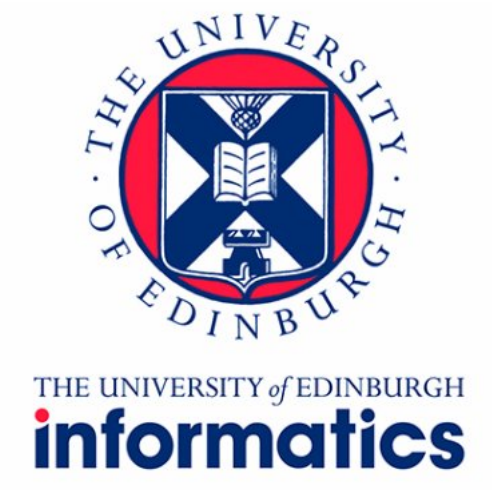
THE UNIVERSITY
of EDINBURGH

Learning Dynamic Motor Skills for Terrestrial Locomotion

By CHUANYU YANG

First Supervisor: ZHIBIN LI
Second Supervisor: TAKU KOMURA



School of Informatics
UNIVERSITY OF EDINBURGH

2020

ipab | Institute of Perception, Action and Behaviour

Abstract

The use of Deep Reinforcement Learning (DRL) has received significantly increased attention from researchers within the robotics field following the success of AlphaGo, which demonstrated the superhuman capabilities of deep reinforcement algorithms in terms of solving complex tasks by beating professional GO players. Since then, an increasing number of researchers have investigated the potential of using DRL to solve complex high-dimensional robotic tasks, such as legged locomotion, arm manipulation, and grasping, which are difficult tasks to solve using conventional optimization approaches.

Understanding and recreating various modes of terrestrial locomotion has been of long-standing interest to roboticists. A large variety of applications, such as rescue missions, disaster responses and science expeditions, strongly demand mobility and versatility in legged locomotion to enable task completion. In order to create useful physical robots, it is necessary to design controllers to synthesize the complex locomotion behaviours observed in humans and other animals.

In the past, legged locomotion was mainly achieved via analytical engineering approaches. However, conventional analytical approaches have their limitations, as they require relatively large amounts of human effort and knowledge. Machine learning approaches, such as DRL, require less human effort compared to analytical approaches. The project conducted for this thesis explores the feasibility of using DRL to acquire control policies comparable to, or better than, those acquired through analytical approaches while requiring less human effort.

In this doctoral thesis, we developed a Multi-Expert Learning Architecture (MELA) that uses DRL to learn multi-skill control policies capable of synthesizing a diverse set of dynamic locomotion behaviours for legged robots. We first proposed a novel DRL framework for the locomotion of humanoid robots. The proposed learning framework is capable of acquiring robust and dynamic motor skills for humanoids, including balancing, walking, standing-up fall recovery. We subsequently improved upon the learning framework and design a novel multi-expert learning architecture that is capable of fusing multiple motor skills together in a seamless fashion and ultimately deploy this framework on a real quadrupedal robot. The successful deployment of learned control policies on a real quadrupedal robot demonstrates the feasibility of using an Artificial Intelligence (AI) based approach for real robot motion control.

Lay summary

In this doctoral thesis, we aim to achieve dynamic terrestrial locomotion of legged robots in the complex environments within the real-world. To achieve this goal, we developed a novel multi-expert DRL framework to learn the multi-skill policy capable of performing a diverse set of dynamic locomotion skills needed to deal with the unexpected changes and disturbances in real-world environments. The research work presented within the thesis can be divided into the following three stages.

- *Part I: Validating the Feasibility of Deep Reinforcement Learning Based Control*

  In the first stage, we begin by investigating the feasibility of using DRL for robotic control by designing a DRL based control framework for a simplified biped in a 2D simulation environment as a proof of concept. The DRL based control framework acquires control policies that are capable of performing balancing behaviours, such as ankle push-offs for humanoid robots, without explicit human design of the controllers, demonstrating the capability of DRL based control methods.

- *Part II: Learning Individual Motor Skills*

  In the second stage, we then proceeded to implement DRL in a 3D environment with realistic torque and velocity limits to learn the three essential motor skills for bipedal locomotion which are balancing, walking, and fall recovery. We first designed a learning framework for balancing that is capable of balancing in the sagittal and lateral plane. We subsequently improved upon the learning framework for balancing and added imitation learning and new neural network structures to obtain a learning framework for bipedal locomotion. The framework is then further improved to incorporate upper body movements to enable the learning of a standing-up policy for fall recovery. The three motor skills will be used as a kickstart to speed up the development of the multi-skill policy.

- *Part III: Multi-Skill Locomotion of Real Quadruped*

  In the final stage, we developed a Multi-Expert Learning Architecture (MELA) to incorporate balancing, walking, and fall recovery motor behaviours into a single unified policy. The proposed MELA framework is able to generate various diverse motor skills. The

multi-skill policy obtained from the proposed MELA learning framework has been deployed successfully on a real quadrupedal robot. The successful deployment of learned control policies on in a real quadrupedal robot demonstrates the feasibility of using an AI-based approach for real robot motion control.

Dedication and acknowledgements

I would like to express my gratitude to my supervisor Dr. Zhibin Li and my secondary supervisor Dr. Taku Komura for their guidance and support throughout the three and half years of my PhD. I would also like to thank Chao Li from DeepRobotics Co. Ltd. and Qiuguo Zhu from Zhejiang University for providing me the opportunity to conduct experiments on the Jueying quadruped robot. I would also like to thank Kai Yuan, Wolfgang Merkt, Wanming Yu and Shuai Heng for their collaboration in my published papers. Finally, I would like to express my utmost gratitude to my family for their support and love.

Publications

First authored publications:

1. **Yang, Chuanyu**, Taku Komura, and Zhibin Li. "Emergence of human-comparable balancing behaviours by deep reinforcement learning." *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017.

2. **Yang, Chuanyu**, Kai Yuan, Wolfgang Merkt, Taku Komura, Sethu Vijayakumar, and Zhibin Li. "Learning whole-body motor skills for humanoids." *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018.

3. **Yang, Chuanyu**, Kai Yuan, Shuai Heng, Taku Komura, and Zhibin Li. "Learning natural locomotion behaviors for humanoid robots using human bias." *2020 IEEE Robotics and Automation Letters*. IEEE, 2020.

4. **Yang, Chuanyu***, Kai Yuan*, Qiuguo Zhu, Wanming Yu, and Zhibin Li. "Multi-expert learning of adaptive locomotion behaviours." *Revision under review*.

5. **Yang, Chuanyu**, Wanming Yu, Quentin Rouxel, and Zhibin Li. "Generating locomotion recovery behaviours for terrestrial robots by responsive motor skills." *Manuscript in preparation*.

Co-authored publications:

1. Song, Doo Re, **Chuanyu Yang**, Christopher McGreavy, and Zhibin Li. "Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge." *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018.

2. Yuan, Kai, Christopher McGreavy, **Chuanyu Yang**, Wouter Wolfslag, and Zhibin Li. "Decoding Motor Skills of Artificial Intelligence and Human Policies: A Study on Humanoid and Human Balance Control." *IEEE Robotics & Automation Magazine* (2020).

3. Sun, Zhaole, Kai Yuan, Wenbin Hu, **Chuanyu Yang**, and Zhibin Li. "Learning Pregrasp Manipulation of Objects from Ungraspable Poses." In 2020 IEEE international conference on robotics and automation (ICRA). IEEE, 2020

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, an overview of the thesis is provided. We begin by providing a basic introduction to the research topic. We then list the problem statements and research objective, followed by our research contributions to the problem statements. Finally, we present an outline of the rest of the thesis.

Humans and animals are able to navigate through terrains in an agile manner using a collection of motor skills and are thus commonly used as inspiration for designing legged robots for movement over complex terrains. For example, humanoid robots have a morphology similar to that of humans and are designed to traverse complex and dynamic environments easily accessible by humans. Legged robots generally exhibit high maneuverability and flexibility and are thus capable of achieving stable locomotion while navigating through uneven terrain and stepping over obstacles. Considering the physical limitations of a wheeled robot, there are many advantages to choosing legged locomotion over wheeled locomotion, as legged robots are able to traverse complex terrains, such as stairs, gaps, and obstacles. Moreover, knowledge of legged locomotion can also help us design prosthetic limbs and exoskeletons capable of producing human-like natural walking gaits, which will be of great benefit to people with gait pathologies.

Honda's ASIMO humanoid robot had a large impact on robotic locomotion. Research on humanoid locomotion has boomed since ASIMO was unveiled in 2000. Currently, there are multiple high-performance humanoid robots that have been developed by different universities and institutions, such as Valkyrie, HRP2+, Walkman, and Atlas, as seen in Figure 1.1.

Despite the advances in hardware design for humanoid robots, the locomotion performances of humanoid robots are still not robust enough to be used in real world environments and are restricted to structured laboratory environments. This situation is partly due to their unstable inverted pendulum structure, which makes them prone to falling, a failure on full display at the DARPA Robotics Challenge. Multi-legged robots, such as quadruped robots, exhibit better locomotion performances in terms of speed, energy efficiency, and obstacle negotiation skills [2]. The most publicly renowned quadruped is Spot from Boston Dynamics.

Figure 1.1: Humanoid robots; from left to right, Atlas, NASA Valkyrie, Walkman, and HRP2.

Figure 1.2: Quadruped robots; from left to right, Spotmini, Anymal, and Jueying.

Apart from Spot, there are other high-performance quadruped robots that have been developed by different universities and institutions, such as Anymal and Jueying, as seen in Figure 1.2.

Bipedal locomotion is a very difficult problem to tackle, as it takes a great deal of effort for a humanoid to maintain stability and not fall over. Currently, bipedal locomotion is mainly accomplished via analytical engineering approaches. However, engineering-based analytical approaches require a lot of substancial understanding of locomotion in terms of designing the controllers and additional efforts in tuning, which is a disadvantage. A majority of analytical approaches produce unnatural behaviours, such as keeping the knee bent and the foot constantly flat on the ground. These unnatural behaviours are due to an engineering attempt to avoid the singularity of a straight knee and under-actuation caused by foot tilting. Usually, walking gaits that exhibit these behaviours use more energy compared to more human-like

walking gaits.

Machine learning approaches, e.g., reinforcement learning (RL), require less human effort compared to analytical approaches. Although RL also requires a certain amount of human knowledge, the main human effort is directed towards the construction of the RL agent and reward, instead of the explicit controllers. Once the proper agent and reward are constructed, the agent will be capable of learning the optimal policy by itself. The effort spent on designing learning agents and rewards is relatively less than the effort spent on designing controllers using analytical approaches. Regarding the issue of unnatural and inefficient motion behaviours, for RL, as long as the reward is designed properly, it will be able to learn a diverse set of more natural, human-comparable behaviours through exploration.

The success of AlphaGo has jump started the robotics research community's interest in Deep Reinforcement Learning (DRL), as it has demonstrated the capabilities of DRL algorithms in solving complex tasks by beating professional GO players. The recent announcement of the success of AlphaGo Zero serves as a further indication of the potential of DRL. Various works regarding novel DRL algorithms capable of working in continuous state and action spaces authored by researchers from OpenAI and DeepMind have also demonstrated the capability of DRL in solving highly complex and dynamic motor-control tasks. Given the increasingly more powerful DRL algorithms being developed, an increasing number of research works have used DRL to solve control tasks and are beginning to examine the possibility of utilizing RL to deal with continuous-control tasks involving complicated dynamics.

1.1 Terrestrial Locomotion using Deep Reinforcement Learning

In recent years, there has been a growing interest in using DRL algorithms to solve complex control tasks. Recent developments in DRL have shown that DRL can be a possible alternative to analytical approaches. Quite a few studies have been focusing on using DRL for the locomotion of bipedal or humanoid characters in physical simulation. Research done by OpenAI and DeepMind has demonstrated the capabilities of DRL with respect to learning complex and dynamic behaviours for humanoids in a simulated environment.

Researchers in the computer science and robotics community have published papers on using DRL for both humanoid motion control and quadruped control, as bipedal and quadrupedal locomotion are typical cases of legged terrestrial locomotion. Peng et al. successfully applied Continuous Actor Critic Learning Automaton (CACLA) [3], [4] to train a bipedal character to learn terrain traversal skills for a terrain with gaps and walls [5]. Later, they developed a hierarchical DRL framework with a low-level controller (LLC) specializing in balance and limb control and a high-level controller (HLC) focusing on navigation and trajectory planning. Using their framework, the bipedal character successfully learned the skills necessary to perform tasks such as guiding a soccer ball, following a path, and avoiding an obstacle [6].

Apart from basic locomotion, DRL has also been used to learn more complex and diverse motions. Kumar et al. used DRL to learn a safe falling strategy for humanoids to minimize damage during falls. Their algorithm is based on CACLA and the Mixture of Actor-Critic Experts (MACE) architecture [7]. In this architecture, each joint is assigned with an independent actor-critic pair. The actor with the highest corresponding critic value is activated to generate action. This architecture combines both continuous and discrete controls. Peng et al. combined imitation learning and RL to learn dynamic motions, such as back flips, side rolls, jumping and so forth, for simulated humanoids and quadrupeds [8], [9].

Learning locomotion policies for legged locomotion skills will prove to be valuable alternatives to engineering-based approaches as machines become more prevalent in unstructured environments because it is very difficult to consider all a priori needs during the manual design process for a controller. DRL alleviates the tedious manual efforts required in designing controllers by automating the entire process through trial and error without human intervention. With DRL, it is possible to build complex systems capable of processing rich, high-dimensional sensory information as well as responding robustly and adapting to unfamiliar situations and unstructured environments with minimal human effort.

1.2 Problem Statement

Humans and Animals are able to able to traverse over complex terrains in the real-world in an agile and robust manner. The main objective of the thesis project is to design a controller to replicate the terrestrial locomotion capabilities of humans and animals over complex terrains on artificial robot agents. In order to achieve the same locomotion capabilities of humans and animals, the controller needs to be able to perform multiple different skills to deal with the unexpected changes and disturbances in real-world environments. We propose a novel DRL framework to learn the multi-skill controller capable of performing diverse set of dynamic locomotion skills necessary for traversing over complex and unstructured terrains in the real-world.

A three stage research plan was drawn up to achieve the goal of learning the multi-skill control policy. The first stage investigates the capability of DRL using a toy example in a simplified physics simulation. The second stage focuses on learning basic locomotion skills individually in more realistic 3D simulation. The final stage will use the learned basic locomotion skills as a kickstart, and merge the skills into a single unified multi-skill control policy.

1.3 Thesis Overview and Contribution

The thesis consists of eight chapters, with chapter 1 presenting the motivation, objective and aim of the thesis and chapter 2 summarizing the background knowledge and related work. Chapters 3 to 7 constitute the main part of the thesis, cover the research work from five papers,

and are organized following the order of research. Chapter 8 concludes with the limitation of the work and discusses potential future research directions. To give the thesis a more coherent structure and improve the readability, we divided the research work presented in chapters 3 to 7 into three parts, which also coincides with the three stage research plan mentioned above.

- *Part I: Validating the Feasibility of Deep Reinforcement Learning Based Control*

  In part I, we designed a learning framework based on the state-of-the-art DRL algorithm, which was then deployed in a simplified 2D simulation environment to validate the feasibility of learning human-comparable motor skills with DRL.

- *Part II: Learning Individual Motor Skills*

  In part II, we focused on designing various learning frameworks to learn multiple fundamental motor skills for humanoid and quadruped robots (namely balancing, walking, and fall recovery) individually in a physically realistic simulation environment. The learned motor skills will serve as a kickstart for the multi-skill control policy.

- *Part III: Multi-Skill Locomotion of Real Quadruped*

  In part III, we presented a novel Multi-Expert Learning Architecture (MELA) that is able to fuse multiple different individual motor skills into a single multi-skill control policy. We have further demonstrated MELA's effectiveness by successfully deploying the learned multi-skill control policy on a real quadruped.

1.3.1 Validating the Feasibility of Deep Reinforcement Learning Based Control

Chapter 3 In this chapter, we verify the feasibility of using DRL for continuous motor- control tasks with a simplified humanoid model in a 2D simulation environment. This chapter presents a hierarchical framework based on DRL that naturally acquires control policies capable of performing balancing behaviours, such as ankle push-offs in humanoid robots, without explicit human design of the controllers. The reward for training the neural network is specifically formulated based on physical principles and quantities, rendering it explainable. Furthermore, the successful emergence of human-comparable behaviours through DRL demonstrates the feasibility of using an AI-based approach for humanoid motion control in a unified framework. Moreover, the emergence of human-like dynamic balancing behaviours proves the feasibility and potential of using DRL to obtain complex and dynamic motor skills for robotic control, suggesting a research direction for using learning-based controls to explore motor skills for optimal performance. The contents of this chapter are presented in the paper "Emergence of human-comparable balancing behaviours by deep reinforcement learning" published in the Humanoids 2017 conference proceedings.

Contribution The initial idea of using deep reinforcement learning to learn control policies for balancing was provided by the supervisor. The author built upon the initial idea and realised it in simulation. The author has set up the entire learning framework individually, which includes the simulation environment and the DRL algorithm. The author also conducted the data analysis and written the first draft of the paper.

1.3.2 Learning Individual Motor Skills

Chapter 4 In this chapter, we present a DRL framework for obtaining the motor skills for balancing. The presented framework has a hierarchical control structure and is capable of acquiring motor skills for a variety of push recovery and balancing behaviours, i.e., ankle, hip, foot tilting, and stepping strategies. The policy is trained in a physics simulator with realistic settings of the robot model and low-level impedance control that make it easy to transfer the learned skills to real robots. The advantage over traditional methods is the integration of high-level planner and feedback control within a single coherent policy network, which is generic in terms of learning versatile balancing and recovery motions against unknown perturbations at arbitrary locations (e.g., legs, torso). Furthermore, the proposed framework allows the policy to be learned quickly by many state-of-the-art learning algorithms. The contents in this chapter are presented in the paper "Learning whole-body motor skills for humanoids" published in the Humanoids 2018 conference proceedings.

Contribution The majority of the work were done by the author. The author proposed the idea of using DRL to learn bipedal balancing policies in 3D simulation environment. The author also handled the bulk of the work in setting up the DRL framework, conducting the data analysis, and writing the paper. Kai Yuan contributed equally on data analysis and paper writing. Wolfgang Merkt has significant contribution on setting up the 3D simulation environment.

Chapter 5 In this chapter, we present the DRL framework for obtaining the motor skills for walking. The presented learning framework leverages knowledge from imitation learning, DRL, and control theories to achieve a human-style walking motion for the locomotion of humanoids that is natural, dynamic, and robust. We proposed two novel approaches to introduce human bias, i.e., motion-capture data and a special multi-expert network structure. We used the multi-expert network structure to blend behavioural features smoothly and use an augmented reward design for the task and imitation rewards. Our proposed reward design principle is composable, tunable, and explainable because it uses fundamental concepts from conventional humanoid control. We rigorously validated and benchmarked the learning framework, and find that it consistently produces robust locomotion behaviours in various test scenarios. Further, we demonstrated its capability of learning robust and versatile policies in the presence of disturbances, such as terrain irregularities and external pushes. The content in this chapter is

presented in the paper "Learning natural locomotion behaviours for humanoid robots using human bias" published in the journal RA-L.

Contribution The majority of the work were done by the author. The author proposed the idea of combining Imitation Learning (IL) and DRL to learn human-like walking gaits for simulated humanoids. The author also came up with the idea of using multi-expert network structures for walking gaits together with Kai Yuan. The author handled the bulk of the work in setting up the DRL framework, conducting the data analysis, and writing the paper. Kai Yuan provided useful insights on multi-expert network structures, and contributed equally to the data analysis and paper writing. Shuai Heng assisted in the comparison studies and data analysis.

Chapter 6 In this chapter, we investigate the potential of using DRL to learn challenging fall recovery maneuvers involving multiple contacts. The ability to recover from a failure caused by a fall is an essential skill for traversing rugged terrains successfully. A common approach is to handcraft a trajectory, which requires significant engineering effort. Handcrafted trajectories also suffers from lack of generalization, cannot be used for different robot models, and are prone to failure in corner cases. In this chapter, we presented a framework based on model-free DRL to learn a control policy that is able to utilize its entire body, including all limbs, to generate a standing-up recovery maneuver. The learning-based approach requires minimal human engineering effort and is capable of generalizing across robots with different morphologies, i.e., bipeds and quadrupeds. The effectiveness of the framework is further validated by the robust performance of the policy while being deployed on a real-world quadruped. The contents of this chapter will be presented in a future paper with the title "Generating locomotion recovery behaviours for terrestrial robots via responsive motor skills." which is currently under preparation.

Contribution The initial idea of using DRL to learn fall recovery control policies for humanoids and quadrupeds was provided by the supervisor. The author built upon the initial idea and realised it in simulation. The author handled the bulk of the work in setting up the DRL framework, conducting the data analysis, and writing the paper. Wanming Yu contributed equally to the data analysis and paper writing, and worked on part of the simulation setup.

1.3.3 Multi-Skill Locomotion of Real Quadruped

Chapter 7 In this chapter, we present a Multi-Expert Learning Architecture (MELA) that is capable of fusing the motor skills involved in balancing, locomotion, and fall recovery all into one unified control framework. Achieving versatile and adaptive skills for robot locomotion is challenging and can underpin a wide range of mobility solutions, Furthermore, developing algorithms for acquiring new and effective skills is essential for enabling legged robots to

be flexible in various scenarios. The proposed MELA is dynamically fuses multiple deep neural networks (DNNs) into one synthesized DNN to generate new skills for adapting to new situations. MELA first learns the core skills for distinct tasks via bootstrapping in separate deep neural networks (DNNs), then trains all DNNs to refine more advanced skills across various locomotion modes, including all the dynamic transitions that fall in between. During runtime, MELA constantly recombines multiple DNNs and fuses a new synthetic DNN dynamically to generate new behaviours in response to changing situations. The proposed MELA framework has produced successful multi-skill locomotion in a real quadruped robot that performed coherent trotting, steering, and fall recovery autonomously, indicating the merit of multi-expert learning that adapts flexible motor skills to novel scenarios. The contents in this chapter are presented in the paper "Multi-expert learning of adaptive motor skills," which is currently under review.

Contribution Both the author and Kai Yuan contributed equally to this work, with the author having more contribution on the real-world experiments. The author and Kai Yuan have worked together to come up with the idea of using multi-expert learning structure for robot control. The author focused on designing the learning framework and conducting the data analysis, while Kai Yuan focused on programming the control software. Qiuguo Zhu developed the robot hardware and provided the facilities necessary for the experiment. Wanming Yu contributed to the paper writing. The supervisor has directed the research and has contributed to the majority of the paper writing.

