Processes and Unscented Kalman Filters”. In: *IEEE Transactions on Industrial Electronics* 71.3 (Mar. 2024), pp. 3044–3054. ISSN: 1557-9948.

Appendix - Contents

Chapter A

Detailed Explanations and Supplementary Results

In this chapter, the detailed explanations of the concepts discussed in the main text and present supplementary results that contribute to a clearer overview of the key outcomes of this thesis. These additional findings provide valuable context and insights that complement the primary research findings.

A.1 Quaternions to Euler Angles

Even though Euler angles are used to represent rotations, the transmission of two poses are calculated by Quaternion, which is a mathematical construct that extends the notion of complex numbers to four dimensions. They comprise four components: a scalar part represented by w, and a vector component denoted as $\vec{v} = (x, y, z)$. Quaternions are expressed as:

$$q = w + xi + yj + zk$$

where x, y, z, and w are real numbers, and i, j, and k are three imaginary units that satisfy the following multiplication rules: $i^2 = j^2 = k^2 = ijk = -1$. This approach involves establishing the initial pose of the robot as a quaternion and subsequently updating the current pose through sensors like accelerometers and gyroscopes. By computing the difference between the initial and current poses, the rotational component of the transformation can be extracted. This rotational component is often represented as a 3x3 rotation matrix R, which can then be used to derive the Euler angles 'yaw', 'pitch', and 'roll', providing insights into the robot's orientation in three-dimensional space.

$$R = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{bmatrix}$$

The conversion from the rotation matrix 'R' to Euler angles 'yaw', 'pitch', and 'roll' involves the following mathematical expressions:

$$\begin{aligned} \text{yaw (Y)} &= \text{atan2}(R_{21}, R_{11}) \\ \text{pitch (X)} &= -\text{asin}(R_{31}) \\ \text{roll (Z)} &= \text{atan2}(R_{32}, R_{33}) \end{aligned}$$

A.2 Kalman Filter Update

The Kalman filter, a fundamental tool in state estimation for dynamic systems, continuously refines its state estimate in the presence of noisy measurements. This section provides an overview of the Kalman filter's update process, shedding light on its predictive and corrective steps.

State Transition Model (Predictive Step): In the predictive step, the Kalman filter employs the system's dynamics to forecast the next state ($x_{k|k-1}$) based on the previous state estimate ($x_{k-1|k-1}$). This forecast hinges on a linear transformation captured by the state transition matrix (A):

$$x_{k|k-1} = A \cdot x_{k-1|k-1} + Bu_{k|k}$$

The state covariance ($P_{k|k-1}$) is also updated to account for the inherent uncertainty in this prediction:

$$P_{k|k-1} = A \cdot P_{k-1|k-1} \cdot A^T + Q$$

Here, Q represents the process noise covariance matrix, characterizing the uncertainty introduced by the system dynamics.

Measurement Update (Corrective Step): In the corrective step, measurements (z_k) are incorporated into the state estimate to enhance its accuracy. The Kalman gain (K_k) is computed to determine the weight assigned to the measurement versus the prediction:

$$K_k = P_{k|k-1} \cdot H^T \cdot (H \cdot P_{k|k-1} \cdot H^T + R)^{-1}$$

In this equation, H represents the measurement matrix, mapping the state to the measurement space. The measurement noise covariance matrix (R) captures the uncertainty associated with sensor measurements.

Leveraging the Kalman gain, the state estimate and its covariance are updated as follows:

$$x_{k|k} = x_{k|k-1} + K_k \cdot (z_k - H \cdot x_{k|k-1})\ P_{k|k} = (I - K_k \cdot H) \cdot P_{k|k-1}$$

In these equations, I denotes the identity matrix. The Kalman filter's iterative process continually refines the state estimate and its associated uncertainty, making it a powerful tool for state estimation in the presence of noise and uncertainty.