Chapter 2

Literature Review and Background

In this section, we first present the basic background knowledge and concepts related to DRL and legged locomotion that are essential for the understanding of the project. We then present some of the related work that involve using DRL to learn control policies for legged locomotion.

2.1 Background

2.1.1 Reinforcement Learning

Reinforcement learning (RL) as a machine learning algorithm resembles the learning process of animals and is partly inspired by behavioral studies of animal learning psychology. The fundamental concept is that an intelligent agent, either biological or artificial, is able to learn a desirable behavior by rewarding intended outcomes and penalizing an undesired ones in through trial-and-error interactions with the environment.

Apart from the environment and agent, a reinforcement learning system consists of four other main components, namely, (i) policy, (ii) reward, (iii) value function, and optionally, the (iv) model of the environment [10]. The *policy* can be viewed as a mapping from an observed state to the executed action and is the determining factor of the behavior of the agent. The *reward* indicates the performance of the immediate action generated by the policy at a given state observation according to a specified goal. The *value function* is the accumulated reward over the future, and indicates the long-term desirability of the state and action. The *model* describes the state transition within the environment. Given a state and an action, the model is able to predict the resulting next state, which is necessary for long-term planning. Not all reinforcement learning algorithm uses a model, the algorithms that uses model for long-term planning are called model-based reinforcement learning, while those that do not are called model-free reinforcement learning.

2.1.2 Deep Reinforcement Learning

Recent breakthroughs in RL and deep learning have given rise to DRL, which is a combination of RL and deep neural networks. Within DRL, deep neural networks are utilized to approximate the value function $V(s)$, or $Q(s)$, the policy $\pi(s,a)$ and model (state transition and reward) [11].

DRL has enjoyed increased popularity in recent years in its capacity as a non-linear state abstraction tool capable of dealing with action decisions. A great deal of progress has been made in the development of DRL since it was first proposed by Mnih et al. [12]. The rise of DRL has allowed agents to perform more complex and dynamic tasks in high-dimensional continuous state and action spaces. There are wide variety of DRL algorithms dedicated to solving problems in high-dimensional continuous state and action spaces, such as Trust Region Policy Optimization (TRPO) [13], Normalized Advantage Function (NAF) [14], Proximal Policy Optimization (PPO) [15], Asynchronous Advantage Actor Critic (A3C) [16], Soft Actor Critic (SAC) [17], and Deep Deterministic Policy Gradient (DDPG) [18]. Meanwhile, SAC and PPO are the most commonly used DRL algorithms.

2.1.3 Concepts in Legged Locomotion

The following are the commonly appearing concepts in the research field control and legged locomotion. These concepts will be referred to in the following chapters to be used as a guideline to design the reward function and construct the DRL control framework.

2.1.3.1 Linear Inverted Pendulum Model

The linear inverted pendulum model is a classical approach that is used to model the dynamics of walking. According to the linear inverted pendulum model, the total mass of the robot is concentrated at a single point mass at the centre of mass (COM); as a result, the COM acts as an inverted pendulum around the stance foot. The COM maintains a constant height when it moves along the stance foot [19].

2.1.3.2 Support Polygon

A support polygon is the convex hull (i.e. smallest convex region) that covers all the contact point between the feet of the robot and the ground. It is the region in which the projection of COM along gravity must lie within to achieve static stability.

2.1.3.3 Zero Moment Point

The zero moment point (ZMP) is the point on the ground where the component of the tipping moment created by gravity and inertial forces is tangential to the supporting surface [20]. ZMP was introduced initially by Vukobratovis et al. in 1972 and is used commonly in the motion planning for anthropomorphic gaits in bipedal robots. The concept of centre of pressure (COP)

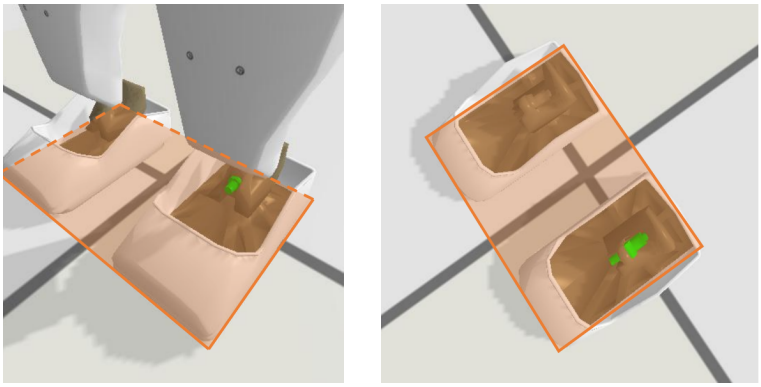


Figure 2.1: The support polygon created by the feet of the Valkyrie humanoid robot while standing.

is also used commonly when researching anthropomorphic gaits. The COP is the point on the ground where the ground reaction force produces a zero moment [20].

The relationship between the ZMP and COP is worth mentioning. The ZMP is defined with respect to gravity plus inertial forces, while the COP is defined with respect to ground-foot reaction forces. Borovac et al. has clarified that the concept of ZMP is only valid when the ZMP lies within the support polygon in the case of a dynamically balanced gait [21]. They state that when the ZMP lies within the support polygon and the biped is dynamically balanced, the COP coincides with the ZMP during flat foot contact [21].

During static walking, the projection of the COM of the robot must lie within the support polygon. During dynamic walking, the projection of the COM of the robot can lie outside of the support polygon for a limited amount of time as long as the ZMP is kept within the support polygon [22].

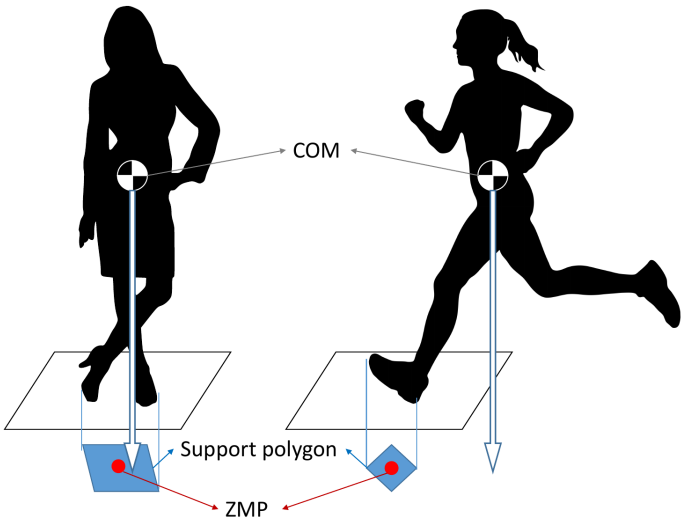


Figure 2.2: Depiction of the support polygon, ZMP and projection of COM.

2.1.3.4 Capture Point

The capture point is another concept used commonly in humanoid locomotion. It is defined as a point on the ground that the robot can step to in order to bring itself to a complete stop [23]. Knowing the velocity and height of the inverted pendulum as well as the gravitational acceleration, we are able to compute the capture point:

$$x_{\text{capture}} = x_{\text{COM}} + \dot{x}_{\text{COM}}\sqrt{\frac{z_c}{g}}, \quad J_{\text{reject}} = m\sqrt{\frac{g}{z_c}}\Delta_{\text{COP}}, \tag{2.1}$$

where the impulse J_{reject} derived by the capture point is the theoretical maximum of the impulse that can be rejected without taking a step, z_c is the COM height, Δ_{COP} is the relative horizontal distance between the edge of the foot and the COM, and m is the total mass of the inverted pendulum [24].

Normally, during humanoid balancing, the COM and COP are considered to stay within the support polygon created by the foot. The capture point, as an indication of balance, is also considered to be within the support polygon, so the maximum reachability of the capture point without taking a step is at the edge of the foot. If the total duration of impulse t_{impulse} and the mass of the humanoid m are known, we can derive other useful physical properties, such as the maximum force F_{max} the robot can withstand and the maximum velocity disturbance V_{max} of the COM the robot can withstand and still be able to balance.

$$V_{\text{max}} = \frac{J_{\text{reject}}}{m}, \quad F_{\text{max}} = \frac{J_{\text{reject}}}{t_{\text{impulse}}} \tag{2.2}$$

2.1.3.5 Hierarchical Structure of High-Level Joint Angle Control and Low-Level Torque Control

The choice of output action parameterization has been proven to have a significant effect on the performance of RL. Peng et al. compared the impact of four different actuation models that has different action parameterization on DRL: 1. direct torque control; 2. muscle activation for musculotendon units (MTU); 3. target joint angle for proportional-derivative controllers; 4. target joint angle velocity. Their study showed that action parameterization that includes basic feedback such as target angle for PD control and muscle activation for MTU can improve policy performance and learning speed [25]. The hypothesis proposed for the explanation of the improved performance of PD control is that a PD controller exhibits spring damping properties that resembles the biomechanics of biological systems, e.g. muscles.

Therefore, instead of directly controlling the joint motor torque, we designed a hierarchical control structure with a neural network as the high-level controller that generates the desired joint angles, and a PD controller as the low-level controller that generates the torque. The PD controller is used to translate joint angles produced by the neural network into joint torques.

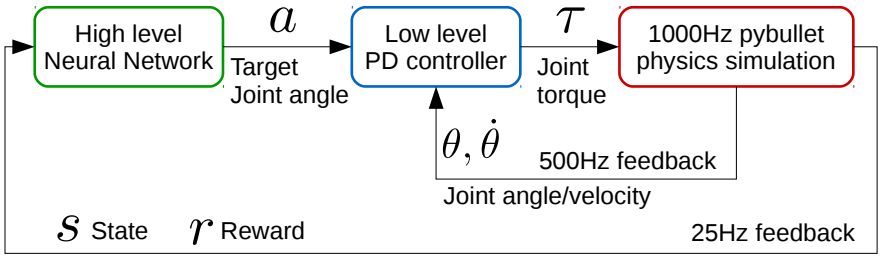


Figure 2.3: Hierarchical control system overview. The high-level neural network generates target joint angles, while the low-level PD controller translate the angles to the target joint torques.

The resulting torque is computed as:

$$u = K_p(q_{\text{target}} - q_{\text{measured}}) - K_d \dot{q}_{\text{measured}}, \tag{2.3}$$

where K_p, K_d are the PD gains respectively, q_{target} is the targeted joint angle, and $q_{\text{measured}}, \dot{q}_{\text{measured}}$ are the measured joint angles and velocities respectively.

2.2 Related Work

In this section, we will do a literature review on the state of the art that involves learning a control policy for legged locomotion. The contribution of our work over state of the art will also be briefly discussed.

2.2.1 Balancing using Reinforcement Learning

Balancing is one of the fundamental motor skills needed for the successful locomotion of legged robots, including bipedal and quadruped robots. During bipedal locomotion, the robot is often subject to large impact forces from external disturbance, such as pushes. The impact force will destabilize the bipedal robot and cause the robot to fall, not only hindering the execution of the task but also damaging the robot. Therefore, it is important to design a controller that is able to recover from push disturbances and regain its balance.

There have been a few research papers that have focused on obtaining a balance controller for the push recovery of bipedal robots using the RL paradigm. Peng et al. designed a balance controller that learns the optimal ankle impedance using integral RL [26]. Zhou et al. proposed a fuzzy RL approach based on a fuzzy neural network [27]. Apart from bipedal balancing, RL has been used to obtain push recovery control policies for quadruped robots [28].

Our work: Compared to previous works that uses RL to learn balancing policies for bipedal robots, we went a step further and achieved natural human-like natural balancing policies.

We have demonstrated that human-like knee lock and toe-tilt balancing behaviours emerge naturally from DRL in a 2D simulation within the sagittal plane [29]. We later extended upon the work and achieved human-comparable 3D balancing behaviors in both the sagittal and lateral planes [30]. The resulting work on bipedal balancing will be presented in detail in Chapter 3 and 4.

2.2.2 Locomotion using Reinforcement Learning

The main goal of the research is to achieve the same natural human and animal like locomotion behaviors on artificial robot agents using DRL. The locomotion behaviors of humans and animals are symmetric and periodic. Therefore, in order to achieve human-like and animal-like locomotion, we need to explore a way to enforce symmetry and periodicity.

2.2.2.1 Enforcing Periodic Motions in Locomotion

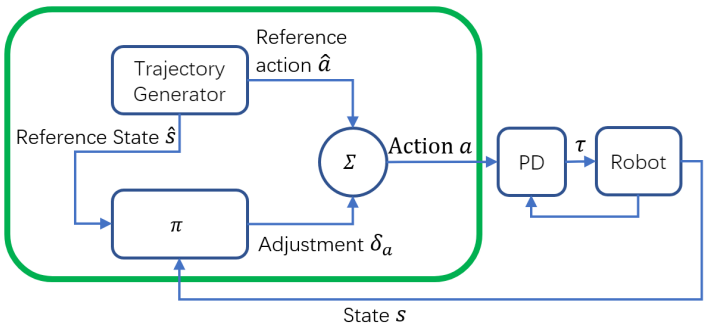
Steady locomotion gait patterns can emerge without any gait phase information being provided as an input or reward when using the DRL algorithm, as shown by various studies [31]. However, most of the locomotion policies generate asymmetric jerky gaits that look unnatural [31]. Further, they are inefficient and feature a lot of unnecessary movements.

Researchers have come up with various solutions to solve those problems. The solutions can be divided roughly into three categories: (1) learning from demonstration, (2) periodic network architecture, and (3) feedback control of a predefined trajectory.

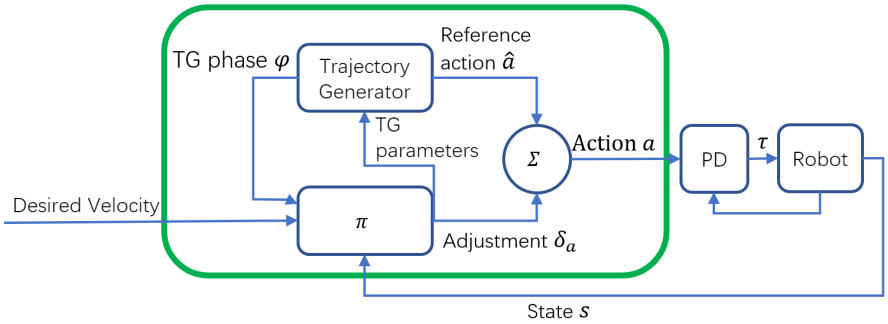
1. *Learning from Demonstration*

The artificial agent can be guided to learn periodic and symmetric walking behaviours by providing human walking motion as an expert demonstration for the agent to learn from. This technique, which extracts information from the reference motion generated by expert demonstrations to guide an agent, is referred to as learning from demonstration. Examples include Behaviour Cloning (BC) [32], Inverse Reinforcement Learning (IRL) [33], Generative Adversarial Imitation Learning (GAIL) [34], and Imitation by Tracking (IT) [35]. BC minimizes the difference between the student and expert behaviour in a supervised learning fashion through a loss function, while IRL fits a reward function to describe the demonstration and seeks to maximize it. GAIL learns a discriminator to measure the similarities between expert demonstrations and behaviours generated by the policy, and the objective of the agent is to learn a policy that is indistinguishable from the demonstration. Finally, IT involves designing a tracking reward dedicated to measuring the similarities between the agent's state and the demonstration data [8], [36], [37].

Furthermore, there are a few research papers in the field of character animation that have utilized learning from demonstration to learn impressive human-like behaviours for humanoids. Merel et al. implemented GAIL [34] to train a neural network policy to produce human-like walking and standing-up behaviours from human motion capture data [38]. Peng et



(a) Feedback control with fixed trajectory generator



(b) Feedback control with adaptive trajectory generator

Figure 2.4: Learning feedback control policies with trajectory generators

al. implemented a tracking reward to imitate various motions, such as walking, performing a backflip, running, and ball throwing, in a simulated humanoid [8] as well as trotting and jumping in quadrupeds [9].

2. *Periodic Network Architecture*

Locomotion is periodic in nature; therefore, many locomotion controllers have structures that produce rhythmic outputs, such as the Central Pattern Generator (CPG). The CPG is used commonly to construct a bio-inspired neural network control policies for bipedal locomotion due to its ability to produce coordinated rhythmic and periodic gaits [39]. However, the CPG is not necessary for designing a periodic controller, as other periodic structures can be used. Holden et al. proposed a special architecture named Phase-Functioned Neural Networks (PFNN) to generate locomotion animations from motion-capture data for computer graphics and have successfully synthesized various human motions. Sharma et al. incorporated the PFNN architecture with DRL and developed a phase-parametric action-value function and a phase-parametric policy to learn locomotion policies in simulation [40].

3. *Feedback Control of a Predefined Trajectory*

Instead of training the agent to learn a natural-looking locomotion gait directly, a trajectory generator can be used to provide a periodic and symmetric locomotion trajectory and allow the agent to focus instead on learning a feedback control loop that fine tunes the trajectory to

adapt to external disturbances. Usually, the trajectory generator is a separate entity and can function individually without the learning agent. Xie et al. used a feedback controller on top of a predefined joint trajectory to control the bipedal Cassie robot in simulation [41]. Their method is able to generalize in terms of different velocities and terrain slope features. Tan et al. has come up with a similar approach for the quadrupedal Minotaur robot by using a trajectory generator that generates a fixed trajectory [42], and the structure of the control framework is shown in Fig. 2.4a. Iscen et al. improved upon the work of Tan et al. and developed a framework that consists of an adaptive trajectory generator and a learned feedback controller that modulates both the output of the trajectory generator and the parameters of the trajectory generator [43], improving the generalization capability of the policy. This framework is illustrated in Fig. 2.4b.

2.2.2.2 Enforcing Symmetric Motions in Locomotion

Animal gaits are symmetric in nature; hence, using policies that learn symmetric motions might introduce benefits in terms of learning speed and task performance. Abdolhosseini et al. have conducted a thorough review on the different methods that can be used to introduce symmetric motion behaviours [44].

The solutions proposed for enforcing symmetry can be roughly divided into four categories: (1) learning from demonstration, (2) feedback control of the predefined trajectory, (3) auxiliary loss, and (4) symmetric network architecture. The first and second approaches are the same approaches that can be used to enforce periodic behaviour and are described in detail in the previous section. In the following sections, we will introduce auxiliary loss and symmetric network architectures, both of which are dedicated solely to enforcing symmetry.

1. *Auxiliary Loss*

An auxiliary loss can be designed to enforce symmetry [45]. Yu et al. proposed the following symmetry auxiliary loss:

$$L_{sym}(\Theta) = \sum_{t=1}^{T} ||\pi_\theta(s_t) - M_a(\pi_\Theta(M_s(s_t)))||^2, \tag{2.4}$$

where $M_a(a)$ is the function that mirrors the action space, and $M_s(s)$ is the function that mirrors the state space. In the paper, the auxiliary loss is added to the proximal policy optimization loss, and the two losses are computed altogether. As opposed to designing a reward to enforce symmetry, the auxiliary loss is differentiable, and the gradient can be propagated back directly into the neural network.

2. *Symmetric Network Architecture*

Symmetry can be enforced by integrating symmetric properties into the network architecture. There are multiple ways of imposing symmetry on the neural network design. Abdolhosseini et al. introduced a universal method that can convert any neural network into a symmetric neural network, which involves designing a neural network architecture containing a policy module

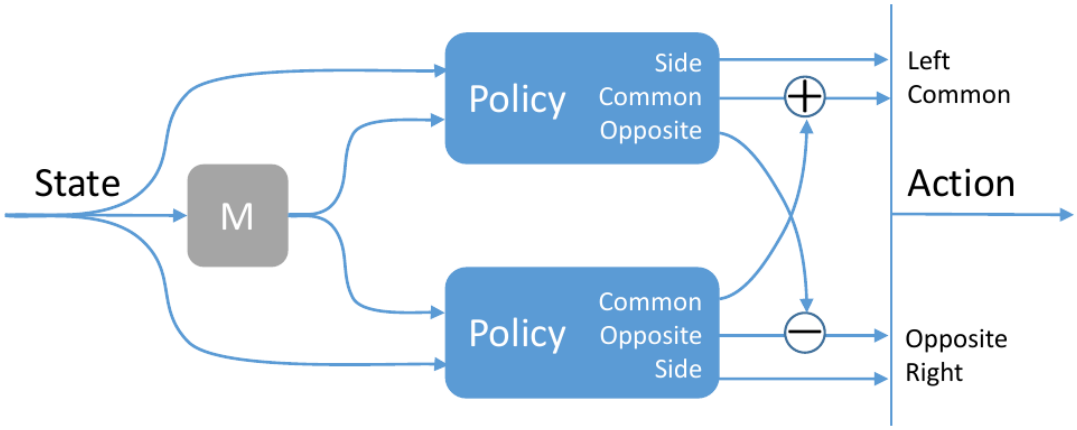


Figure 2.5: Symmetric neural network structure consisting of four types of outputs: left side, right side, common, and opposite.

and its mirrored counterpart [44]. The original and mirrored policy modules are combined to create the symmetric neural network architecture as shown in Fig. 2.5.

For mirroring purposes, Abdolhosseini et al. divided the joints of the robot into three categories: common, opposite, and side. Common joints are joints that are invariant to the mirroring. Opposite joints are joints for which the direction needs to be inverted. Side joints are joints that need to be swapped with their mirroring counterparts. Using a humanoid robot as an example, torso pitch, head pitch and many other joints that rotate around the pitch are common joints. The torso roll and torso yaw joints are opposite joints. The joints on the arms and legs are side joints. Common, opposite, and side joints exhibit different behaviours and need to be treated differently during the mirroring procedure.

The drawback of this method is that it requires sufficient knowledge on the symmetry relations between state and action to design the network properly. Complications also arise in the network design when there are body parts with no mirroring counterparts, such as the head and torso of a humanoid robot.

Our work: We realized that Learning from Demonstration is a simple and effective way of enforcing symmetry and periodicity into the learned control policy. In our work, through the use of a tracking reward, we are able to learn a periodic and symmetric walking motions that closely resembles the demonstrated human walking motion [37]. The tracking reward is crucial for learning the human-like walking motions, without the tracking reward, the DRL agent will learn an asymmetric leaping motion [37]. The resulting work on human-like bipedal walking will be presented in detail in Chapter 5.

2.2.3 Robust Recovery from Fall using Reinforcement Learning

Robots operating in complex environments need to be able to react to unforeseen circumstances and to recover from failure. Although it is possible to generate safe and robust motions with the most advanced state-of-the-art techniques for locomotion, there will always unforeseen circumstances in which the disturbance is too much for the robotic system to handle. First, the robot needs to be able to detect or predict when a fall occurs [46] and perform the appropriate safe falling maneuvers to minimize hardware damage [47], [48], [49]. After the fall, the robot then needs to be able to recover automatically from many different possible configurations, especially when operating in dangerous environments, such as search and rescue, disaster response, and exploration.

There have been various attempts in the robotics community to design a controller for humanoids and quadrupeds that will get them to stand back up and recover from fall. Such a controller would be especially useful in the case of humanoids due to their unstable nature. Standing back up on two feet is not a trivial task for humanoid robots, as it involves multiple contact points with the ground, is difficult to model, and poses many challenges for optimization-based controllers. Humans are particularly good at standing up from a fallen position, as their soft bodies and compliant muscle joints allow for many different stable types of standing-up motions. Humanoid robots, in contrast, are usually lacking some fundamental degrees of freedom present in humans (e.g., in the torso). In addition, their joints lack the necessary ranges and torques required for performing standing-up motions.

One common approach to designing a fall recovery controller for humanoids and quadrupeds is to handcraft the trajectory manually by imitating the recovery motions used by humans and dogs. Stückler et al. have designed a controller for standing up by manually scripting the target joint angles for the entire trajectory of the standing routine [50]. Kanehiro et al. designed a fall-recovery controller for the HRP-P2 robot by designing a graph consisting of the key contact states within the standing motion and devising a ZMP-based controller for the transitions between the contact states. They have deployed the controller successfully on a real HRP robot, achieving the task of standing up from supine and prone positions [51]. Finally, Semini et al. handcrafted a self-righting recovery sequence for the HyQ2MAX quadruped robot manually [52].

The above hard-coded methods require a large amount of effort and human knowledge to design and usually break down in corner cases due to poor generalization. Recent advancements in RL have given rise to more automatic and intelligent methods of obtaining fall recovery control policies. Model-free RL appears to be a promising alternative for solving the problem of fall recovery. With model-free RL, the learning agent is able to obtain the policy through interactions with the environment, avoiding the need to model complex real-world dynamics explicitly. Endeavors have been made within the research community to obtain fall recovery control policies using the RL approach. Jeong et al. applied Q-Learning to solve the problem of

fall recovery for humanoids by discretizing the state and action spaces of the simulated robot [53], while Lee et al. has trained a fall recovery policy applying proximal policy optimization to learn a continuous fall recovery control policy and deployed the policy successfully on a real ANYmal quadruped robot [54].

Our work: The work listed above all focuses on solving fall recovery for a very specific robot model, and does not appear to be generalizable to other robots. In comparison, we developed a versatile DRL framework that is capable of learning fall recovery policies for various different humanoid and quadruped robot morphologies. The resulting work on the generalizable fall recovery learning framework will be presented in detail in Chapter 6.

2.2.4 Deep Reinforcement Learning on Real-World Robots

Some research groups have achieved dynamic and agile locomotion policies from DRL and have successfully implemented the learned policies on real-world robots. Hwangbo et al, have learned the control policies for trotting, self-righting, and standing-up and successfully deployed the learned policies on the Anymal quadruped robot [54]. Xue et al. have used imitation learning and DRL to learn natural animal-like dynamic motions such as pacing, trotting, hopping. They have also successfully bridged the simulation to reality gap and deployed the learned policy on Laikago robot [9].

Despite the successful implementation of the learned DRL control policies from Hwangbo et al. and Xue et al. on real-world quadrupeds, there are some limitations to their proposed approach. First of all, their policies are only able to perform a single locomotion skill for each. In order to execute different skills, a switching mechanism has to be designed separately, which can be a simple human command or a programmed state machine. Secondly, the learned policies are only tested on flat terrain, which does not demonstrate the robustness of the learned policies.

Our work: Our work goes beyond and achieves a multi-skilled locomotion policy that is able to perform multiple behaviors coherently and robustly in real-world unstructured environments. Our learned locomotion policy exhibits multiple locomotion related skills, including trotting, steering, and fall recovery. Moreover, our policy has been validated in the real world on uneven terrains with grass and pebbles. The resulting work on multi-skill locomotion will be presented in detail in Chapter 7.

Chapter 3

Learning Balancing Skills within a 2D sagittal plane for Bipeds

In this chapter, we investigate the feasibility of using DRL for locomotion-related motor control tasks. We designed a bipedal balancing task in a simplified 2D environment as a toy example for the investigation.

This chapter presents a control framework based on deep reinforcement learning that naturally acquires control policies that are capable of performing balancing behaviours such as ankle push-offs for humanoid robots, without explicit human design of controllers. Only the reward for training the neural network is specifically formulated based on the physical principles and quantities, and hence explainable. The successful emergence of human-comparable behaviours through the deep reinforcement learning demonstrates the feasibility of using an AI-based approach for humanoid motion control in a unified framework. Moreover, the balance strategies learned by reinforcement learning provides a larger range of disturbance rejection than that of the zero moment point based methods, suggesting a research direction of using learning-based controls to explore the optimal performance.

3.1 Human-Comparable Balancing Behaviours

Humans efficiently make use of under-actuated motions, such as toe tilting and heel rolling, for keeping balance while standing and walking. Biomechanical study of human walking has discovered the advantage of rolling around the heel and toe during walking phase [55]. From a biomechanical point of view, foot tilting creates better foot-ground clearance allowing the maximum ankle torques to be exploited [56], [24], [57].

Ankle push-off in humanoids without toes creates a control problem as an underactuated degree of freedom (DOF) is introduced. Once foot tilting occurs, the edge of the foot namely the heel or toe, becomes the only contact point between the foot and the ground which the robot pivots around. The physical feasible range of centre of pressure (COP) reduces to a

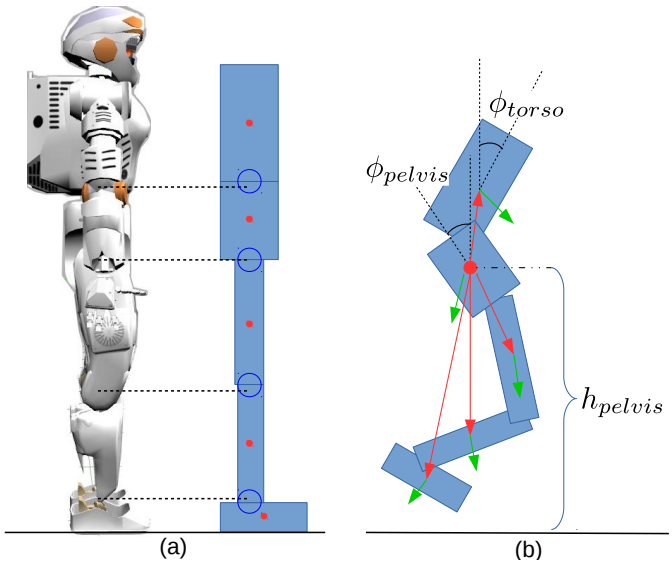


Figure 3.1: Depiction of the humanoid character. (a) Side view of 2D humanoid and the Valkyrie robot. (b) State features.

singular boundary line on the edge of the foot. This new pivot is an underactuated DOF as zero torque can be applied on the pivoting axis, thus can not be controlled by the controller.

Many humanoid robots are designed to closely resemble the human morphology to perform human-comparable behaviours. However, their control mainly produces flat-footed locomotion, which is unnatural and inefficient. The reason does not lie in the physical capabilities, but rather the limitation in the control paradigm. Most zero moment point (ZMP) based balance and walking controls assume the foot is placed flat on the ground creating a large size of support polygon as a fixed base. Most ZMP based methods will fail during the underactuation, as they require the restriction of the ZMP or COP to be within a support polygon.

Controllers that permit the COP to lie on the narrow edge of the foot have been developed to generate underactuated foot tilting behaviours, demonstrating the possibility and existence of control that is capable of actively dealing with underactuated phases during balance recovery [24].

Recently, an increasing number of research work have used machine learning, such as Deep Reinforcement Learning (DRL), to solve control tasks. Engineering based approaches require a lot of human knowledge in designing the controllers and additional effort in tuning, which is a disadvantage. Machine learning approaches, e.g. DRL, require less manual tuning. Though RL also requires a certain amount of human knowledge, rather, the main effort is in the construction of the RL agent and reward, instead of structuring explicit controllers. Once the proper agent and reward are constructed, the agent will be capable of learning the optimal policy by itself. Recent work on DRL have demonstrated that the capability of learning very complex and dynamic motor tasks. Therefore, we are motivated to explore the potential of RL

in learning a control policy to deal with both flat foot and foot tilting during humanoid balance control.

This work presents a hierarchical framework based on DRL that exploits the under-actuated behaviour for humanoid balancing control. We contribute to a reward design in an explainable manner by analysing the principles of balancing. Since DRL paradigm allows very distinct and complex behaviours to emerge from simple rewards [31], following the prior work [24], we demonstrate a successful study of exploring the DRL to acquire a policy for producing human-comparable behaviours during push recovery, without any prior knowledge of the control policies.

This chapter is organized as follows. The limitations and inevitable uncertainties of current model-based methods and the progress in DRL are discussed in Section 4.2. The proposed methodology is elaborated in Section 5.3. Results obtained in simulation are presented in Section 5.4, followed by final remarks and future work in Section 5.5.

3.2 Related Work and Motivation

Recent breakthroughs in RL and deep learning have given a rise to DRL, which is a combination of RL and deep neural networks. The DRL has enhanced the capability of agents to perform more complex and dynamic tasks. Moreover, there are well known DRL algorithms suitable for continuous state and action spaces [13], [16], [18].

There are a few successful studies on using DRL for humanoid motion control. Peng et al. successfully applied Continuous Actor Critic Learning Automaton (CACLA) [3] to train a bipedal character to learn traversal skills for terrain with gaps and walls [5]. Later, they developed a hierarchical DRL framework that has the low-level controller (LLC) to specialize on balance and limb control, while the high-level controller (HLC) focuses on navigation and trajectory planning. Using their framework, the bipedal character successfully learned to perform tasks such as soccer ball guiding, path following and obstacle avoidance [6]. Kumar et al. used DRL to learn a safe falling strategy for humanoids to minimize the damage. Their algorithm is based on CACLA and the Mixture of Actor-Critic Experts (MACE) architecture [7] in which each joint is assigned with an independent actor-critic pair. The actor with the highest corresponding critic value will be activated to generate the action. This architecture combines both continuous and discrete controls.

Most controllers developed by classical control methods for locomotion can tackle balancing problems to some degrees [58], [59], [60], but they do not intentionally deal with foot tilting as the restriction of the COP to a single point causes the system to be underactuated, which creates immense difficulties for the design of controllers. Instead, this problem is bypassed by restricting at least one foot to remain flat on the ground while the other foot simply imitates the heel-to-toe motion [61].

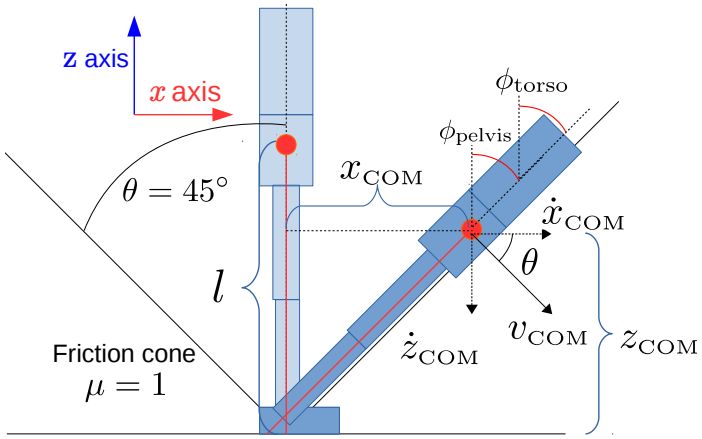


Figure 3.2: Physical quantities for reward design.

Some work has explored active foot tilting for balance recovery. The work in [24] analysed thoroughly the dynamics of foot tilting and had successfully designed a control strategy for underactuated ankle push-off with an implementation on a real robot. The underlying mechanism of foot tilting and the concrete mathematical proof in [24] suggest that foot tilting balance strategy is more robust against force perturbations than a flat-footed balance strategy.

Since the physical viability of stable underactuated behaviours has been achieved in a deterministic and analytic approach using control techniques [24], our study hereby aims to answer whether similar balancing performance and behaviour can be a natural outcome using the machine learning approach, specifically, the DRL.

3.3 Principles

3.3.1 State Representation

The bipedal character configuration used in this project is shown in Fig. 3.1(a), it is roughly modelled according to Valkyrie robot, with the goal to apply the DRL based balancing control on the Valkyrie robot in the future. The bipedal character has a the height and mass of 1.604m and 127.96kg and has 4 DOF: waist, hip, knee, ankle.

State features that contain the information of kinematics and dynamics of the humanoid were selected. Fig. 3.1(b) shows the selected state features, including pelvis height (h_{height}), joint angle and joint velocity, the angle (ϕ_{torso}, ϕ_{pelvis}) and angular velocity ($\dot{\phi}_{\text{torso}}$, $\dot{\phi}_{\text{pelvis}}$) of the pelvis and torso, ground contact information, displacement of COM of each link with respect to the pelvis (red) and the linear velocities of all body links (green). The total number of state features used as the input for the controller is 26.

3.3.2 Explainable Design of the Reward

In this section, the design of our reward function based on physical models and quantities is described. The reward has to be designed carefully in order to produce desired results. Amodei et al. has mentioned that a poorly designed reward function in reinforcement learning can cause the performance and safety issues [62].

The physical quantities needed for computing the reward is shown in Fig. 3.2. $\dot{x}_{\text{COM}}$ and $\dot{z}_{\text{COM}}$ are the actual COM velocities. x_{COM} and z_{COM} is the position of the COM on the sagittal plane. ϕ_{torso} and ϕ_{pelvis} are orientation of the upper body. l is the length from the centre of the foot to the pelvis.

Balancing can be decomposed into six objectives: keeping the torso and pelvis orientation upright ($r_{\phi_{\text{torso}}}$, $r_{\phi_{\text{pelvis}}}$), keeping the horizontal position of the COM close to the centre of the foot ($r_{x_{\text{COM}}}$), keeping the COM vertical position at a certain height ($r_{z_{\text{COM}}}$) and minimizing the horizontal and vertical velocity of the COM ($r_{\dot{x}_{\text{COM}}}$, $r_{\dot{z}_{\text{COM}}}$). The total reward is the linear combination of each objective as:

$$\begin{aligned} r =& w_{\phi_{\text{torso}}} r_{\phi_{\text{torso}}} + w_{\phi_{\text{pelvis}}} r_{\phi_{\text{pelvis}}} + w_{x_{\text{COM}}} r_{x_{\text{COM}}} \\ &+ w_{z_{\text{COM}}} r_{z_{\text{COM}}} + w_{\dot{x}_{\text{COM}}} r_{\dot{x}_{\text{COM}}} + w_{\dot{z}_{\text{COM}}} r_{\dot{z}_{\text{COM}}}, \end{aligned} \tag{3.1}$$

where the weights $w_{[\cdot]}$ of each objective in the reward are set to 1 by default, except $w_{z_{\text{COM}}}$ which is set to 5 for counteracting gravity.

The individual reward terms $r_{[\cdot]}$ are defined such that the individual parameters are attracted to their target values, while degrading exponentially as deviation gets larger:

$$\begin{aligned} r_{\phi_{\text{torso}}} &= \exp(\alpha\ (\phi_{\text{torso}}/e_{\phi_{\text{torso}}})^2) \\ r_{\phi_{\text{pelvis}}} &= \exp(\alpha\ (\phi_{\text{pelvis}}/e_{\phi_{\text{pelvis}}})^2) \\ r_{x_{\text{COM}}} &= \exp(\alpha\ ((x_{\text{COM}}^{\text{target}} - x_{\text{COM}})/e_{x_{\text{COM}}})^2) \\ r_{z_{\text{COM}}} &= \exp(\alpha\ ((z_{\text{COM}}^{\text{target}} - z_{\text{COM}})/e_{z_{\text{COM}}})^2) \\ r_{\dot{x}_{\text{COM}}} &= \exp(\alpha\ ((\dot{x}_{\text{COM}}^{\text{target}} - \dot{x}_{\text{COM}})/e_{\dot{x}_{\text{COM}}})^2) \\ r_{\dot{z}_{\text{COM}}} &= \exp(\alpha\ ((\dot{z}_{\text{COM}}^{\text{target}} - \dot{z}_{\text{COM}})/e_{\dot{z}_{\text{COM}}})^2). \end{aligned} \tag{3.2}$$

The target for the orientation of the torso and pelvis is 0 rad. $x_{\text{COM}}^{\text{target}}$ and $z_{\text{COM}}^{\text{target}}$ are the target horizontal and vertical COM position. $\dot{x}_{\text{COM}}^{\text{target}}$ and $\dot{z}_{\text{COM}}^{\text{target}}$ are the target horizontal and vertical COM velocity.

Normalization has to be performed because the range of the values of the physical properties are different. For normalization, we divide the error of each physical quantity with their respective maximum range $e_{[\cdot]}$. The individual rewards presented in (3.2) are expected to be 1×10^{-5} when the physical error reaches the maximum. Let $r_{[\cdot]} = \exp(\alpha \cdot 1) = 1 \times 10^{-5}$, we obtain the coefficient $\alpha = -11.513$.

We now describe how to compute the maximum error range $e_{[\cdot]}$ for each term. Regarding the maximum error range for the torso angle $e_{\phi_{\text{torso}}}$ and pelvis angle $e_{\phi_{\text{pelvis}}}$ with respect to

the gravity line, that occurs at $\phi_{\text{torso}} = \phi_{\text{pelvis}} = \pi/2$, when the body is fully horizontal. Thus, $e_{\phi_{\text{torso}}} = e_{\phi_{\text{pelvis}}} = \pi/2$ rad.

The maximum error range for the horizontal and vertical COM positions $e_{x_{\text{COM}}}$, $e_{z_{\text{COM}}}$ can be determined by physical constraints using the linear inverted pendulum model. Fig. 3.2 shows a humanoid leaning on the boundary of the friction cone, which is an extreme situation, because any configuration that falls outside the friction cone is destined to fail. Assuming a coefficient of friction $\mu = 1$, the maximum angle of the friction cone θ_{max} is $\pi/4$ rad. As shown in Fig. 3.2, the maximum error range for the horizontal and vertical COM positions can be computed by

$$\begin{aligned} e_{x_{\text{COM}}} &= -x_{\text{COM}} & e_{z_{\text{COM}}} &= l - z_{\text{COM}} \\ &= \sin(\theta_{\text{max}})l, & &= (1-\cos(\theta_{\text{max}}))l. \end{aligned} \tag{3.3}$$

From (3.3), we get $e_x = 0.768$ m and $e_z = 0.318$ m.

Regarding the maximum error range for the horizontal and vertical COM velocities, $e_{\dot{x}_{\text{COM}}}$, $e_{\dot{z}_{\text{COM}}}$, we can compute them using the extreme orientation angle θ_{max} and the V_{max} based on the capture point in (2.2) presented in Chapter 2. As mentioned above, we consider $\theta_{\text{max}} = \pi/4$ rad to be the extreme condition. The height of the COM is $z_0 = l\cos(\theta_{\text{max}})$ and the horizontal displacement of the COM to the centre of foot is $\Delta x_{\text{COM}} = l\sin(\theta_{\text{max}})$. The target horizontal velocity $\dot{x}_{\text{COM}}^{\text{target}}$ is set by referring to V_{max}. From (2.1) and (2.2), we can derive the target horizontal velocity as $\dot{x}_{\text{COM}}^{\text{target}} = -\Delta x_{\text{COM}}\sqrt{\frac{g}{z_c}}$ that converges to zero equilibrium with the sign opposite to Δx_{COM}, because $\dot{x}_{\text{COM}}^{\text{target}}$ always points back to the equilibrium. The target vertical velocity $\dot{z}_{\text{COM}}^{\text{target}}$ is set to 0 as we wish to minimize the vertical movement of the COM. As a result, $e_{\dot{x}_{\text{COM}}} = \dot{x}_{\text{COM}}^{\text{target}} - \dot{x}_{\text{COM}}$, $e_{\dot{z}_{\text{COM}}} = \dot{z}_{\text{COM}}^{\text{target}} - \dot{z}_{\text{COM}}$ can be computed:

$$\begin{aligned} e_{\dot{x}_{\text{COM}}} = &-\sin(\theta_{\text{max}})l\sqrt{\frac{g}{\cos(\theta_{\text{max}})l}} \\ &-\cos(\theta_{\text{max}})\sqrt{2g(1-\cos(\theta_{\text{max}}))l}, \\ e_{\dot{z}_{\text{COM}}} = &-\sin(\theta_{\text{max}})\sqrt{2g(1-\cos(\theta_{\text{max}}))l} \end{aligned} \tag{3.4}$$

The maximum error for horizontal and vertical COM velocity is thus 4.510 m/s and 1.766 m/s.

Substitute the calculated maximum range of the physical properties into (3.2), the individual reward components are obtained. The total reward can be computed by adding each weighted reward into (3.1).

3.3.3 Deep Deterministic Policy Gradient

The algorithm we chose to use is the Deep Deterministic Policy Gradient (DDPG) algorithm [18], which is a model-free, off-policy DRL algorithm based on Deterministic Policy Gradient [63] and Deep Q Networks [64].

DDPG is a type of actor critic RL algorithm, it uses two separate networks to parameterize the actor function and the critic function, respectively. The actor network $\mu(s|\theta^{\mu})$ maps the

states to a deterministic action, and the critic network $Q(s,a|\theta^Q)$ maps the state action pair to a Q-value. The learning curve of the training process can be seen in Fig. 3.4

The critic network is trained by minimizing the loss function:

$$L_Q(\theta^Q) = \mathbb{E}_{s_t,a_t,r_t,s_{t+1}\sim\mathscr{R}}\left[(Q(s_t,a_t)-y_t)^2\right], \tag{3.5}$$

Where $y_t = r_t + \gamma Q'(s_{t+1},a)|_{a=\mu'(s_{t+1})}$.

The actor network is trained by applying the deterministic policy gradient:

$$\nabla_{\theta^\mu} J = \mathbb{E}_{s_t\sim\mathscr{R}}\left[\nabla_a Q(s_t,a|\theta^Q)|_{a=\mu(s_t)}\nabla_{\theta^\mu}(s_t|\theta^\mu)\right]. \tag{3.6}$$

Algorithm 1 Deep Deterministic Policy Gradient

Initialize critic $Q(s,a|\theta^Q)$ and actor network $\mu(s,a|\theta^\mu)$
Initialize target networks Q' and μ': $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathscr{R} \leftarrow \emptyset$
for episode=1,M **do**
  Initialize random process $\mathscr{N}$ for action exploration
  Receive initial state observation s_1
  for t=1,T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathscr{N}_t$
    Execute a_t and observe reward r_t and new state s_{t+1}
    Store transition (s_t,a_t,r_t,s_{t+1}) in $\mathscr{R}$
    Sample N transitions (s_i,a_i,r_i,s_{i+1}) from $\mathscr{R}$
    for i=1,N **do**
      if state s_{i+1} is terminal state **then**
        Set $y_i = r_i$
      else
        Set $y_i = r_i + \gamma Q'(s_{i+1},\mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      end if
    end for
    Update critic by minimizing loss:
    $L = \frac{1}{N}\sum_i (y_i - Q(s_i,a_i|\theta^Q))^2$
    Update actor using sampled policy gradient:
    $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s_i,a_i|\theta^Q)\nabla_{\theta^\mu}\mu(s_i|\theta^\mu)$
    Update target networks:
    $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}, \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
  end for
end for

3.3.4 Exploration through Noise

One of the major challenge in RL is to ensure the exploration provides samples diverse enough for the RL agent. In our DRL control framework, the 2D biped is initialized in the same upright standing posture for every episode, and we rely on using the Ornstein-Uhlenbeck process (OU

noise) to generate temporally correlated values, and applied the value as perturbation to the action output of the policy to induce exploration [18].