A.3 ANCOVA

Analysis of covariance (ANCOVA) is a powerful statistical tool that allows us to rigorously assess the impact of our independent variables while controlling for covariates. In our case, the covariate of interest is the reference walking speed (v_{ref}), which could potentially influence the dependent variables. The DoF represent the number of values in the final calculation of a statistic that are free to vary, and Sum of Squares(Sum Sq) quantifies the variability in our data and helps us understand the contribution of different factors to that variability. Mean square is the average of the sum of squares values. It is a key statistic in ANCOVA, as it allows us to compare the variation explained by the model to the unexplained variation (error). We calculate Mean Square Model (MSM) and Mean Square Error (MSE). The F-statistic is a crucial test statistic in ANCOVA. It quantifies the ratio of the variation explained by the model (MSM) to the unexplained variation (MSE). The F-statistic is used to determine whether the independent variable (in our case, the RL method: MBRL or MFRL) has a significant effect on the dependent variables. Probability (p) values associated with the F-statistic indicate the likelihood of observing such a statistic if the null hypothesis is true. In our case, a low p-value suggests that the RL method has a significant effect. The Pro>F value indicates the probability of obtaining an F-statistic as extreme as the one observed, given that the null hypothesis is true. Here is how does these measures calculated:

- DoF: (For the model) $\text{DoF}_{\text{Model}} = \text{Number of groups (RL methods)} - 1$, (For the error) $\text{DoF}_{\text{Error}} = \text{Total number of observations} - \text{DoF}_{\text{Model}} - 1$
- Sum of Squares (Sum Sq):SST $= \sum (y_i - \bar{y})^2$, where y_i is each individual data point, and $\bar{y}$ is the overall mean.
- Model Sum of Squares (SSM): SSM $= \sum (n_i)(\bar{y}_i - \bar{y})^2$, where n_i is the number of observations in each group.
- Error Sum of Squares (SSE): SSE $= \sum \sum (y_{ij} - \bar{y}_i)^2$, where y_{ij} is each individual data point within each group.
- Mean Square Model (MSM): MSM $= \frac{\text{SSM}}{\text{DoF}_{\text{Model}}}$
- Mean Square Error (MSE): MSE $= \frac{\text{SSE}}{\text{DoF}_{\text{Error}}}$
- F-statistic: F-statistic $= \frac{\text{MSM}}{\text{MSE}}$