3.3.5 Bounding Network Output

The output of the network is the desired joint angles. Joints have mechanical limits which restrict the range of motion, therefore we have to bound the action space within the angle limit. Using a squashing sigmoid activation function such as $\tanh$ in the output unit is a common way to bound network outputs. However, using $\tanh$ has its disadvantages: it is easily saturated at the upper and lower bound of the range, and requires many updates to decrease, increasing the training time and hindering the performance.

Hence, we use an approach called inverting gradients to bound the output action parameters [65],

$$\nabla_p = \nabla_p \cdot \begin{cases} (p_{\text{max}} - p)/(p_{\text{max}} - p_{\text{min}}) & \text{if } \nabla_p \geq 0 \\ (p - p_{\text{min}})/(p_{\text{max}} - p_{\text{min}}) & \text{otherwise} \end{cases}, \tag{3.7}$$

where ∇_p indicates the critic gradient with reference to action parameter, p_{max}, p_{min}, p indicate the minimum, maximum and current activation of the action parameter, respectively. From (3.7), we can see that with inverting gradients approach, the gradients are reduced as the output parameter approaches the boundary of the desired value range, and are inverted if the parameter exceeds the boundary, hereby restraining the value of the output.

Since joint angles were chosen as the action parameters of the actor network within this work, the minimum p_{min} and maximum p_{max} activation of the action parameter correspond to the lower and upper boundaries of the physical joint. The inverting gradients acts as a soft constraint for the output action and does not guarantee that the generated action stays within the set range. We therefore further applied a hard clipping in the environment to ensure that the output action does not exceed the physical limits of the joints. The clipped joint angle is then translated to joint torque by the PD controller (2.3). The generated joint torque will also be clipped to bound it within the torque limits.

3.4 Hierarchical Structure of High-Level Learning and Low-Level Control

The overall system is designed to have a hierarchical architecture that can be easily applied to the real robot system. The idea of constructing a hierarchical control architecture is widely adopted by many studies [6], [66]. In such control systems, the Lower-Level controller (LLC) and High-Level controller (HLC) work at different frequencies, where the HLC usually works at a lower frequency. Details of the hierarchical structure is described in Chapter 2. The overall structure of the control system is shown in Fig 2.3.

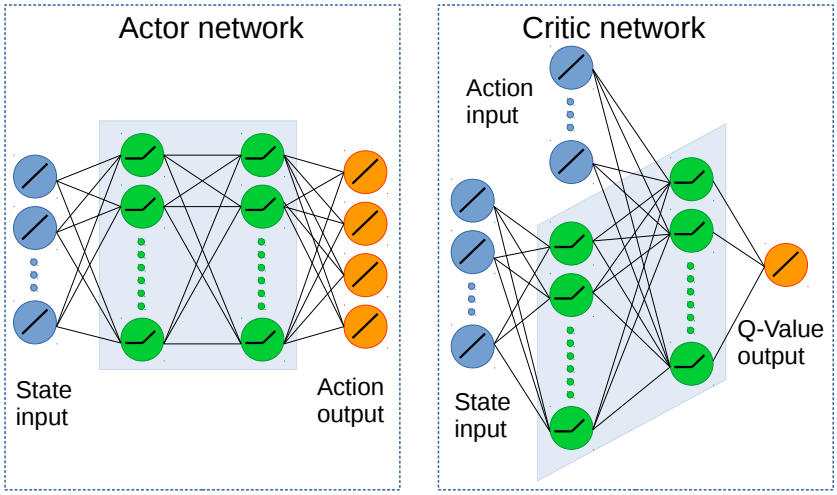


Figure 3.3: Overview of neural network structure.

3.4.1 High-Level Controller

DDPG is used to learn the high level control policy for producing the desired motion synergies, i.e desired joint angles, given the feedback. As the network structure shown in Fig. 3.3, both the critic and actor network have 2 hidden layers, and each hidden layer contains 100 nodes followed by a rectified linear unit (ReLU) activation function. In addition to the 26 state features, the critic network also takes the 4 action parameters as inputs, the action value is directly forwarded to the second hidden layer. The outputs of the actor network are the 4 references of joint angles. The network inputs of both the actor and critic network consists of 24 continuous state features, which are filtered through lowpass butterworth filters with a cutoff frequency of 10Hz, and 2 discrete state features that remain untouched.

3.4.2 Low-Level Controller

We used PD controller as the LLC, the input for the PD controller is desired joint angles produced by the HLC, and the output is the joint torque:

$$\tau = K_p(\theta_{\text{target}} - \theta_{\text{measured}}) - K_d\dot{\theta}_{\text{measured}}. \tag{3.8}$$

The feedback for the PD controller is filtered through a lowpass butterworth filter with a cutoff frequency of 50Hz. The parameters of the PD controller are tuned empirically and are shown in Table 3.1.

3.5 Results

We can calculate the maximum rejectable impulse when the humanoid is in its stable balancing configuration according to capture point theory. In the stable configuration of the policy

Table 3.1: PD gains

	Joints			
PD parameters	Waist	Hip	Knee	Ankle
K_p (Nm/rad)	720	1080	2580	3160
K_d (Nms/rad)	60	70	150	300

representation of the trained network, the horizontal distance from the COM to the front tip and back tip of the feet is respectively 0.189m and 0.111m. The height of the COM is 1.084m. According to (2.1), the maximum forward rejectable impulse is 72.8 N·s, maximum backward rejectable impulse is 42.6 N·s. A push force with duration of 0.1s is applied on the pelvis, which coincides with the COM, for simulating the impulse disturbance, therefore magnitude of the force are 728N and 426N respectively for the forward and backward pushes. Previous work has proven that foot tilting balancing strategy is capable of working under boundary rejectable impulse conditions [24].

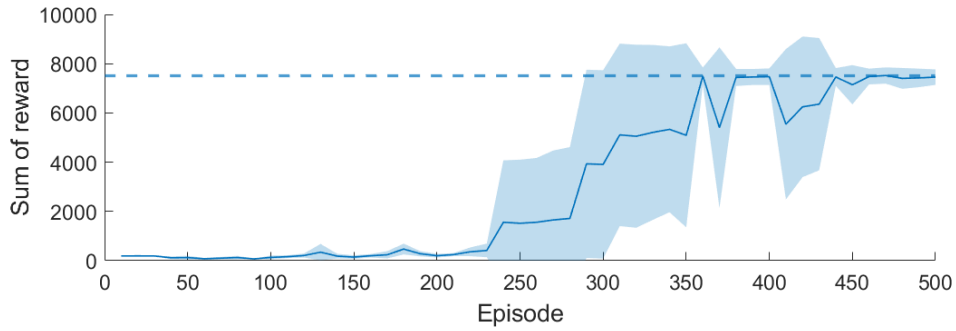


Figure 3.4: The learning curve is obtained by averaging over 6 trials, each with a different random seed during training. All 6 trials are able to obtain a successful balancing policy.

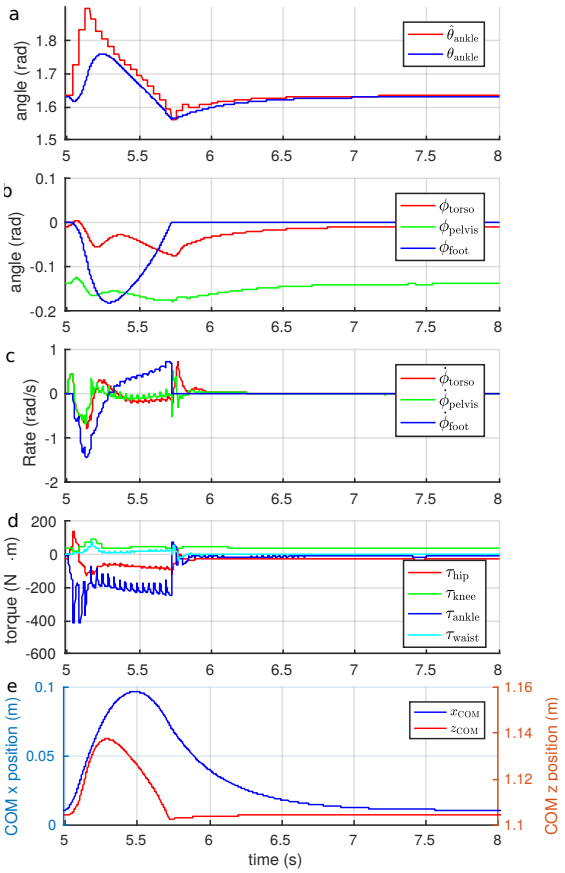


Figure 3.5: Simulation data of forward push recovery (72.8 N·s). (a) Reference/measured ankle joint angle; (b) Orientation of torso/pelvis/foot pitch; (c) Angular rate of torso/pelvis/foot pitch; (d) Ankle joint torque; (e) COM x and z position.

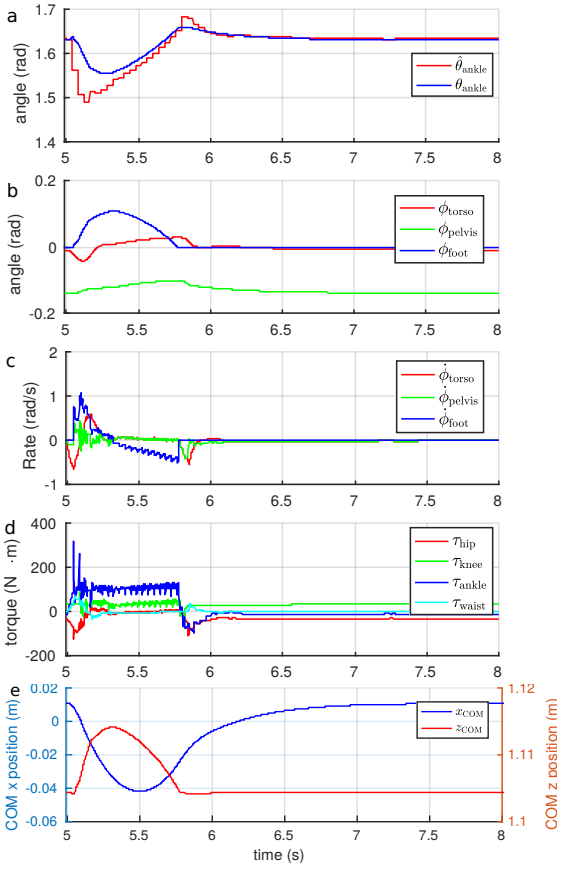


Figure 3.7: Simulation data of backward push recovery (-42.6 N·s). (a) Reference/measured ankle joint angle; (b) Orientation of torso/pelvis/foot pitch; (c) Angular rate of torso/pelvis/foot pitch; (d) Ankle joint torque; (e) COM x and z position.

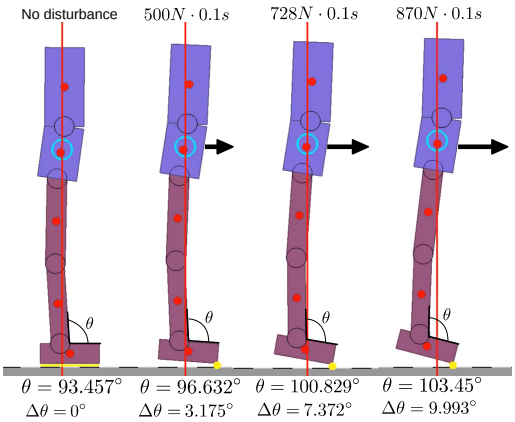


Figure 3.6: Responses generated by the policy upon forward pushes.

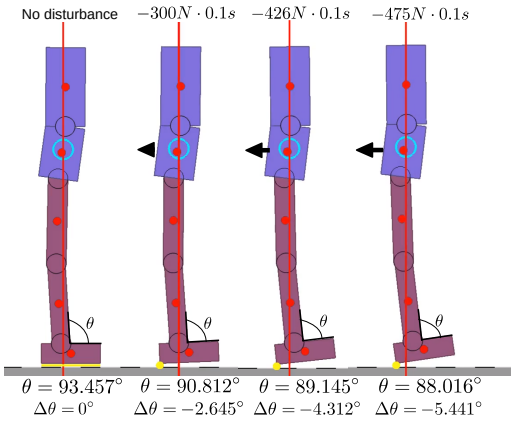


Figure 3.8: Responses generated by the policy upon backward pushes.

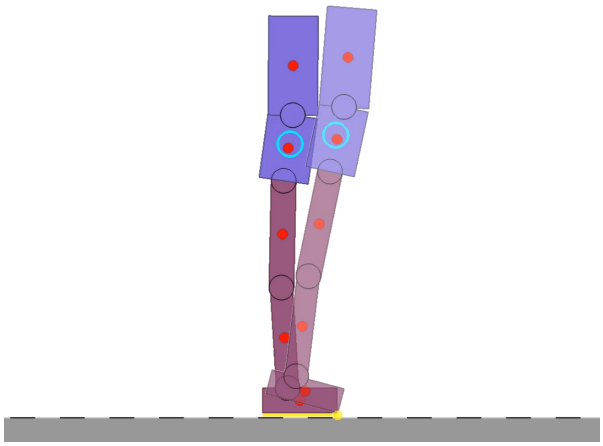


Figure 3.9: Overlay figure of the biped model highlighting the ankle push-off and knee lock behaviour.

Figure 3.5 and 3.7 respectively presents the data from forward and backward push recoveries. Figure 3.6 and 3.8 show the snapshots of maximum ankle joint angles from successful balance recoveries of forward and backward pushes under various pushes. The snapshots show that under different amount of disturbances, the tilting angles of the foot and the angle of ankle joint are different. From the simulation results, it is observed that emerged behaviours are comparable to that of humans after sufficient amount of training without any prior knowledge explicitly given by designers:

- Knee lock behaviour naturally emerges;
- Heel/toe tipping behaviours naturally emerge;
- Able to work beyond the maximum rejectable impulse calculated using capture point (due to vertical motion);
- Able to actively push-off ankle joint and create foot tilting stably in response to different disturbance;
- Exploitation of maximum achievable ankle torque.

The amount of impulse the control system is capable of withstanding is slightly larger than the rejectable impulse calculated from the capture point theory, since the capture point uses a linear model assuming constant COM height. The network learns a balancing policy capable of withstanding impulse up to 87N·s and -47.5N·s, which is respectively 119.5% and 111.5% the amount of the forward and backward maximum rejectable impulse calculated using capture point (72.8N·s and -42.6N·s). This is because while the humanoid is pivoting around its toe, the horizontal velocity is partially redirected upward, increasing the height of the COM, therefore converting part of the kinetic energy into potential energy, slowing down the overall COM

velocity. Learning-based control system is less restricted than traditional methods using ZMP in this aspect, because any stable action than improves the balance recovery will be reinforced such as ankle push-off, knee lock or any possible upper body movement.

Some human-comparable features naturally emerge after sufficient amount of training without any prior knowledge given explicitly by humans. From Fig. 3.5(b) and 3.7(b), we can see the policy actively adjusts the ankle angle to produce ankle push off behaviour. It is also shown that the humanoid has also learned to actuate its knee in a knee-lock configuration that minimizes knee torque and provides a lot of stability by simply exploiting the biomechanical constraint of the knee joint, very similar to what humans do.

The balance strategy learned by reinforcement learning has shown to have the ability to actively adjust the ankle joint angle and the tilting angle of the foot in response to the amount of disturbance applied. The active change in magnitude of ankle rotation $\Delta\theta$ relative to home position increases as the magnitude of force increases as seen in Fig. 3.6 and 3.8.

Figure 3.5(f) and 3.7(f) show the torque responses of all sagittal joints, where ankle joint in particular fully exploits the maximum achievable ankle torque for balance recovery. The control system responses to the disturbance by quickly generating ankle torque as large as possible, firstly larger than the gravitational torque for a short period to accelerate foot for tilting around the toe/heel, and then sustaining the maximum achievable torque for staying at the toe/heel with a total underactuation time about 0.8s. From the thickness, length and tilting angle of the foot, plus the mass of the body, it can be calculated that the magnitude of the maximum achievable torque while tilting around the toe and heel is respectively 216.14N·m and 106.86N·m. Simulation results from Fig. 3.5(f) and 3.7(f) shows that the magnitude of ankle torque applied during underactuation is around 210.41N·m and 110.24N·m for forward and backward push, which is close to the theoretical maximum achievable torque. The frontal section of the foot is longer than the rear section, thus the maximum achievable torque during forward push is larger than that of the backward push.

3.6 Conclusion

Previous studies have already demonstrated that human-comparable balancing behaviours such as foot tilting behaviours can be achieved using deterministic and analytical engineering approaches. The study presented in this chapter concerns about whether it is possible to produce similar or better humanoid balance strategies that involves stable underactuated ankle push-off behaviour comparable to humans using deep reinforcement learning approach.

The preliminary results demonstrated the feasibility and realizability of using deep reinforcement learning to learn a human-like balancing behaviour with limited amount of prior structure being imposed on the control policy. We transfer knowledge from control engineering based methods and applied them into the design of rewards for RL. The importance of a physic based

reward design shall be acknowledged. Otherwise it is difficult to balance the influence among different physical quantities and the balance behaviour is difficult to be guaranteed by RL. The ankle push-off behaviour learned by RL is able to work robustly under circumstances where impulses are as much as the theoretical maximum that can be rejected. Moreover, DRL has learned an adaptive way of actively changing ankle push-off angle in response to the applied disturbance.

The scope of the work presented within this chapter only covers standing balance within the sagittal plane in a 2D simulation as a proof-of-concept using learning approach. The next step will be to perform simulation in a 3D environment with more realistic physical settings. The details of the implementation of 3D balancing framework will be presented in the next chapter.

Chapter 4

Learning Balancing Skills for Bipeds in 3D simulation

In Chapter 3, we described how human-like balancing behaviours, such as active foot tilting and knee locking, emerge naturally from deep reinforcement learning in a 2D simulation environment. The balancing policy presented in Chapter 3 is limited to the 2D sagittal plane. In this chapter, we describe how to extend the work done on 2D balancing to a 3D balancing policy that operates in both the sagittal and lateral planes.

This chapter presents a hierarchical framework for deep reinforcement learning that acquires motor skills for a variety of push recovery and balancing behaviours, i.e., ankle, hip, foot tilting, and stepping strategies. The policy is trained in a physics simulator with realistic setting of robot model and low-level impedance control that are easy to transfer the learned skills to real robots. The advantage over traditional methods is the integration of high-level planner and feedback control all in one single coherent policy network, which is generic for learning versatile balancing and recovery motions against unknown perturbations at arbitrary locations (e.g., legs, torso). Furthermore, the proposed framework allows the policy to be learned quickly by many state-of-the-art learning algorithms. By comparing our learned results to studies of preprogrammed, special-purpose controllers in the literature, self-learned skills are comparable in terms of disturbance rejection but with additional advantages of producing a wide range of adaptive, versatile and robust behaviours.

4.1 Introduction

Legged robots have great potential for being deployed in environments where wheeled robots are limited, such as obstacle obstructed terrain as well as narrow and elevated surfaces (e.g., stairs). However, in contrast to wheeled or tracked robots, humanoids are intrinsically unstable and require active control to balance due to their limited support area, high center of mass, and limited actuator capabilities. Therefore, the range of possible scenarios in which humanoids

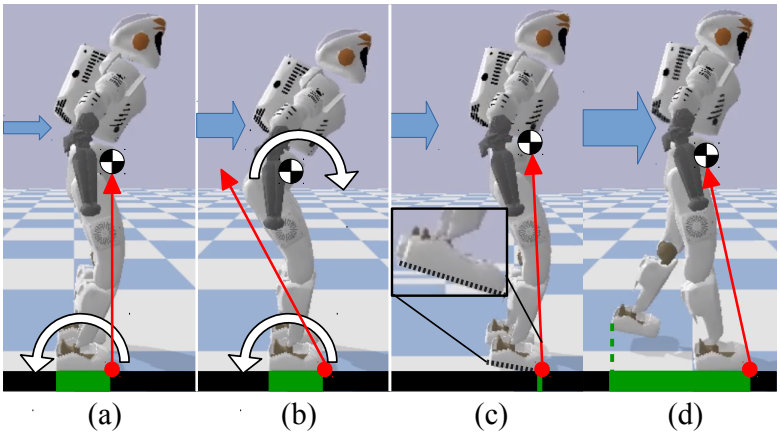


Figure 4.1: Learned push recovery behaviour: (a) ankle strategy, (b) hip strategy, (c) foot-tilting strategy, (d) stepping strategy.

can be deployed is mostly limited by the humanoids' ability to maintain balance and deal with disturbances and uncertainties. Balance can therefore be considered as one of the core skills for humanoid robots and locomotion.

Classical control methods propose a wide range of balance recovery algorithms, which however lack in the universality of their application. In order to deal with a wide range of pushes, different control strategies need to be applied and traditionally a switching between controllers for the given situation is needed. Generally, different sets of parameters are used for the four main push recovery strategies: ankle, hip, foot-tilting, and stepping (cf. Fig. 4.1).

Methods from machine learning on the other hand provide a promising alternative as they can incorporate multiple push recovery skills without the need for hand-tuned gains. Their use is motivated by three main factors:

First, supplementing existing control strategies with learning methods allows dealing with scenarios that are hard to engineer in a traditional sense such as sudden, high impact forces and discrete, sudden switches of contact.

Second, in contrast to planning and control algorithms that demand high computational power to run at or close to real-time, e.g. Model-Predictive Control, the computation for machine learning approaches can be outsourced offline [67]. I.e., the computation for Deep Reinforcement Learning (DRL) can be off-loaded into the neural network training phase. By doing so, faster online performance for high dimensional control systems such as humanoids can be achieved.

Last, in recent years, DRL has been shown to be capable of solving complex manipulation and locomotion tasks that involve learning a control policy in high-dimensional continuous observation and action spaces [13], [18], [15]. Instead of manually tuning the control parameters, a feasible policy is learned through interaction with the environment.

While there exist various studies using DRL to learn bipedal locomotion for humanoids [68],

[6], [45], the used robot models leverage simplified dynamic and collision models and environments in order for faster than real-time simulations at the cost of less realistic simulations. The motivation of this work is to learn locomotion skills using a realistic robot model obtained from system identification in a realistic simulation environment in order to apply the learned skills on the real robot. We leverage recent advances in DRL to design a unified balance recovery controller which is able to generate sequences of actions that perform similar to or even exceed traditional methods with respect to their disturbance rejection ability. Our work has the following contributions:

1. Application of DRL on an accurate robot model of the Valkyrie platform with realistic settings for simulation.

2. Design of a learning framework that generates a generic policy. This policy captures a variety of sensor-motor synergies and various control strategies emerge in a unified manner without the need of multiple controllers and the related switching mechanism.

3. Proposal of a balance recovery specific reward design and training settings of disturbances. These allow the exploration of versatile motions and as a result, human-like balancing behaviours, such as foot-tilting and stepping, emerge naturally.

4. Benchmarking the learned policy against control methods. The learned policies generate balance recovery strategies which reject impulses in a similar (or even superior) magnitude as traditionally designed controllers.

This chapter is organized as follows. A brief review on conventional push recovery methods and DRL is presented in Section 4.2. Background information on some concepts of DRL and push recovery is explained in Section 5.3. The proposed methodology is elaborated in Section 5.4. The obtained results are demonstrated and discussed in Section 5.5. Finally, a conclusion is drawn in Section 5.6.

4.2 Related Work

4.2.1 Conventional Push Recovery Methods

Over the past two decades, remarkable progress in the field of push recovery for humanoid robots has been made. Without the use of arms, humanoids can leverage four lower body balancing strategies: ankle, hip, stepping, and foot-tilting. The first three strategies, controlling ankle torque, angular momentum around the Center of Mass (COM), and the timing and position of steps, are analyzed with respect to their ability to reject disturbances in [69]. A control framework for the foot-tilting strategy has been proposed in [24] demonstrating a humanoid's ability to use foot-tilting for push recovery. Traditionally control schemes can be

divided into predictive schemes which calculate reference motions, and reactive schemes which respond to sudden disturbances.

A Model Predictive Control (MPC) scheme that constrains the Center of Pressure to be within the Support Polygon has been proposed in [70]. Strategies involving modulating the Angular Momentum to reactively deal with disturbances have been formally analyzed in [23], [71]. Due to the limited size of the contact area, i.e. the foot size, stepping strategies have been proposed [72], [73]. This idea was formalized as the Capture Point (CP), the point on which one needs to step in order to come to a complete halt [23]. Enlarging the support area by stepping has been extended to multi-contact push recovery scenarios in [74], [75]. Methods for balancing on inclined slopes has also been proposed [76]. Lastly, strategies modulating the height of the COM in order to compensate for disturbances have been proposed in [77]. This COM height modulation can be achieved by either lengthening the leg or in form of foot tilting [24].

4.2.2 Deep Reinforcement Learning of Locomotion

There exists various successful studies using model-free DRL to solve bipedal locomotion tasks in 3D simulation environments. Schulman et al. proposed a DRL algorithm, Proximal Policy Optimization (PPO), which was applied to successfully learn a locomotion policy that is capable of heading towards a target location in the Roboschool humanoid simulation environment [15]. PPO, together with Deep Deterministic Policy Gradient (DDPG) [18] and Trust Region Policy Optimization (TRPO) [13], are the most commonly used state-of-the-art DRL algorithms for continuous observation-action space control. Further extensions include a parallel computing version of PPO, Distributed Proximal Policy Optimization (DPPO) [31], which was applied on a humanoid and successfully learned dynamic and diverse parkour movements for the humanoid character.

Various frameworks have been proposed to allow the DRL agent to learn a policy that generates human-like locomotion behaviour for bipedal locomotion tasks. Merel et al. proposed a framework that uses generative adversarial imitation learning [34] to enable the network to learn a policy that produces human-like locomotion gait using limited demonstrations from motion capture data [38]. Peng et al. proposed a framework that incorporated imitation learning by reshaping the reward through the introduction of an imitation term that provides higher reward when the motion is closer to the reference motion capture data [8].

4.3 Background

4.3.1 Software Setup

The simulation environment of the Valkyrie robot is built using PyBullet [1] (Fig. 4.2). The robot model used in the simulation is the NASA Valkyrie robot [78] with realistic inertia, center

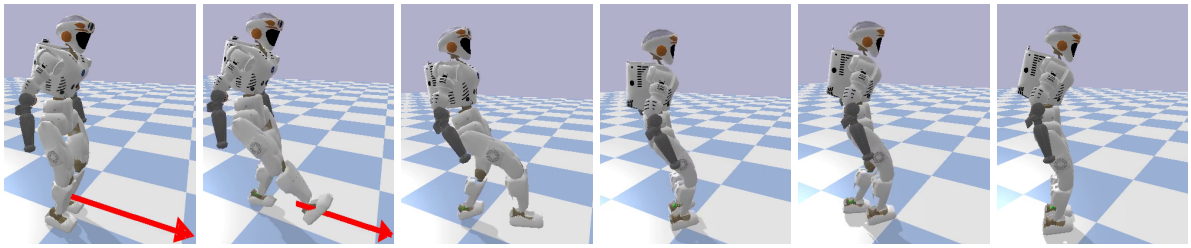


Figure 4.2: Snapshots of Valkyrie recovering from an impulse at the shin of $108Ns$, which is a test scenario not encountered during training. The learned policy automatically generates a stepping behaviour (cf. `https://youtu.be/43ce2cLV0ZI`).

of mass, and joint actuation limits. Self-collisions are enabled in the simulation. The DRL algorithm is built using Tensorflow [79].

4.3.2 Deep Reinforcement Learning

For learning a suitable policy, Deep Reinforcement Learning, particularly model-free policy gradient methods, are used. Policy gradient algorithms operate by maximizing the direct sum of rewards with reference to a stochastic policy. The policy gradient algorithms used in this work are the TRPO [13], PPO [15], and DDPG [18]. Due to resulting in the best and most robust policy, the TRPO algorithm will be outlined in the following.

4.3.2.1 Trust Region Policy Optimization

In practice, policy gradient methods suffer from high variance which can lead to fluctuations in the performance of the policy between iterations. This problem of instability during training is remedied by introducing a trust region to the numerical optimization which takes a step in the improving direction within a determined trust region. By constraining the amount of changes to the parameters, measured by the Kullback-Leibler (KL) divergence, TRPO guarantees a theoretical monotonic performance improvement of the reward.

For every parameter update iteration, TRPO performs several rollouts and stores the state s_t, action a_t and reward r_t into a batch $\mathscr{D}$ until enough data samples are collected, which will then start the update process. During the update process, TRPO updates the policy parameters by minimizing a surrogate loss function while constraining the KL divergence between the new and old policies π_θ, $\pi_{\theta_{old}}$ to remain within a trust region:

$$\min_\theta L_{\theta_{old}}(\theta) = -\mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}A_t\right] \tag{4.1}$$

$$\text{subject to } \mathbb{E}_t\left[KL\left[\pi_{\theta_{old}}(\cdot|s_t)\pi_\theta(\cdot|s_t)\right]\right] \leq \delta, \tag{4.2}$$

where δ is the hyperparameter that determines the trust region, $A_t = R_t - V(s_t)$ is the advantage which is calculated by subtracting the return with a baseline. A value estimation $V(s_t)$ provided by a critic is used as the baseline.

4.3.2.2 Discounted Return

The total return is used as an evaluation of performance and is determined by calculating the discounted reward,

$$R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}, \tag{4.3}$$

where T is the total number of samples in an episode and γ is the discount factor. The half-life of future rewards is used as a reference to decide the value of the discount factor γ. For balancing, a time horizon between 0.5s and 2s is short. With a frequency of 25Hz, 0.5s equates to 13 time steps. We choose the discount value in a way that the half-life of the future reward occurs at 0.5s, meaning that the accumulated discount factor equates to 0.5 at 13 time step $\gamma^{13} = 0.5$, hence $\gamma \approx 0.95$.

4.3.2.3 Generalized Advantage Estimation

With the policy gradient method and a stochastic policy, we obtain an unbiased estimate of the gradient of the expected total reward, however the estimated policy gradient has high variance. An effective variance reduction scheme for policy gradients called Generalized Advantage Estimator (GAE) was proposed in [80]. GAE interpolates between a high bias and low bias estimator through the parameter $\lambda \in [0,1]$. One can adjust the bias/variance trade-off by tuning λ. The GAE for the parameters γ, λ at time t is:

$$\begin{aligned} A_t^{GAE(\gamma,\lambda)} &:= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+1}^V \\ \delta_{t+1}^V &= r_t + V(s_{t+1}) - V(s_t). \end{aligned} \tag{4.4}$$

4.3.3 Capture Point

The Capture point (CP) describes the point on the ground on which the robot should step on in order to come to a complete rest [23]. The detailed explanation of the capture point is presented in Chapter 2.

When the CP is within the support polygon, the robot does not need to perform any footstep to maintain balance. Knowing the feet dimensions and therefore the support polygon, the theoretical maximum impulse which can be rejected without foot-stepping can be approximated as follows. [23]:

$$J_{\text{reject}} = m\sqrt{\frac{g}{z_c}}\Delta_{\text{COP}}, \tag{4.5}$$

Where Δ_{COP} is the distance between the COM and the closest border of the Support Polygon in the direction of the push. For the nominal, upright-standing pose the dimensions of the Support Polygon of Valkyrie is $0.26m \times 0.38m$, the COM height is at $1.1m$, the mass of the Valkyrie robot is $137kg$. Equation (4.5) yields an approximate maximal impulse of $53Ns$ in the sagittal plane and $78Ns$ in the lateral plane for $\Delta_{\text{COP}} = [0.13m, 0.19m]$.

Table 4.1: PD gains for the joints of Valkyrie. Only the torso pitch, left and right hip pitch & roll, knee pitch, and ankle pitch & roll joints are actuated.

	Valkyrie	
	K_p (Nm/rad)	K_d (Nms/rad)
Torso pitch	3000	300
Hip roll	1500	150
Hip pitch	2000	180
Knee pitch	2000	180
Ankle pitch	2000	120
Ankle roll	1500	90

4.4 Methodology

4.4.1 Hierarchical Control Framework

We designed our control framework to have a hierarchical structure (Fig. 2.3). A hierarchical structure allows implementation of two (high and low-level) layers that are independent from each other and can be designed and calibrated separately. The high-level control works under a frequency of 25Hz while the low-level control works at 500Hz. The high-level control is responsible for generating joint angles for a desired motion and the low-level control is responsible for translating the joint angles into joint torques. Details of the hierarchical control structure can be seen in Chapter 2. The PD gains for the low-level PD controller are shown in Table 4.1.

4.4.2 Observation Space and Action Space

Input states are chosen in a way such that they can be acquired by sensors on the robot with minimal amount of computation. Immeasurable states are inferred or estimated by the Neural Network. All the sensory information provided as the observational input for the policy is heading-invariant. For balancing, the rotation along the direction of the gravity vector is irrelevant to the balancing state, therefore information about the heading is not needed as feedback, i.e. the policy will perform the same action regardless of global yaw orientation. In order to make the state observation heading invariant, we preprocessed the state by performing transformation of the observations along the gravity axis.

The state $\mathscr{S} \in \mathbb{R}^{47}$ consists of 11 joint angles and 11 joint velocities, pelvis states (translational and angular velocity, orientation), COM states (translational velocity and position w.r.t. pelvis), ground contact force, torso position w.r.t. pelvis, and foot position w.r.t pelvis. The observation states are sampled at a frequency of 500Hz and are filtered by a first-order Butterworth filter with a cut-off frequency of 10Hz.

Under consideration of computation efficiency, we minimize the size of action space. A minimum of 11 joints that includes only roll and pitch joints are sufficient for balancing. The action space $\mathscr{A} \in \mathbb{R}^{11}$ of the policy describes the motion of the joint angles. The upper body joints are locked in a nominal position, while for the lower body, only the pitch joint and roll joint are controlled. The controlled joints therefore are: torso pitch, left and right hip pitch & roll, knee pitch, and ankle pitch & roll.

4.4.3 Design of Reward Function

The design of the reward function is a crucial part in reinforcement learning as the reward governs the outcome behaviour. The reward design follows a similar design rule as in [29]. Balancing can be divided into four subtasks: regulating upper body pose, regulating COM position, regulating COM velocity, and regulating ground contact force. The individual reward is calculated using $K(x, x_{target}, \alpha_i) = \exp(\alpha_i(x_{target} - x)^2)$, with x_{target} as the desired value, x as the real value, and α_i as the normalization factor. These are then weighted by w_i (Table 4.2). Furthermore, additional penalty terms are added: ground contact regulation, loss of contact with the ground, and when other parts of the body other than the foot make contact with the ground. We also apply a penalty for the control effort used. The overall reward can be viewed as a sum of the individual reward terms:

$$\begin{aligned} r = & r_{pose} + r_{CoM\_pos} + r_{CoM\_vel} + r_{GRF} + \\ & r_{contact} + r_{power}. \end{aligned} \tag{4.6}$$

4.4.3.1 Upper Body Pose Modulation

The upper body pose is represented by the pitch and roll angle of the torso and pelvis. The desired orientation for the pitch roll angle for both pelvis and torso is 0, which is the orientation of the upper body when it is upright:

$$\begin{aligned} r_{pose} = & w_{\phi_{\text{torsoPitch}}} \tilde{r}_{\phi_{\text{torsoPitch}}} + w_{\phi_{\text{pelvisPitch}}} \tilde{r}_{\phi_{\text{pelvisPitch}}} + \\ & w_{\phi_{\text{torsoRoll}}} \tilde{r}_{\phi_{\text{torsoRoll}}} + w_{\phi_{\text{pelvisRoll}}} \tilde{r}_{\phi_{\text{pelvisRoll}}}. \end{aligned} \tag{4.7}$$

4.4.3.2 COM Position Modulation

The reward term for COM modulation is decoupled into horizontal and vertical components. For the horizontal COM position, the target position is the center of the support polygon to provide maximum disturbance compensation. For the vertical COM position, the robot should stand upright and maintain a certain height,

$$r_{CoM\_pos} = w_{xy_{\text{CoM}}} \tilde{r}_{xy_{\text{CoM}}} + w_{z_{\text{CoM}}} \tilde{r}_{z_{\text{CoM}}}. \tag{4.8}$$

Table 4.2: Detailed description of task reward terms. The terms are combined to construct the task reward. The corresponding normalization factor and weight for the reward terms are α and w.

Task reward terms			
Torso pitch	$K(\theta_n, 0, \alpha), n =$ torso pitch	$\alpha = -2.80$	$w = 0.059$
Torso roll	$K(\theta_n, 0, \alpha), n =$ torso roll	$\alpha = -2.80$	$w = 0.059$
Pelvis pitch	$K(\theta_n, 0, \alpha), n =$ pelvis pitch	$\alpha = -2.80$	$w = 0.059$
Pelvis roll	$K(\theta_n, 0, \alpha), n =$ pelvis roll	$\alpha = -2.80$	$w = 0.059$
Position in xy plane	$K(xy, \hat{xy}, \alpha)$	$\alpha = -14.10$	$w = 0.118$
Position in z axis	$K(z, \hat{z}, \alpha)$	$\alpha = -14.10$	$w = 0.235$
Velocity in xy plane	$\begin{cases} 0, \text{if no foot contact with ground} \\ K(\dot{xy}, \hat{\dot{xy}}, \alpha) \end{cases}$	$\alpha = -6.91$	$w = 0.118$
Velocity in z axis	$K(\dot{z}, 0.0, \alpha)$	$\alpha = -6.91$	$w = 0.059$
Left foot contact force	$K(F_{Left}, 673.1, \alpha)$	$\alpha = -2.45e-6$	$w = 0.059$
Right foot contact force	$K(F_{Right}, 673.1, \alpha)$	$\alpha = -2.45e-6$	$w = 0.059$
Ground contact	$\begin{cases} 0, & \text{if no foot contact with ground} \\ 1, & \text{if upper body contact with ground.} \end{cases}$		$w = 0.059$
Power penalty	$0.005 \cdot \sum_{j=0}^{11} (\tau^j)^2$		$w = 0.059$

4.4.3.3 COM Velocity Modulation

Similar to the COM position, the reward for COM velocity is decoupled into two components: velocity in the horizontal and vertical planes. The COM velocity is represented in the world frame. The desired vertical COM velocity is 0 as we want to minimize vertical movement, while the desired velocity for horizontal COM velocity is derived from capture point. The Capture Point is only valid when the robot has contact with the ground with no slipping, therefore when the robot is in the air, the reward term for horizontal COM velocity $\tilde{r}_{\dot{xy}_{\text{CoM}}}$ is deemed invalid and is set to 0:

$$r_{CoM\_vel} = \begin{cases} w_{\dot{xy}_{\text{CoM}}} \tilde{r}_{\dot{xy}_{\text{CoM}}} + w_{\dot{z}_{\text{CoM}}} \tilde{r}_{\dot{z}_{\text{CoM}}}, & \text{foot contact} \\ w_{\dot{xy}_{\text{CoM}}} \cdot 0 + w_{\dot{z}_{\text{CoM}}} \tilde{r}_{\dot{z}_{\text{CoM}}}, & \text{no foot contact.} \end{cases} \tag{4.9}$$

4.4.3.4 Contact Force Modulation

The force has to be evenly distributed between both feet for a stable robust balance. The total mass of 137kg yields a force of 671.3N for each foot:

$$r_{GRF} = w_{F_{left}} \tilde{r}_{F_{left}} + w_{F_{right}} \tilde{r}_{F_{right}}. \tag{4.10}$$

4.4.3.5 Ground Contact

When the robot is standing, only the feet are in contact with the ground, therefore a penalty is introduced whenever both feet lose contact with the ground or body parts other than the feet

make contact with the ground:

$$r_{contact} = w_{contact} \cdot \begin{cases} -2, & \text{if no foot contact with ground} \\ -10, & \text{if upper body contact with ground.} \end{cases} \tag{4.11}$$

4.4.3.6 Power Consumption

The power consumption is calculated as follows,

$$r_{power} = w_{power} \cdot 0.005 \cdot \sum_{j=0}^{11} \left\| \tau^j \cdot \dot{q}^j \right\|, \tag{4.12}$$

with τ^j is the torque applied on individual joints, and $\dot{q}^j$ is the joint velocity.

4.4.4 Network Structure

The stochastic policy $\pi_\theta(a|s)$ is represented as a conditional Gaussian policy $\pi_\theta(a|s) \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta)$. The mean of the Gaussian policy is parametrized by a neural network with parameters θ, the covariance of the Gaussian policy is independent from the neural network and is maintained by a separate set of parameters σ_θ.

The critic V_ϕ parametrizes the value function with a separate set of neural networks using parameters ϕ. Both the actor and the critic are parametrized by a fully connected feedforward neural network that consists of 3 hidden layers with 100, 50 and 25 neurons for each layer. The actor network uses $\tanh$ activation for the hidden layers while the critic uses ReLU activation for the hidden layers. The output of both network is produced by linear activation.

The actor network is trained to maximize the reward function (section 4.4.3), while the critic network is trained by minimizing the loss function $L_V(\phi)$:

$$L_V(\phi) = \mathbb{E}_t \left[(V(s_t) - y_t)^2 \right], \tag{4.13}$$

with the discounted Return y_t (eq. 4.3), value function $V(s_t)$.

4.4.5 Exploration during Training

In order to learn a policy capable of withstanding large push disturbances, sufficient exploration during the training phase needs to be provided. Therefore, in addition to the stochastic policy, random forces are applied on the pelvis during the training. From Capture Point theory, the maximum disturbance in the sagittal plane without foot-stepping is $53Ns$. The bounds of the training disturbances is chosen to be $[53 \times 0.5Ns, 53 \times 2Ns]$. The orientation of the force in the horizontal plane and the disturbance in the bound are randomized using a uniform distribution. Disturbances are applied to the robot multiple times during each trial, with 5s interval between subsequent push disturbances for push recovery. The posture during the initialization of each episode is fixed at the same upright standing posture and does not contribute to the exploration.

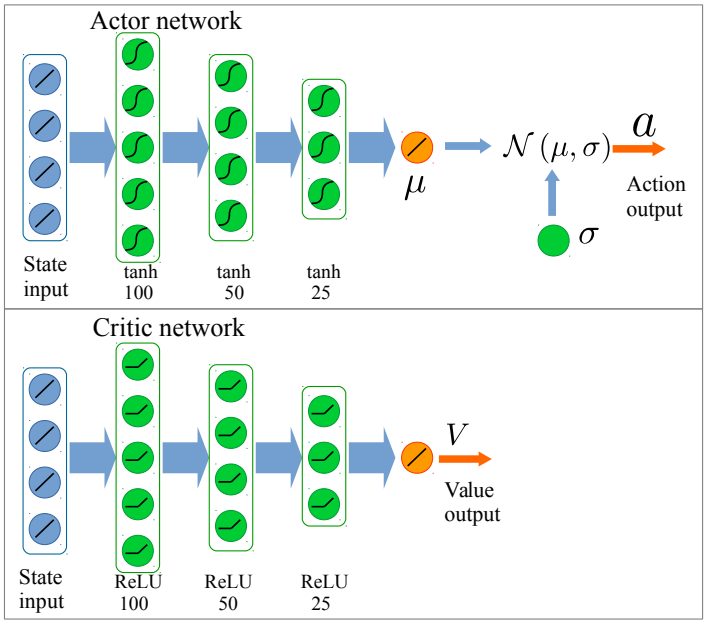


Figure 4.3: Overview of neural network structure.

4.4.6 Deep Reinforcement Learning

Due to the structure and choice of our framework, the learned policy is independent of the type of learning algorithm. We trained a policy for maintaining balance via TRPO, PPO, and DDPG, and found similar resulting behaviour (cf. Table 4.3). All four balancing strategies (Fig. 4.1) emerges regardless of the DRL algorithm used. However, from our simulations, TRPO is able to achieve higher rewards and is able to withstand higher impulses. Figure 4.4 shows the learning curves for the policies learned in Table 4.3. DDPG is trained off-policy and utilizes a replay buffer, whereas TRPO and PPO are trained on-policy batch-wise, which makes it difficult to directly compare.