A.4 Single Step Prediction of two NNs

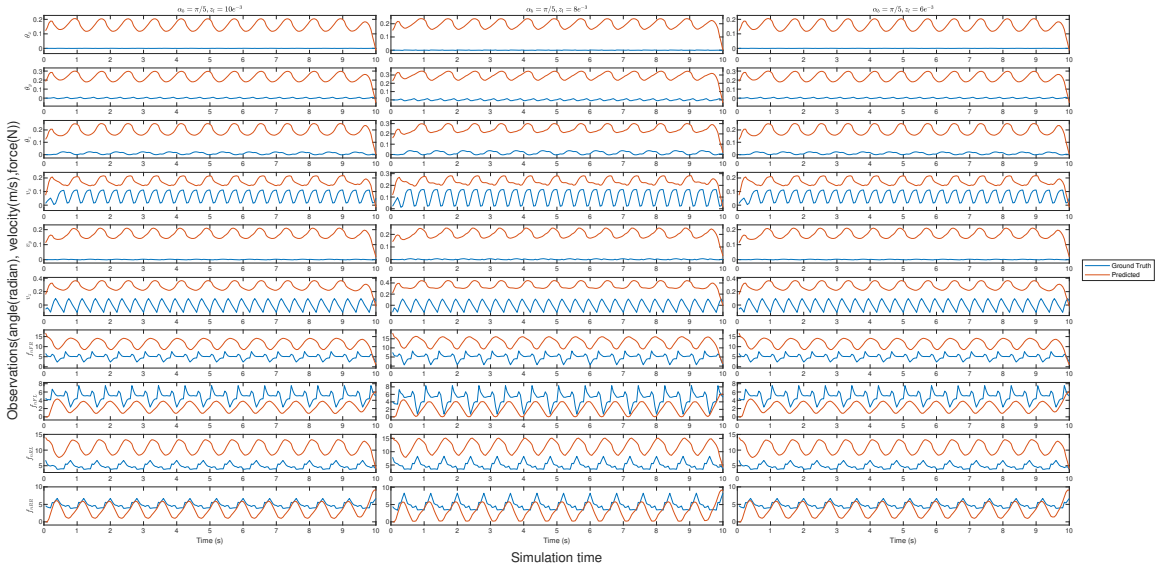


Figure A.4.1: Single step prediction test of LSTM network

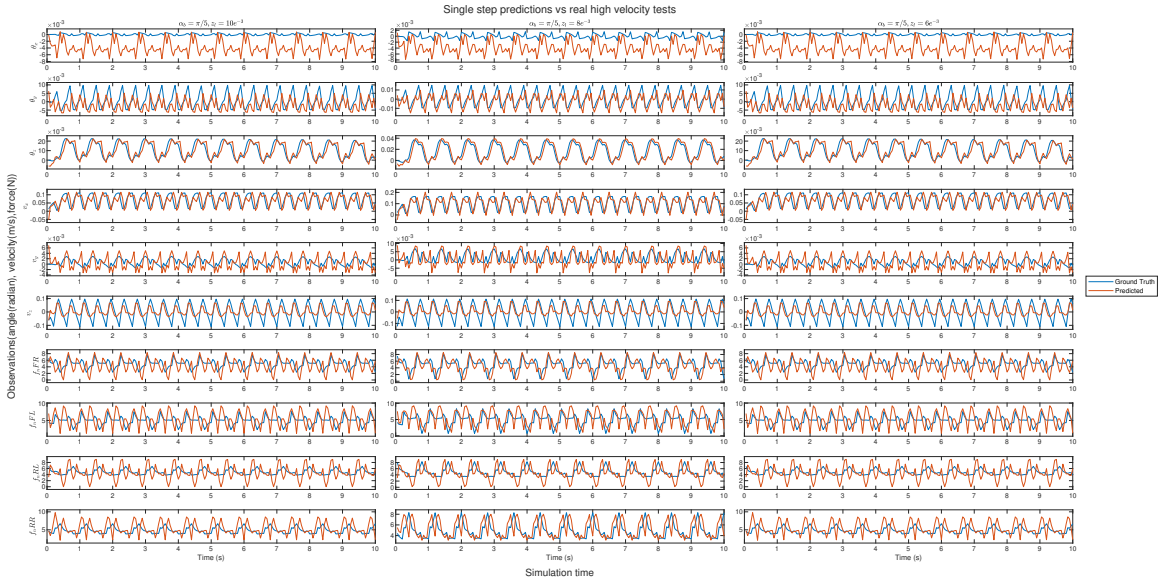


Figure A.4.2: Single step prediction test of DNN

A.5 Expert Gait Walking in Validation

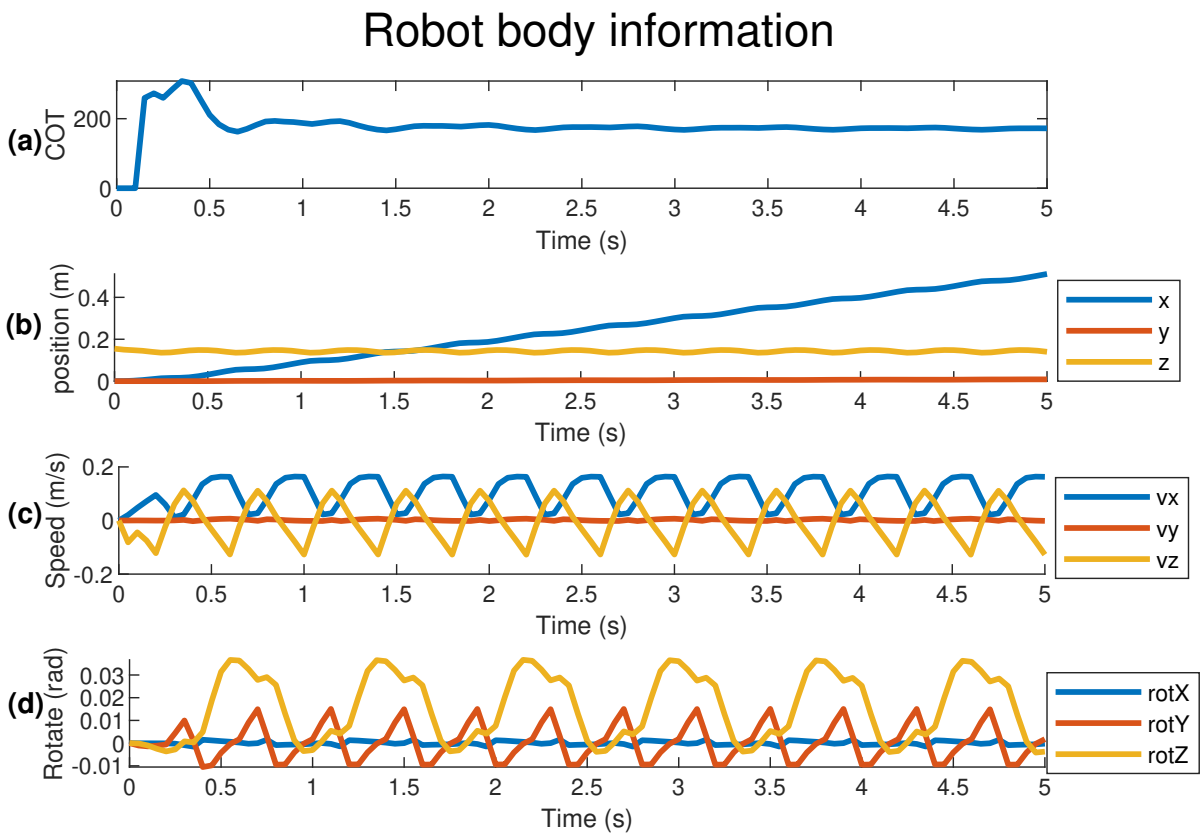


Figure A.5.1: Expert Gait Tested in Validation Results. (a) COT of the robot while walking; (b) Distance traveled along the x, y, and z axis; (c) Speed of the robot along the x, y, and z axis; (d) Orientation of the robot along the x, y, and z axis.

Chapter B

Additional Showcases

In this chapter, the main the source code and block diagrams implemented in Simulink is provided, together with some videos of validating the trained agents. These elements play a crucial role in this thesis, enabling the replication of experiments, validation of results, and the potential for further advancements in our work.

B.1 Main Code of SoftQ

There are two main codes used for controlling and running the robot, Python code are designed for the main brain. Simulink generated C code for controller, but other than that, main code need for training MBRL and MFRL are shown.

Python Code for SoftQ

The Python code for SoftQ can be found in here: Python Code

Matlab Code for MBRL

The Matlab code for MBRL training can be found in here: MBRLmain Code

The Matlab code for creating RL training agent can be found in here: createAgent Code

The source code in Matlab for research questions 1 could be found in here: RQ 1 Code

The source code in Matlab for research questions 2 could be found in here: RQ 2 Code

B.2 Block Diagrams from Simulink

Block Diagrams for Collecting Stochastic Data.

The block diagrams for MBRL training and simulation in simulink could be found in here: Stochastic data acquisition block diagram

Block Diagrams for MBRL

The block diagrams for MBRL training and simulation in simulink could be found in here: MBRL training block diagram

Block Diagrams for Continue Training and MFRL

The block diagrams for continue training and MFRL in simulink could be found in here: Continue training and MFRL block diagram

Block Diagrams for Simulink Controller

The block diagrams for real time controller implemented in Simulink could be found in here: Real time controller block diagram

B.3 Video from Simulation and Field Tests

MBRL with Continous Training Validation and Field Test

The video for MBRL with continous training validation in simulation and field test could be found in here: Bilibili or YouTube

Predefined Expert Gait Validation and Field Test

The video for predefined expert gait validation in simulation and field test could be found in here: Bilibili or YouTube