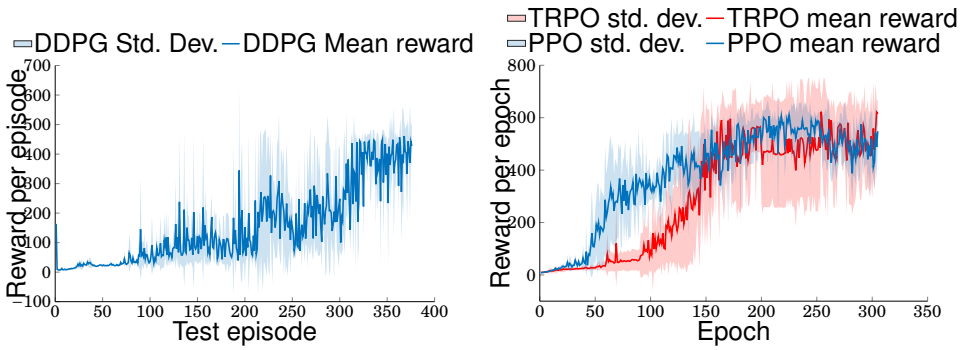
All three DRL algorithms are able to learn a feasible balancing policy. The difference in performance can be attributed to the randomness in different trials of training and hyperparameters. Training is performed entirely on a single Intel Core i7-6700K with 4.0 GHz and converges in two days.

4.5 Results

In the following, a series of test scenarios are presented to evaluate the performance of the control policy acquired by the deep reinforcement learning agent. Furthermore, we show its robustness to external disturbances, as well as noise in the observation (measurement) and action (actuation) spaces. Next, a comparison against traditional methods from other work is made. Lastly, the physical validity of the generated motions is analysed and verified. Please

Table 4.3: Maximal rejectable impulses for the various learning algorithms without taking steps.

	Maximal disturbance in Ns	
Learning algorithm	Sagittal	Lateral
TRPO	240	78
DDPG	75	160
PPO	192	36
Baseline from (4.5)	53	78



(a) DDPG (b) TRPO & PPO

Figure 4.4: Learning curves for DDPG, PPO, and TRPO. The performance are evaluated using the deterministic policy. The mean of the Gaussian policy learned by PPO/TRPO is used for evaluation. The results are averaged over 7 learning trials.

refer to the accompanying video for the results (cf. `https://youtu.be/43ce2cLV0ZI`).

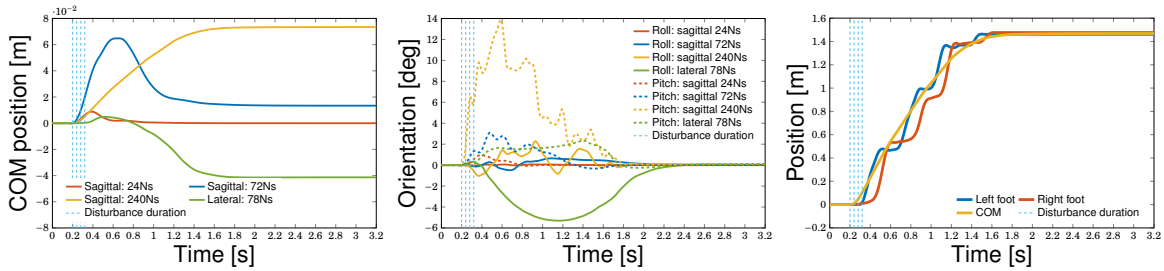
4.5.1 Horizontal Push on Pelvis

Horizontal disturbances were applied on the pelvis during the training phase, and therefore the DRL agent should be able to learn to withstand such type of disturbances. The robot exhibits different behaviour depending on the amount of disturbance applied (Table 4.4, Fig. 4.5). Different control strategies emerge and range from generating ankle torque to shift the COP (ankle strategy), generating angular momentum (hip strategy), over tilting the foot to dissipate the disturbance (foot tilt strategy), to taking a step to recover from the large push (stepping strategy). The magnitude of the lateral pushes, for which the robot is capable of withstanding, is significantly smaller than in the sagittal plane. This is due to the fact that the support leg will block the swing leg in the lateral direction, limiting the range for leg movement. Dealing with this problem involves either jumping to take a step, or crossing the legs, which, due to the kinematic constraints, is not possible for Valkyrie. These jumping manoeuvres were not learned by the policy, as high velocities in the COM resulted in lower rewards.

From the COM position (Fig. 4.5a) and the pelvis orientation (Fig. 4.5b) it can be inferred that the robot is standing still after 3s. As can be seen by the eight flat plateaus in Figure 4.5(c),

Table 4.4: Emerging behaviour for impulse disturbances of different magnitudes. A checkmark indicates that the respective strategy is applied in addition to the other marked strategies.

	Impulse disturbance in Ns			
Emerging behaviour	$24Ns$	$72Ns$	$240Ns$	$78Ns$
Push direction	sagittal	sagittal	sagittal	lateral
Ankle strategy	✓	✓	✓	✓
Hip strategy	✓	✓	✓	✓
Foot tilt strategy	×	✓	✓	✓
Stepping strategy	×	×	✓	×



(a) COM motion over time for multiple forces. The COM position of the $240Ns$ impulse is scaled by 0.05

(b) Pelvis orientation over time.

(c) Foot and COM movement for sagittal impulse of $240Ns$.

Figure 4.5: Resulting motions from impulse disturbance and balance recovery.

eight steps are taken in order to deal with a impulse disturbance of $240Ns$ at the pelvis. After the eighth step the robot stands still in the nominal pose. For other horizontal pushes, the stepping behaviour is similar.

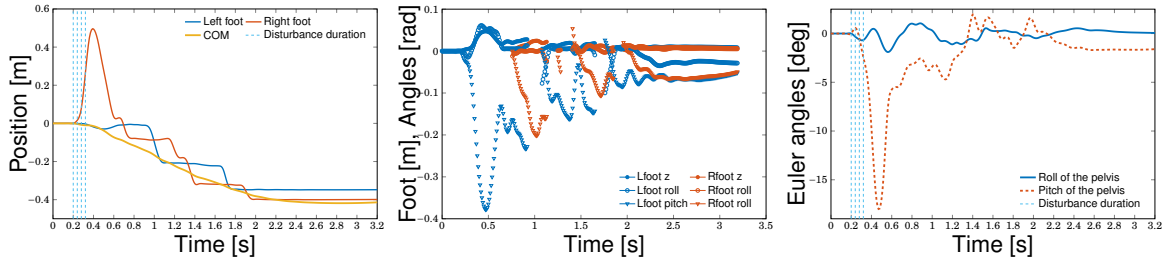
4.5.2 Force Disturbance on other Body Segments

During the training phase only horizontal disturbances were applied on the pelvis. It is well known in machine learning that the test set should vary from the training set. Therefore, we also designed test scenarios which the DRL agent has never encountered before during training to see how well the policy generalizes.

In push recovery studies, the disturbance is usually applied near the COM to avoid introducing any torque into the system, as it is more challenging to balance a robot with high amount of angular momentum. We are interested in how well the policy will perform when disturbance is applied on other parts of the body far away from the CoM. We chose three body parts for which a large torque and angular movements would result when force is applied on: the upper torso, leg thigh, and leg shank (Table 4.5). The resulting motion for being pulled at the shank can be seen in Figure 4.6. In Figure 4.6a) six steps for recovering balance are observed. The support foot height, and roll and pitch angle relative to the ground can be seen in Figure 4.6b). Finally,

Table 4.5: Maximum rejected impulse for different body parts.

Body part	Max. impulse in $[Ns]$	Lever in $[m]$	Torque in $[Nm]$	Amount of steps
Upper torso	120	0.32	320	6
Leg thigh	108	0.50	450	4
Leg shank	108	0.70	630	6



(a) COM and feet motion.

(b) **Support** foot behaviour (swing foot not depicted).

(c) Angular movement of the pelvis.

Figure 4.6: Resulting motions from an impulse disturbance at the shank. The robot takes 6 steps before standing stably.

the angular movement of the pelvis Euler angles (Fig. 4.6c) show that the robot recovers into an almost nominal pose after six steps.

4.5.3 Landing from Height

A different type of impact is applied to the robot by dropping it from a height above the ground with the objective to land stably. Landing involves high impact and sudden changes in ground contact. Various control strategies involving the Zero-Moment Point (ZMP) or CP assume steady contact with the ground and, without a switching mechanism, fail to perform when the robot is in the air. This test scenario is used to test whether the policy can handle high impact and sudden change in ground contact and land stably.

The robot is capable of landing after being dropped from a maximum height of $0.55m$ above the ground. Furthermore, it is able to handle randomly initialized pitch orientations of the pelvis within $[-5°, 5°]$ under a dropping height of 0.4m (with a leg length of $1m$, this is a displacement of $0.087m = 1m \cdot \sin(5\frac{\pi}{180}$ compared to the initial position).

4.5.4 Combined Test Case

Lastly, a combined test case involving linear and angular momentum disturbances is designed. In this test scenario, we first apply a vertical force upward to lift the robot off the ground and then apply a horizontal force when the robot is in the air. As a result, the robot has to handle linear and angular momentum and sudden high impacts at the same time. The robot is able to recover for a vertical impulse of $500Ns$, and a horizontal impulse of $70Ns$. In order to deal with

this combined disturbance, the policy used all four push recovery strategies: ankle, hip, toe, step. Apart from the physical impossibility of throwing a 137kg robot, the resulting joint torques, velocities are within the joint limits of the real Valkyrie (cf. Section 4.5.7).

4.5.5 Robustness against Noise in Observation and Action Space

In real world applications, noisy measurements and actuation signals are a main contributor to the discrepancy between simulations and reality. In order to test our control framework's ability to be applied in the real world, its ability to handle noise both in action and observation space is tested. For this, a Gaussian distributed noise $d \sim \mathcal{N}(\mu, \sigma)$ is added to both action space ($\mu = 0, \sigma = 0.1$), and state space ($\mu = 0, \sigma = 0.5$). We found that the control framework is able to handle both. The observation noise is filtered by the Butterworth lowpass filter, while noise in the action space is handled by the robustness of the policy.

4.5.6 Comparison against other Control Methods

In order to compare the results obtained from the policy with other work, the disturbance is normalized. The applied force needs to be put in relation with the duration of the push, resulting in the impulse acting on the system. Furthermore, the mass and the resulting inertia is a crucial variable to a robot's ability to deal with disturbances. Therefore, the impulse is normalized by the weight of the robot.

The normalized impulse is used for comparison between the controllers of other work [72], [70], [81], [82] and the learned policy (Table 4.6). We compared sagittal and lateral pushes. For the sagittal push, two impulses are chosen such that foot stepping occurs for the larger impulse ($1.73 Ns/kg$), while the smaller impulse ($0.57 Ns/kg$) will result in a strategy without stepping. By comparing the rejectable normalized impulse of the strategies not taking a step (A, B, E, G), it can be seen that our policy performs similar (E: $0.57 Ns/kg$, G: $0.56 Ns/kg$) to the other controllers (A: $0.6 Ns/kg$, B: $0.52 Ns/kg$). For the strategies taking a step (C, D, F), our policy is able to perform better (F: $1.73 Ns/kg$) than the stepping controllers C ($0.52 Ns/kg$) and D ($0.6 Ns/kg$). Albeit our results are obtained from simulation whereas C and D are obtained from real experiments, we show in the next section that the generated motions are realistic and within the real physical constraints.

4.5.7 Realism of Generated Motions

Despite learning is trained in a simulator, we emphasize realistic motions by enforcing joint angle, velocity, and torque limits, which are the same as on the real Valkyrie robot. Therefore, the learned motion could be applied on the real Valkyrie robot without violating physical constraints. Table 4.7 compares the peak torques and velocities for different scenarios. The chosen scenarios require the largest joint torque for dropping and the largest joint velocities for

Table 4.6: Push disturbance from various push recovery studies

	A: Wieber 2006a [70]	B: Wieber 2006b [81]	C: Stephens 2010 [82]	D: Urata 2011 [72]	E: Sagittal push w/o foot stepping	F: Sagittal push w/ foot stepping	G: Lateral push w/o foot stepping
Robot	HRP-2	Biped model	Sarcos Primus	HRP3L-JSK	Valkyrie	Valkyrie	Valkyrie
Robot height $[m]$	1.539	1.425	1.575	N/A	1.8	1.8	1.8
COM height $[m]$	N/A	N/A	N/A	0.803	1.1	1.1	1.1
Mass $[kg]$	58	40	92	53	137	137	137
Force $[N]$	1500	750.0	N/A	597	600	2000	650
Interval $[s]$	0.025	0.025	N/A	0.05	0.12	0.12	0.12
Impulse $[Ns]$	37.5	18.8	42.0	29.9	72.0	240.0	78.0
Normalized impulse $[\frac{Ns}{kg}]$	0.6	0.52	0.52	0.6	0.57	1.73	0.56
Stepping	No	No	Yes	Yes	No	Yes	No
Simulated	Yes	Yes	No	No	Yes	Yes	Yes

Table 4.7: Peak torques and velocities for different scenarios.

	Peak joint torque $[N]$						Peak joint velocity $[rad/s]$					
	Torso pitch	Hip pitch	Hip Roll	Knee pitch	Ankle Pitch	Ankle Roll	Torso pitch	Hip pitch	Hip Roll	Knee pitch	Ankle Pitch	Ankle Roll
Joint limit	150	350	350	350	205	205	9.00	6.11	6.11	11.00	11.00	11.00
Nominal standing	68.4	39.5	57.1	122	44.9	46.2	0.0	0.0	0.0	0.0	0.0	0.0
0.55m Drop	150	221	350	350	205	205	4.69	2.01	6.11	9.95	11.0	11.0
$78Ns$ Pelvis (lateral)	104	147	103	264	117	106	0.15	0.58	0.46	1.09	0.96	11.0
$240Ns$ Pelvis (sagittal)	150	187	350	350	205	205	6.49	1.47	6.11	5.69	11.0	11.0

taking multiple steps due to large pushes at the pelvis. All other presented test cases required less joint torque and velocity than the the ones presented in Table 4.7.

4.6 Conclusion

In this chapter we proposed a learning framework which is able to learn a versatile unified control policy via Deep Reinforcement Learning. We found that the policy is able to deal with different types of disturbances and has comparable performance to conventional controllers. The policy acquired is capable of functioning in unseen situations, demonstrating that it is generalizing well over tasks. Furthermore, the proposed learning framework is learning algorithm independent. We showed successful balance recovery with a policy trained with three of the state-of-the-art DRL algorithms: TRPO, PPO, and DDPG. The emerging motions for push recovery are similar to human motions demonstrating the ankle, hip, foot-tilting, and stepping strategy. We compared the learned policy against traditional push recovery controllers and found similar disturbance rejection capabilities.

Chapter 5

Learning Walking Skills for Bipeds

In Chapters 3 and 4, we explained how deep reinforcement learning can be used to obtain robust 2D and 3D bipedal balancing policies. In this chapter, we use the learning framework from Chapters 3 and 4 as a basis for designing the learning framework for bipedal locomotion.

This chapter presents a new learning framework that leverages the knowledge from imitation learning, deep reinforcement learning, and control theories to achieve human-style locomotion that is natural, dynamic, and robust for humanoids. We proposed novel approaches to introduce human bias, i.e. motion capture data and a special Multi-Expert network structure. We used the Multi-Expert network structure to smoothly blend behavioural features, and used the augmented reward design for the task and imitation rewards. Our reward design is composable, tunable, and explainable by using fundamental concepts from conventional humanoid control. We rigorously validated and benchmarked the learning framework which consistently produced robust locomotion behaviours in various test scenarios. Further, we demonstrated the capability of learning robust and versatile policies in the presence of disturbances, such as terrain irregularities and external pushes.

5.1 Introduction

How can one advance the autonomous capabilities of legged robots by leveraging and incorporating 50 years of research on legged locomotion into a new paradigm [24], [83]? This chapter proposes a Deep Reinforcement Learning (DRL) Framework that is able to incorporate both prior research knowledge in legged locomotion and human reference motions for training the robotic gait. We investigate to what extent human inductive bias can and should be incorporated into a learning framework to *aid the exploration* while *not limiting the discovered motion*, and *generating realistic motions*. Human bias is mainly induced by incorporating imitation data, and designing a DRL framework that emphasizes on generating realistic, implementable, and energy-efficient motions.

Design choices reflecting human inductive bias for versatile motion on the real robot can

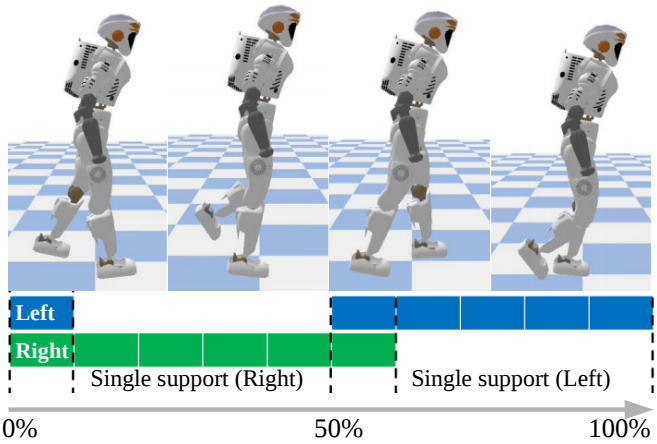


Figure 5.1: Natural human-like symmetric walking pattern generated by the learning framework. The blue and green bar represents left, right foot contact phases respectively.

be roughly categorized by the choices influencing (a) the training results, and (b) the behaviour during run-time. During training, one can influence the behaviour and success of the learning in the reward design, providing reference motions, designing appropriate training procedures, and selecting the appropriate network structure. For run-time, the designer needs to guarantee that the trained agent produces realistic and feasible motions that can be implemented on the real system while also keeping the reality gap between simulation and the real robot as small as possible. Here, we aim to provide novel technical approaches that lead to the success of both training and deployment of learned policies on simulated robots.

The key question we want to investigate is: *how can we induce more human bias for more realistic and natural looking motion*? We have studied three novel approaches in the framework design where bias can be introduced: 1.) initialization and termination of the state, 2.) selection of a task appropriate network structure, 3.) augmented reward design: task completion and imitation of reference motions.

The contribution of this chapter are summarized as follows.

- Novel approaches to introduce human bias in the framework generating human-like, realistic motions through imitation learning;
- A multi-expert network structure with smooth blending properties for humanoid bipedal locomotion;
- Integration and design of control system and the reinforcement learning framework for better replication of the results.

In the following, related research involving leveraging human knowledge in machine learning are briefly reviewed in Section II. Background information of the robot platform, simulation environment and control framework is presented in Section III. Next, the details involved in the

framework design is expanded on in Section IV. The training results are demonstrated and analyzed in Section V. Finally, the work is concluded in Section VI.

5.2 Related Work

Commonly, reinforcement learning attempts to automate the learning process to avoid over-using human engineering and thus preventing human bias by allowing the agent to infer the reward by itself [84]. While over-engineering and shaping the reward leave possibility of reward exploitation [62], carefully introducing human knowledge in DRL, such as specially designed network structures, can yield better results compared to baseline vanilla Fully-Connected Neural Networks (FCNN) [85]. Incorporating human prior knowledge into learning based methods is also known to increase the data efficiency [86]. To generate realistic motions, we review two main methods, i.e. imitating reference motions and network structure choice, in the following sections.

5.2.1 Leveraging Demonstrations

By incorporating human inductive bias such as reference motions from human expert demonstrations, the quality of the resulting motions can be improved. Learning from demonstration is a technique that extracts information from the reference motion generated by expert demonstrations to guide an agent. Notable examples include Behaviour cloning (BC) [32], Inverse Reinforcement Learning (IRL) [33], and Generative Adversarial Imitation Learning (GAIL) [34]. BC minimizes the difference between the student and expert behaviour in a supervised learning fashion. IRL predicts a reward function such that RL can reproduce the demonstrated motion. GAIL learns a discriminator to measure the similarity between demonstrations and behaviours generated by the policy.

Directly leveraging demonstrations in the reinforcement learning paradigm can be achieved through the use of a tracking reward [8]. This method involves designing a reward dedicated to measuring the similarity between the robot state and the demonstration dataset. The tracking reward will then be combined with the task reward.

5.2.2 Leveraging Human Knowledge in Network Design

Introducing human preferences by designing special network structures has shown to improve the overall performance and learning speed of the agent in multiple benchmarking simulation environments [85].

Residual policy learning methods involves hand-designing a base policy, and learning a residual policy that augments upon the base policy to adapt to external disturbance [87]. Many specially designed network structures fall into this category. For locomotion, a base periodic trajectory is hand designed to generate periodic gaits, and a residual neural network

is added upon to regulate the output of the base policy [43]. Additionally, this type of residual networks are not only used to construct policies, but also to augment simulators for more realistic dynamics [88].

Mixture of Experts (MoE) is a supervised learning architecture composed of of many separate experts, each of which is specialized for a subdomain of the task. A gating mechanism is responsible for selecting the required expert for a given input. Special neural network architectures inspired by MoE, such as Phase-Functioned Neural Networks (PFNN) [89] and Mode-Adaptive Neural Networks (MANN) [90], have been proposed to generate smooth locomotion behaviours for animation of humanoid and quadrupedal characters

Additionally, a bio-inspired approach uses the Central Pattern Generator (CPG), a neural oscillator, to model the functionality neural circuits within the spinal cord of verterbretes for rythmic movements. CPG is commonly used to construct the bio-inspired control policy for the locomotion of legged robots due to their ability to produce coordinated rhythmic and periodic gaits [91].

5.3 Learning Setup for Locomotion

In this section, we describe the specifications of the robot platform and the simulation environment, and detail the control structure and the human demonstration data.

5.3.1 Control Structure

The control framework contains two layers and is explained in detail in Chapter 2. The high-level control loop consists of a DRL agent that operates at 25Hz, while the low-level control loop consists of a PD controller that operates at 500Hz. The neural network generates desired joint angles for the low-level PD controllers that produce joint torques (c.f. Fig. 2.3). The gains for the PD controller can be seen in Table 5.1.

5.3.2 Robot Platform

The algorithm is simulated in an environment with the Valkyrie robot platform. Valkyrie has a total of 26 DOF, for which only the 15 lower body joints are actuated in this chapter: 3 waist joints (roll, pitch, yaw), and two 6 DoF leg joints (hip roll, hip pitch, hip yaw, knee pitch, ankle pitch, ankle roll).

5.3.2.1 Action Space

The outputs of the policy are the 15 target joint angles for the lower body joints of Valkyrie robot. The target joint angles are sent to the PD controller to be translated into torque for the

Table 5.1: PD gains for the joints of Valkyrie. Only the joints in the torso and lower body are actuated.

	Valkyrie	
	K_p (Nm/rad)	K_d (Nms/rad)
Torso yaw	1000	100
Torso pitch	3000	300
Torso roll	3000	300
Hip yaw	1000	100
Hip roll	1500	150
Hip pitch	2000	180
Knee pitch	2000	180
Ankle pitch	2000	120
Ankle roll	1500	90

joint motor. Low level joint position control was chosen over torque control as this has been shown to achieve better performance [25].

5.3.2.2 State Space

The policy only receives egocentric proprioceptive features as state observations. The state is adjusted to align with the gravity vector in relative coordinates, and is invariant to the yaw orientation of the pelvis in the world frame, and thus not affected by any steering in yaw orientation. The state space is chosen as in Chapter 4 and consists of joint angles and velocities, end-effector-to-pelvis vector, pelvis linear velocity and angular velocity, pelvis orientation relative to the gravity vector, Center of Mass (CoM) velocity, CoM-to-pelvis vector, and ground contact force. The observation is filtered using a low-pass Butterworth filter with a cutoff frequency of 10Hz.

An additional time input is added to the state to provide a time reference. Providing only reference motion for imitation learning without time will introduce temporal ambiguity, since the agent will not be able to infer the temporal relations of the reference motions. Time information is provided as normalized phase input that increments from 0 to 1 and resets after a certain time period [8]. This phase function, however, has an abrupt discrete jump from 1 to 0 that may cause non-smoothness in the learning behaviour in the neural network. To mitigate this problem, we generate smoothness by projecting the 1D phase onto a 2D unit-cycle (Fig. 5.2).

5.3.3 Human Motion Collection

The motion capture data is obtained using the Vicon motion tracking system. The reference features from motion capture data originate from a human of different morphology than the robot.

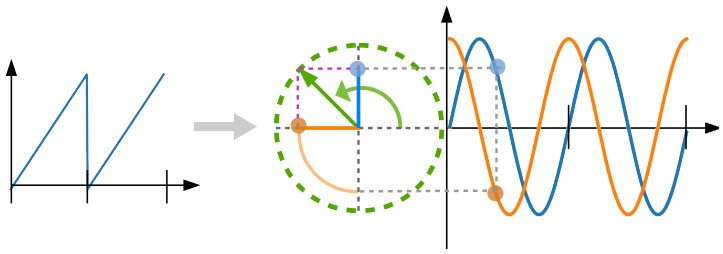


Figure 5.2: The 1D Sawtooth phase is projected onto a 2D unit-cycle for a smooth transition between each cycles.

Therefore, the human motion data has to be preprocessed before being used as reference for imitation learning. Since the human foot is narrower than the robot's, the reference hip yaw and hip roll angles are set to 0 to avoid the robot tripping on its own feet. Due to its limited influence on the gait, the torso roll and torso yaw are also set to 0.

The phase must be labeled for the human motion data and need to be used as a time reference to ensure that the state of the robot is aligned to the corresponding state of the human demonstration. The phase is labeled by observing the contact of both feet. We label the moment when the right foot makes contact with the ground with a phase of 0, the moment when the left foot makes contact with the ground with a phase of 0.5, and the moment when the next right foot makes contact with a phase of 1 (Fig. 5.1). These are then interpolated for the remaining in-between data points.

5.3.4 Deep Reinforcement Learning

The goal of reinforcement learning is to find an optimal policy that maximizes the discounted return. The discounted return is used as an evaluation of performance and is determined by summing the exponentially discounted reward,

$$R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}, \tag{5.1}$$

where T is the total number of samples in an episode and γ is the discount factor.

5.3.4.1 Choosing the Discount Factor

Alternatively to tuning the discount factor γ, the half-life of discounted future rewards can be used as a reference [54], [30]. For locomotion, a time horizon of one foot step with a duration around 0.5s is enough to plan a stepping strategy. At a frequency of 25Hz this equates to 13 time steps. We choose the discount value in a way that the half-life of the future reward occurs at 0.5s, meaning that the accumulated discount factor becomes 0.5 at 13 time step $\gamma^{13} = 0.5$, hence $\gamma \approx 0.95$.

5.3.4.2 Policy Optimization

In this chapter, we chose to use Proximal Policy Optimization (PPO) [15]. PPO is an on-policy DRL algorithm that tackles the problem of convergence by constraining the update step size through the use of clipped surrogate objective.

$$\begin{aligned} L^{\text{CLIP}} &= \mathbb{E}_t\left[\min(r_t(\pi_\theta)A_t, \text{clip}(r_t(\pi_\theta), 1-\epsilon, 1+\epsilon)A_t)\right] \\ r_t(\pi_\theta) &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, \quad \mathscr{L}_{\text{PPO}} = -L^{\text{CLIP}}. \end{aligned} \tag{5.2}$$

L^{CLIP} is the objective function that PPO maximizes. The term $\text{clip}(r_t(\pi_\theta), 1-\epsilon, 1+\epsilon)$ clips the probability ratio, discouraging large policy changes. Furthermore, the clipped objective $\text{clip}(r_t(\pi_\theta), 1-\epsilon, 1+\epsilon)A_t)$ is compared to the unclipped objective $r_t(\pi_\theta)A_t$, and the lower bound is chosen. The advantage A_t will be computed using generalized advantage estimator (GAE) [80].

5.3.4.3 Bounding the Action Space

Bounding the action space is important as the real robot systems have actuation limits. Using hyperbolic squashing functions $\tanh$ for bounding the action space have been shown to be disadvantageous due to saturation close to the boundary [65].

Instead of providing a hard constraint on the boundary of the network, we propose to implement a soft constraint by designing a bounding loss. Penalty is applied when the output of the neural network μ exceeds the lower Δ_{low} and upper Δ_{up} boundary, allowing the network to operate with outputs near the limit without saturation.

$$\mathscr{L}_{\text{bound}} = \begin{cases} 0, & \Delta_{\text{low}} \leq \mu \leq \Delta_{\text{up}} \\ 0.5(\mu - \Delta_{\text{up}})^2, & \Delta_{\text{up}} < \mu \\ 0.5(\Delta_{\text{low}} - \mu)^2, & \mu < \Delta_{\text{low}} \end{cases}. \tag{5.3}$$

The loss function $\mathscr{L}$ used for back-propagation is the sum of $\mathscr{L}_{\text{PPO}}$ and $\mathscr{L}_{\text{bound}}$:

$$\mathscr{L} = \mathscr{L}_{\text{PPO}} + \mathscr{L}_{\text{bound}}. \tag{5.4}$$

The action space of the actor network directly maps to the target joint angle of the robot's actuated motor. The lower Δ_{lower} and upper Δ_{up} boundary of the soft constraint correspond to the lower and upper boundaries of the physical joint. An additional hard clipping is applied in the environment to ensure that the output action does not exceed the physical limits of the joints.

5.4 Framework Design

The major contribution of this work presented in this chapter will be discussed thoroughly here: the approaches of introducing human bias, e.g. reference motions and specialized network design, into the learning framework through imitation reward design.

5.4.1 Reward Design

Reward design is an important aspect in reinforcement learning as it governs the behaviour of the agent. Our reward consists of a task term and imitation term similar to that presented in [8]. The task term provides the guidance necessary for the agent to achieve the locomotion objective, while the imitation term biases the behaviour towards a human-preferred walking pattern.

Due to their bounded nature within the range of $[0,1]$, Radial Basis Function (RBF) kernels are preferred for shaping the reward [29], [30], [54] and will be used in the following:

$$K(x,\hat{x},\alpha) = e^{\alpha(\hat{x}-x)^2}, \tag{5.5}$$

where x is the current state, and $\hat{x}$ is the desired value for that state. The hyperparameter α controls the width of the kernel. The correct choice of α largely impacts learning as it guides the gradient of the reward and thus gives crucial signals for correct credit assignment of the DRL algorithm.

The reward $r = w_1 r_{\text{imitation}} + w_2 r_{\text{task}}$ consists of an imitation term $r_{\text{imitation}}$ and a task term r_{task} weighted by the weights w_1, w_2. The imitation term encourages the robot to follow the human demonstration, while the task term encourages the robot to satisfy the task specific objective:

5.4.1.1 Imitation Reward

The imitation reward consists of a joint position tracking $r_{\text{joint\_pos}}$ and foot ground contact tracking r_{contact} (c.f., Table 5.2):

$$r_{\text{imitation}} = r_{\text{joint\_pos}} + r_{\text{contact}}. \tag{5.6}$$

The joint position reward $r_{\text{joint\_pos}}$ is calculated at phase ϕ by measuring the difference between the measured joint angle q_ϕ and the target joint angle reference $\hat{q}_\phi$. Furthermore, the foot contact has to match the contact configuration in the motion reference, represented by the support phase in the human motion data.

Table 5.2: Detailed description of imitation reward terms. The imitation reward terms are used to measure the distance between the generated and the reference motions. The corresponding normalization factor and weight for the reward terms are α and w.

Task reward terms			
Joint position	$K(\theta_i, \hat{\theta}_i, \alpha)$ (i represents joint index)	$\alpha = -56.69$	$w = 0.454$
Ground contact	$\begin{cases} 1, & \text{matches desired foot contact} \\ 0, & \text{does not match desired foot contact.} \end{cases}$		$w = 0.045$

Table 5.3: Detailed description of task reward terms. The terms are combined to construct the task reward. The corresponding normalization factor and weight for the reward terms are α and w.

Task reward terms			
Torso pitch	$K(\theta_n, 0, \alpha), n =$ torso pitch	$\alpha = -2.80$	$w = 0.017$
Torso roll	$K(\theta_n, 0, \alpha), n =$ torso roll	$\alpha = -2.80$	$w = 0.017$
Pelvis pitch	$K(\theta_n, 0, \alpha), n =$ pelvis pitch	$\alpha = -2.80$	$w = 0.017$
Pelvis roll	$K(\theta_n, 0, \alpha), n =$ pelvis roll	$\alpha = -2.80$	$w = 0.017$
Position in y axis	$K(y, \hat{y}, \alpha)$	$\alpha = -14.10$	$w = 0.034$
Position in z axis	$K(z, \hat{z}, \alpha)$	$\alpha = -14.10$	$w = 0.069$
Velocity in x axis	$K(\dot{x}, 1.0, \alpha)$	$\alpha = -6.91$	$w = 0.138$
Velocity in y axis	$K(\dot{y}, 0.0, \alpha)$	$\alpha = -6.91$	$w = 0.017$
Velocity in z axis	$K(\dot{z}, 0.0, \alpha)$	$\alpha = -6.91$	$w = 0.034$
Yaw velocity	$K(\omega_{\text{yaw}}, 0, \alpha)$	$\alpha = -11.21$	$w = 0.069$
Ground contact force	$K(F, 1342, \alpha)$	$\alpha = -2.45e-6$	$w = 0.017$
Torque penalty	$0.001 \cdot \sum_{j=0}^{11} (\tau^j)^2$		$w = 0.017$
Ground contact	$\begin{cases} 0, & \text{if no foot contact with ground} \\ 1, & \text{if upper body contact with ground.} \end{cases}$		$w = 0.017$
Alive bonus	$\begin{cases} 0, & \text{if no foot contact with ground} \\ 1, & \text{if upper body contact with ground.} \end{cases}$		$w = 0.017$

5.4.1.2 Task Reward

The task reward is the sum of multiple reward terms. Each individual reward term reflects different physical aspects of the system:

$$\begin{aligned} r_{\text{task}} = & r_{\text{pose}} + r_{\text{COM\_pos}} + r_{\text{COM\_vel}} + r_{\text{yaw\_vel}} \\ & + r_{\text{contact}} + r_{\text{fail}} + r_{\text{torque}}. \end{aligned} \tag{5.7}$$

The task reward for the locomotion objective is constructed using the designed principle presented in [30]. The chapter presents a reward design principle for bipedal balancing tasks. By setting the target COM velocity in the x axis to positive $0.5m/s$, the reward for balancing can be repurposed for locomotion objectives. For further details of the reward design, please refer to Table 5.3.

5.4.2 Network Design

Human locomotion is inherently periodic, and is therefore reasonable to incorporate structures and elements that are periodic in nature into the network design. In this work, we investigate the periodic characteristics of PFNN (Fig. 5.3a) and MANN (Fig. 5.3b) and their effects on locomotion tasks.

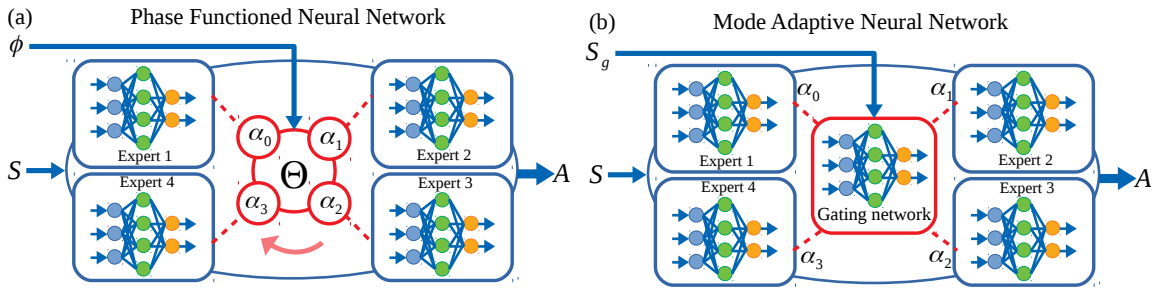


Figure 5.3: The detailed structure of PFNN and MANN. Both have a gating mechanism that generates the blending weights α_i, which are used to blend the expert networks to construct the prediction network.

Considering the unique gating mechanism in MANN and PFNN, the state input of PFNN and MANN differs from FCNN. For PFNN, the phase input is isolated from the other state and is fed into a phase function. For MANN, the input S_{g} for the gating network consists of the phase, joint positions and joint velocities. The expert networks for both PFNN and MANN have access to all state features except from the phase. For FCNN, all available states including the re-parameterized phase are sent into the network.

Unlike most networks where network parameters θ stay fixed during runtime, the parameters of a PFNN are function values that change depending on the phase variable ϕ. Within PFNN exists multiple individual sub-networks which we refer to as expert networks, each of which specializes in a particular task. The expert networks are not directly accessed, instead they are blended to reconstruct a separate network which we will refer as prediction network.

The parameters of the prediction network are computed during runtime by performing a weighted sum operation to blend the parameters of the expert networks:

$$\theta_{\text{prediction}} = \sum_{n}^{i} \alpha_i \theta^i_{\text{expert}}, \tag{5.8}$$

where α_i are the blending weights generated by a phase function $\Theta(\phi)$. For MANN, the phase function $\Theta(\phi)$ is replaced with a separate gating network $G(S_{\text{g}})$ to generate the blending weights α_i. For a detailed explanation of PFNN and MANN, please refer to [89], [90] respectively.

5.4.3 Sample Collection

In locomotion tasks, not all states are reversible. Due to the existence of gravity, the robot is naturally inclined to fall towards the ground. The distribution of the samples will thus be biased towards samples in which the robot is struggling on the ground to get up. Those samples are not necessarily good for the network to learn to achieve the desired locomotion tasks. We thus augment the sample distribution in favor of samples that are relevant to the tasks by changing both the initialization and termination of the episode in certain states:

5.4.3.1 Initializing Starting State

A disadvantage of fixed state initialization lies in the required time for the agent to learn to encounter high value states. Furthermore, the collected samples will suffer from a lack of diversity since it will be dominated by states close to the fixed initialization. The following approaches have been proposed to leviate this problem:

- *Initialize from demonstration:* In cases when demonstrations are available, it is common to initialize the agent with a state sampled from demonstration trajectory [8]. In the absence of demonstration states, hand-designed reference states can also be added [54].

- *Initialize from history samples:* Alternative to initializing from external demonstrations, selected, past samples as initialization states is applied [92].

- *Learning separate initialization policy:* A separate reset policy can be trained in parallel to the task policy. This method is necessary for learning in real world where instant reset is impossible [93].

In this work, we chose to use the approach of initialization from demonstration, since the human motion capture data used for imitation learning can also be reused to initialize the robot.

5.4.3.2 Early Termination

During early termination, the rollout is terminated when the robot reaches a terminal state. The terminal state can be either a successful goal state or an irreversible fail state. Early termination diversifies the sample distribution by increasing the reset frequency. For scenarios with irreversible fails states, termination and resetting the rollout prevents the samples from being dominated by fail states during the early learning stage:

- *Termination with physical criteria:* A common way to determine a irrecoverable fail state is to use a physical criteria from the environment, i.e. pelvis height and undesirable body contact for locomotion tasks [8].

- *Termination with Critic value:* A low critic value can be used as a makeshift criteria to determine a fail state [93] as fail states tend to have low reward, which would then be reflected in the learned critic value.

- *Early termination due to time constraint:* Even in cases where there are no irreversible fail state, it might be useful to terminate the episode after a prolonged period for more frequent reset to diversify the samples [94].

Initialization from demonstration can be combined with early termination to augment the sample distribution in a way that increases sample diversity. In our work, we initialized the robot using joint references from human motion capture data. The termination condition is triggered when the pelvis height is beneath 0.5m and when the upper body contacts with the ground. The episode will be also be terminated and reset after a prolonged period exceeding the time constraint of 30s.

5.5 Results

We first present the learning results compared to the achieved cumulative, undiscounted reward, and then quantitatively and qualitatively show the effect of imitation learning and the choice of the network structure in a comprehensive performance analysis based on three criteria: stability, robustness, and energy consumption.

5.5.1 Learning and Comparison Setup

All agents are trained on a commercial Intel i9 CPU equipped with a Nvidia 2080Ti GPU. For comparison purposes, each policy is trained for 400 iterations with 4096 steps for each iteration using either a FCNN, PFNN, or MANN network structure and converges after 48 hours. Each network consists of two hidden layers, the multi-expert networks use 4 experts with 128×128 neurons. MANN has an extra gating network with 32×32 neurons. To guarantee that the behaviours and performances originate from the network structure and is not constrained by the amount of neurons available, we trained with larger amounts of neurons.

At each iteration the cumulative reward is obtained by executing the deterministic policy on the current state. The reward for including imitation is calculated by $r = 0.5 \cdot r_{\text{imitation}} + 0.5 \cdot r_{\text{task}}$; and $r = 0.0 \cdot r_{\text{imitation}} + 1.0 \cdot r_{\text{task}}$ if no imitation is used (c.f. Section 5.4.1). At a maximum reward of 1 in a single timestep, the highest achievable cumulative reward for a complete episode is 750 at 750 time steps (30s).

Fig. 5.4 depicts the learning curves for different network structures averaged over 5 trials. With imitation reward provided, MANN obtains the highest reward followed by PFNN in close margins. Without imitation, the FCNN with 128 neurons (FCNN 128×128) has the highest reward.

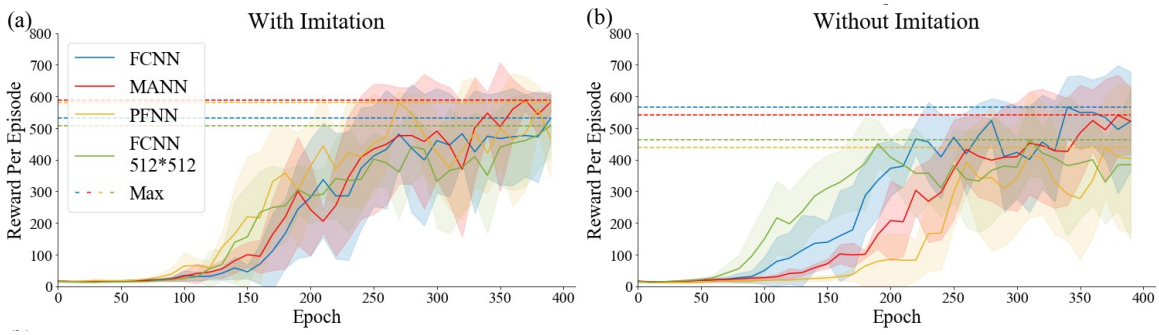


Figure 5.4: Learning curve for 4 different network setups averaged over 5 trials.

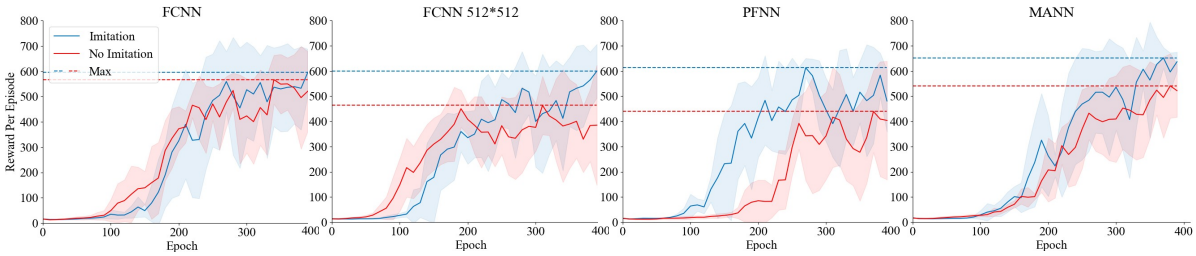


Figure 5.5: Learning curve of the task reward component. The addition of the imitation reward allows the agent to achieve the task objective much better, reflected by the higher task reward value.

5.5.2 Analysis of the Influence of Imitation Learning

The agent is able to learn a successful policy both with and without imitation reward. The task reward itself is sufficient to generate a locomotion behaviour. However, the most significant difference arises in the gait pattern. By including the imitation reward, the agent is able to learn a symmetric gait that resembles the human walking motion data from which it learns. Without imitation learning the agent learns an asymmetric leaping gait with one leg constantly in the front and the other at the back (c.f. Fig. 5.6b).

Introducing the imitation term not only creates a better, human-like gait pattern, but also improves the overall locomotion task performance of the policy. From Fig. 5.5 and Table 5.4 we can see that including the imitation term allows the agent to learn a policy that achieves a higher reward in the task term, meaning that adding human demonstration also allows the agent to achieve the locomotion task better. Amongst all combinations, MANN with imitation achieves the best performance with respect to the collected reward.

However, introducing an imitation term reduces the performance of the agent in terms of generalization (tasks and environments that the policy was not trained in), such as disturbances and walking over uneven terrain (c.f. Table 5.5). This decline in performance can be explained by the fact that the agent encounters less distinct states as the imitation reward encourages

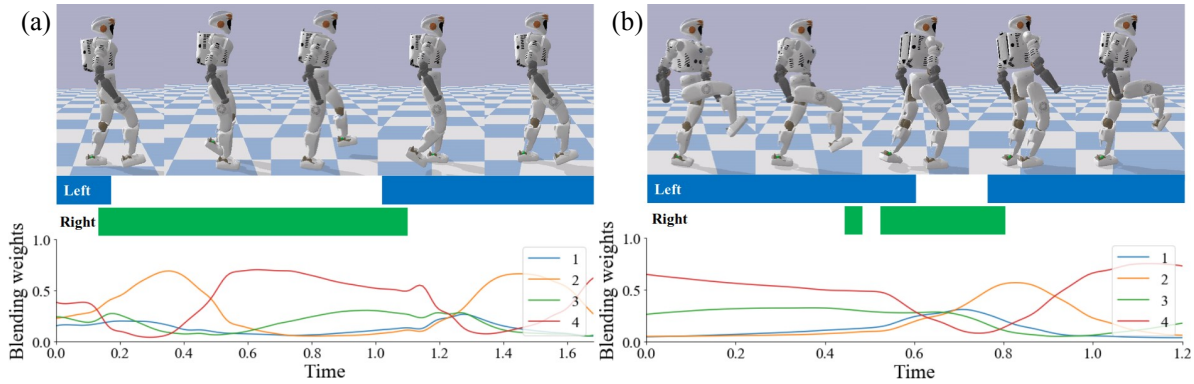


Figure 5.6: Locomotion behaviour of MANN (a) with and (b) without imitation. The blending weights fluctuate over a gait cycle, indicating that the primitive networks are activated differently during different gait phases, demonstrating specialization within the multi-expert structure. (a) a human-like gait pattern with symmetrically distributed left, right foot contact periods. (b) an un-human like gait pattern with asymmetric foot contact period.

Table 5.4: Maximum reward during each episode. MANN with imitation achieves the highest task reward.

Neural Network	w/ Imitation		w/o Imitation
	Imitation term	Task term	Task term
FCNN 128×128	465	596	567
PFNN	549	612	439
MANN	528	**652**	541
FCNN 512×512	416	599	463

the agent to be in states as close as possible to the reference motion.

5.5.3 Comparison Study between Neural Network Structures

The PFNN and MANN network consist of 4 individual experts with two hidden layers containing 128 neurons, therefore having 4 times the neurons. For fair comparison, we included another FCNN with 512 neurons in the hidden layer to even out the advantage of PFNN and MANN.

FCNN (512×512) performs poorly on locomotion in both with and without imitation learning scenarios even though having the same amount of neurons with PFNN and MANN, and 4 times the neurons of FCNN (128×128). Showing that the behaviour is not limited by the expressiveness of the network and increasing network size does not guarantee improvement. This further indicates that the high performance from MANN and PFNN is a result of the network structure rather than the increased number of neurons.

From Fig. 5.4, a difference for all three neural networks in convergence speed and the converged cumulative reward can be seen. With imitation learning, MANN and PFNN converges to a higher reward with faster speed compared to that of FCNN. Furthermore, it can be seen that PFNN performs poorly on the locomotion tasks without imitation reference provided. However,

Table 5.5: Performance analysis for imitation learning and different network structures.

Disturbance type	with Imitation				without Imitation			
	FCNN	MANN	PFNN	FCNN 512	FCNN	MANN	PFNN	FCNN 512
Impulse in [Ns]	280	280	300	290	470	420	400	**550**
Can blindly walk over stairs	✓	✓	✗	✗	✓	✓	✗	✓
Thrown cube weight in [kg]	5	4	6	7	10	7	7	**11**

PFNN is able to generate human-like symmetric periodic gaits without human reference due to its inherent periodic structure, unlike FCNN and MANN which relies on the human demonstration.

We recorded the output of the gating network of MANN to analyze the specialization of each experts. The blending weights α_i for each expert show a distinctive pattern that corresponds to the phase cycle of the walking gait (Fig. 5.6), which indicates that the MANN has an understanding that specialization is necessary for different instance in the phase.

5.5.4 Performance Comparison

Many research papers have used Genetic Algorithm (GA) to obtain the parameters for CPG [95], while some others have used policy search reinforcement learning [96]. However, training CPG with policy search requires special modification of the RL framework [96]. Also, due to its internal states, the CPG neuron is regarded as one type of recurrent neural network (RNN), thus training CPG with policy search requires Back-Propagation Through Time (BPTT). Therefore, CPG has not been included in our comparison study as it requires a different technique - BPTT - in order to be compatible with standard RL procedure. In contrast, both PFNN and MANN can be trained on the same basis using the same back-propagation technique of regular FCNN.

In the following, we analyze the performance for using imitation, and different network structures. The performance is evaluated based on the robustness, stability, and energy efficiency of the resulting gait.

5.5.4.1 Robustness

Different test scenarios are used to evaluate the robustness of the learned locomotion policy:

- *Push on pelvis:* We applied various amount of forces on the robot pelvis and observe how much disturbance the robot can withstand. The robot is able to withstand a impulse up to 550Ns (5500N over 0.1s), c.f., Fig. 5.7(a). A comprehensive comparison across all combinations is shown in Table 5.5. The policies that are trained without the imitation term are able to resist larger disturbances.

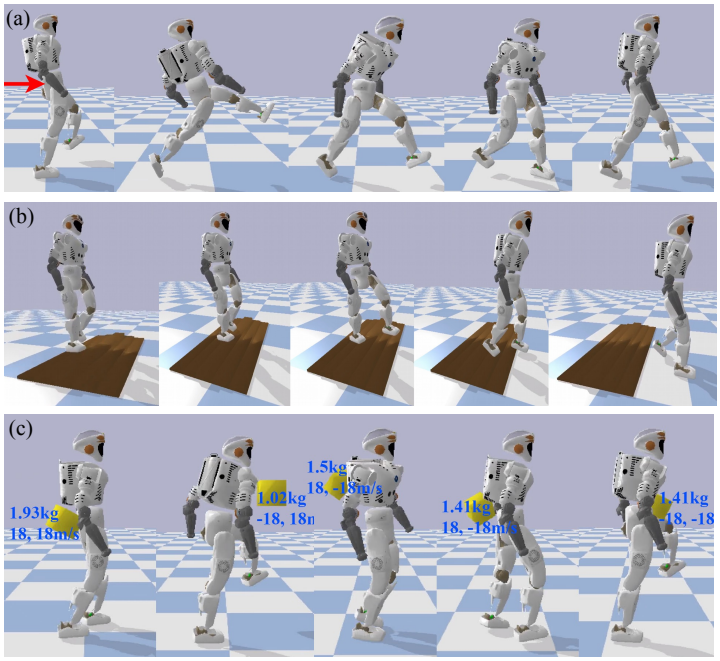


Figure 5.7: Robustness: (a) 550Ns impulse on pelvis; (b) walking over stairs with variable heights: 2.5cm, 5cm, 10cm, 5cm, 2.5cm; (c) constantly throwing cubes at 20m/s initial velocity.

Table 5.6: Performance analysis for imitation learning and different network structures.

	with Imitation								without Imitation							
	FCNN		MANN		PFNN		FCNN 512		FCNN		MANN		PFNN		FCNN 512	
Performance metric	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Distance CP to SP	0.141	0.083	0.154	0.099	0.193	0.131	0.147	0.089	0.182	0.104	0.233	0.106	0.248	0.156	**0.250**	0.116
Cost of Transport [W/m]	301	142	**258**	153	278	130	281	146	259	168	309	155	261	128	405	268

- *Blindly traversing uneven terrain:* Even though the agent is trained without external visual input and in a flat environment, some of the learned policy are able to generate a robust locomotion behaviour traversing over uneven terrain without falling (Fig. 5.7(b)).

- *Persistent small disturbance:* Lastly, we investigate how the policy behaves under frequent small disturbances. Random cubes of various weights are thrown towards the robot with an initial velocity of 20m/s (Fig. 5.7(c)). All policies are able to withstand disturbances induced by the cubes. The robot is able to withstand heavier cubes with policies trained without imitation.

5.5.4.2 Stability

To investigate the stability, the Capture Point (CP) is analyzed with respect to their shortest distance to the edge of the Support Polygon (SP), which are listed in Table 5.6. The larger the distance, the more stable it is as the CP have a larger margin before shifting out of the SP

Table 5.7: Peak torques and velocities of leg joints. Torso joints are omitted due to their limited influence on walking.

	Peak joint torque [N]						Peak joint velocity [rad/s]					
	Hip roll	Hip pitch	Hip yaw	Knee pitch	Ankle Pitch	Ankle Roll	Hip roll	Hip pitch	Hip yaw	Knee pitch	Ankle Pitch	Ankle Roll
Joint limit	350	350	190	350	205	205	6.11	6.11	5.89	11.00	11.00	11.00
FCNN 512	343	350	190	350	205	205	2.99	6.11	3.09	9.58	12.20	11.96
FCNN	234	350	158	350	205	205	2.40	4.69	2.75	6.61	7.01	11.00
PFNN	243	350	141	350	205	205	2.64	6.11	3.92	6.27	11.93	11.14
MANN	278	350	131	350	205	205	1.97	5.47	4.83	8.41	9.08	11.37

5.5.4.3 Energy Consumption

Energy consumption is an important aspect to be considered in reality, and hence, we are interested in whether including imitation reference or changing network structure influences the energy-efficiency of learned gaits. We analyze the *cost of transport* (energy consumed per distance traveled) for each policy, and found no significant correlation between the cost of transport and the training setups.

5.5.4.4 Physics Simulation Setting

The pybullet physics engine uses sequential impact solver to calculate the contact dynamics [1]. We set joint angle, velocity, and torque limits in the physics simulation the same as the real Valkyrie robot, so as to enforce the policy to learn motions that does not violate the physical constraints. Table 5.7 compares the peak torques and velocities from different policies. The ankle joint velocity occasionally exceeds the limit of 11.00 rad/s due to the large ground impacts.

5.6 Conclusion

In this chapter, we present key novel design approaches of a Deep Reinforcement Learning (DRL) Framework, and demonstrate that DRL is able to learn a robust, human-like walking policies that are reactive and robust against external disturbances. In particular, the diversity of behaviours and the variety of gait patterns exhibited during different test scenarios are all learned and emerged naturally, instead of being explicitly programmed as in the traditional approaches.

The comprehensive analysis on the influence of different network structures and imitation of human demonstrations shows that every structure has its advantage: if designed properly using the framework in this chapter, FCNN provides a good baseline that is able to generalize

well and withstand large disturbances due to the intentionally designed training. If no reference for imitation learning is available, PFNN is able to generate periodic, symmetric gaits due to its inherent periodic structure. If a reference is available, MANN using imitation learning is able to accomplish the best task performance. For all network structures, introducing human demonstration proves to be beneficial for locomotion tasks, which is reflected by the increased value of the task objective.

Though we have applied realistic joint torque and velocity constraints in the simulation, the learned control policy will have difficulty to be directly transferred on a real system due to the discrepancies between simulation and the real world, such as variations in mass distribution, friction, and contact dynamics etc. The next step is to bridge the gap between simulated dynamics and real-world physics and deploy the learned policies on a real robot. Further research on real-world deployment of DRL based control policies can be seen in Chapter 7. Furthermore, in Chapter 7, we have also continued the research on multi-expert network structures and developed our own Multi-Expert Learning Architecture.

Chapter 6

Learning Fall Recovery Skills for Bipeds and Quadrupeds

In Chapters 3, 4, and 5, we addressed balancing and walking, both fundamental aspects of bipedal locomotion. However, there are other skills that are important for locomotion in real-world environments. In real-world environments, there are many unexpected perturbations. The robot will likely fall. Hence, a robot needs to have the ability to stand up and recover from a fall so as to continue to pursue the desired locomotion task.

The ability to recover from a failure caused by a fall is an essential skill for traversing over rugged terrains successfully. A common approach is to handcraft a trajectory, which requires significant amount of engineering effort. Handcrafted trajectories also suffers from lack of generalization, cannot be used for different robot models, and are prone to failure in corner cases. In this paper, we presented a framework based on model-free deep reinforcement learning to learn a control policy that is able to utilize its whole body and limbs to generate a standing-up recovery maneuver. We solved the challenge of reactive and dynamic online replanning by learning a highly nonlinear versatile feedback policy represented by a deep neural network. This learning-based approach requires minimal human engineering effort and is capable of generalizing across robots with different morphologies and scales– humanoids and quadrupeds. The effectiveness of the framework has been further validated by the robust performance of the policy while deployed on a real-world quadruped.

6.1 Introduction

Humanoid robots have a morphology and mobility similar to that of humans, which makes humanoid robots better suited to environments designed ergonomically for humans than other kinds of mobile robots. Such an ability provides humanoid robots with greater potential for integrating into human society and everyday life. However, a humanoid robot has a small supporting region consisting of just two feet, thus making it unstable and very prone to falling.

Failure-resilient locomotion is crucial for the operation of legged robots, including humanoid and quadruped robots, in unstructured environments. It is necessary for deployment in disaster response, rescue missions, and scientific expeditions, which all feature unpredictable circumstances. When a robot does fall, it is important for it to be able to recover into a canonical operating state and continue the operation of the desired tasks. The DARPA robotics challenge is a good real-world example illustrating the importance of fall recovery for humanoid robots. The tasks within the DARPA challenge focus on disaster or emergency-response scenarios in unstructured environments that are extremely challenging. Despite tremendous engineering efforts on the part of all the competing teams and the use of cutting-edge technologies, only the WPI-CMU team's Warner robot avoided falling. Among the robots that fell over, Team Tartan's Chimp robot was the only one able to stand up after falling.

Humans and animals are remarkably resilient to fall failure, as they have the ability to recover from any falling posture, making them good sources of inspiration when designing controllers for fall recovery maneuvers. Many fall recovery controllers for humanoids are constructed by handcrafting a trajectory that resembles a human fall recovery maneuver through a heuristic and labor intensive process [97],[52],[51],[50]. The heuristic approaches require strict conditions to operate properly and do not generalize well to corner cases which are not accounted for during the design process. Other methods automate the process by calculating the trajectory depending on the specific falling posture offline [98], [99]. These automated approaches are able to operate under a wider range of fall postures. However, they still lack the responsiveness needed to react to external disturbances in real time.

A promising alternative for obtaining fall recovery motions is model-free reinforcement learning (RL). In this paradigm, an agent interacts with its environment and learns the control policy through a process of trial and error. One major benefit of using RL is that it requires less prior knowledge from human experts and is less labor intensive compared to the handcrafted approach. RL has been used successfully for learning fall recovery policies in humanoid and quadruped robots [53], [54].

In this research, we design a responsive feedback controller by training a neural network policy using Deep Reinforcement Learning (DRL). The controller is able to generate the necessary movements for a humanoid robot to recover to an initial upright standing posture given any initial fall configuration while also responding to an external disturbance. The aim of the project is to investigate the use of DRL to generate a robust humanoid fall recovery controller. The task is to reach an upright standing configuration with two feet in contact with ground from a random fall configuration. The result shown in this research will serve as a proof of concept and inspire future researchers to use DRL for humanoid control. The successful simulation results suggest that it could potentially be used to fully automate humanoid robots able to move in a non-structured environment outside a robotics laboratory.

6.2 Related Work

There has been progress in the robotics community in terms of designing a controller to enable humanoids to stand up and recover from a fall. For humanoids, standing back up on two feet is not a trivial task, as it involves multiple contact points with the ground. Such a scenario is difficult to model and poses many challenges for optimization-based controllers. Fall recovery in humanoid robots has been achieved with various different approaches. One common strategy is to handcraft a standing motion by imitating the standing motion of humans. Stückler et al. have designed a controller for standing up by scripting the target joint angles of the entire trajectory of the standing routine manually [50]. Kanehiro et al. designed a controller for fall recovery for the HRP-P2 robot by constructing a graph consisting of the key contact states within the standing motion and devising a Zero Moment Point (ZMP)-based controller for the transitions between contact states. This controller has been successfully deployed in a real HRP robot, which was able to stand up task from supine and prone positions [51].

Compared to humanoid robots, quadrupedal robots are more stable and are less prone to failures involving a fall and rendering the robot inoperable. Nevertheless, there are still quite a few papers that have tackled fall recovery in quadrupedal robots. Semini et al. handcrafted a self-righting recovery sequence for the HyQ2MAX quadruped robot manually [52]. Castano et al. designed a finite-state machine to achieve fall recovery for the wheeled quadrupedal robot CENTAURO [100]. The fall recovery designed for the CENTAURO robot is unique, as the CENTAURO is a non-standard quadruped that has an upper body with two arms, resembling the mythical centaur. Thus, the fall recovery policy involves utilizing the upper arms.

DRL has shown incredible results in many different fields in recent years. Previous research has shown that DRL can be successfully applied to learn fall recovery for quadrupedal robots like ANYmal. Model-free reinforcement learning appears to be a valid autonomous and intelligent alternative for solving the problem of fall recovery. With model-free RL, the learning agent is able to obtain the policy through interactions with the environment, avoiding the need to model complex real-world dynamics explicitly. The effectiveness of RL for robot control has been validated by existing works. Jeong et al. applied Q-learning to solve the problem of fall recovery for humanoids by discretizing the state and action spaces of the simulated robot [53], while Lee et al. has trained a fall- recovery policy, applied proximal policy optimization to learn a continuous fall recovery control policy, and successfully deployed the policy on the real ANYmal quadruped robot [54].

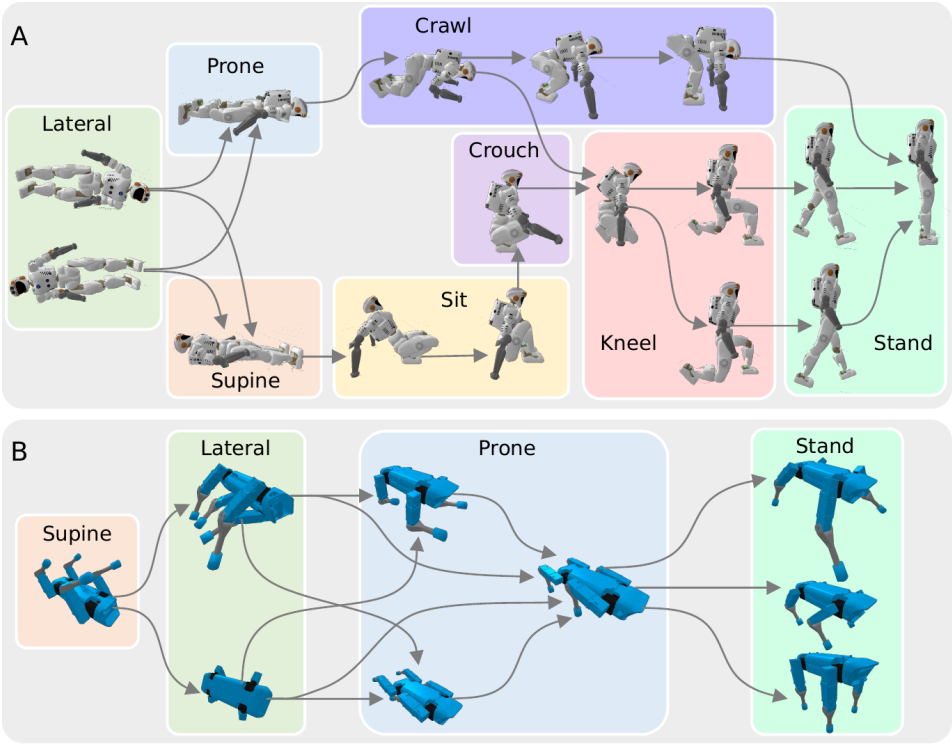


Figure 6.1: (A) The control graph for a standing-up sequence designed for Valkyrie robot, consisting of a total of 15 key poses that will be used as initialization states. (B) The control graph for a standing-up sequence designed for the Spotmicro robot consisting of a total of 9 key poses that will be used as initialization states. The graph only shows a small set of solutions for fall recovery motions; there are many other feasible solutions that lead to successful fall recovery.

6.3 Methodology

6.3.1 Complexity of Fall Recovery Motions

Fall recovery can be viewed as a complex motor behaviour in the context of multi-contact locomotion tasks using the concept of contact-richness, i.e., the greater the contact-richness, the more complex the motion. The richness of contact can be reflected by the potentially large number of unpredictable time-varying points on different body segments making contact with the environment during the motion. Using this concept, in order to perform a successful fall recovery motion, a robot is required to utilize multiple body segments to interact with the environment through multiple contacts.

However, both in simulation and reality, the number of contact points is difficult to specify for particular body segments which are in contact with the environment. Borràs et al. proposed a taxonomy of whole-body poses for analyzing multi-contact motions of humanoid robots [101], [102]. Inspired by the simplification of contact poses within the taxonomy, we simplify our

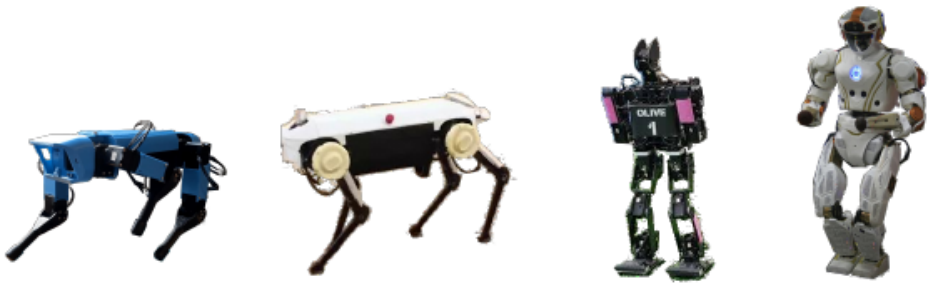


Figure 6.2: The four robot models investigated. From left to right: (i) Spotmicro, (ii) Jueying Pro, (iii) Sigmaban, and (iv) Valkyrie.

explanation of contact-richness by focusing on how many body segments are in contact at each timestep instead of contact points and how the number varies with respect to time during the motion. We further explain body segments in contact and its variation with time as contact locations and contact transitions (or contact timing), respectively. In summary, the complexity of motions for multi-contact cases can be evaluated in terms of contact locations and contact transitions.

We handcrafted the humanoid and quadruped fall recovery contact transitional graphs (Fig. 6.1) to show the possible transitions between multiple contact configurations (body segments in contact at a certain time) capable of enabling successful fall recovery. Here, the complexity of fall recovery behaviours are reflected in the following aspects: (i) there exist multiple fall recovery strategies, i.e., transitions of different combinations of contact configurations, for a single initial fall posture; (ii) each strategy requires transitions between multiple different contact configurations to succeed; (iii) each contact configuration involves multiple body segments coming into contact with the ground. Moreover, Fig. 6.1 shows just a very small set of handcrafted solutions. Many other feasible solutions exist in reality that are capable of leading to successful fall recovery motions, but these solutions are difficult for humans to discover or design manually, making fall recovery a challenging and complex motion.

6.3.2 Robot Model

Learning fall recovery is challenging for a humanoid due to its inherently unstable nature. Therefore, we start with the less challenging quadrupeds and then work towards learning fall recovery for humanoids. We ultimately test the algorithm on two humanoid and two quadruped robot models. The purpose of using multiple models is to determine whether the proposed learning framework can be generalized to different robot morphologies. The four robots in question are: (i) Spotmicro, (ii) Jueying Pro, (iii) Sigmaban, and (iv) Valkyrie.

The Spotmicro is an open-source community project that aims to replicate the Spotmini with a smaller and more miniature form-factor. It uses servo motors as actuators for the 12 leg joints. The Jueying Pro robot is a high- performance quadruped that is capable of performing

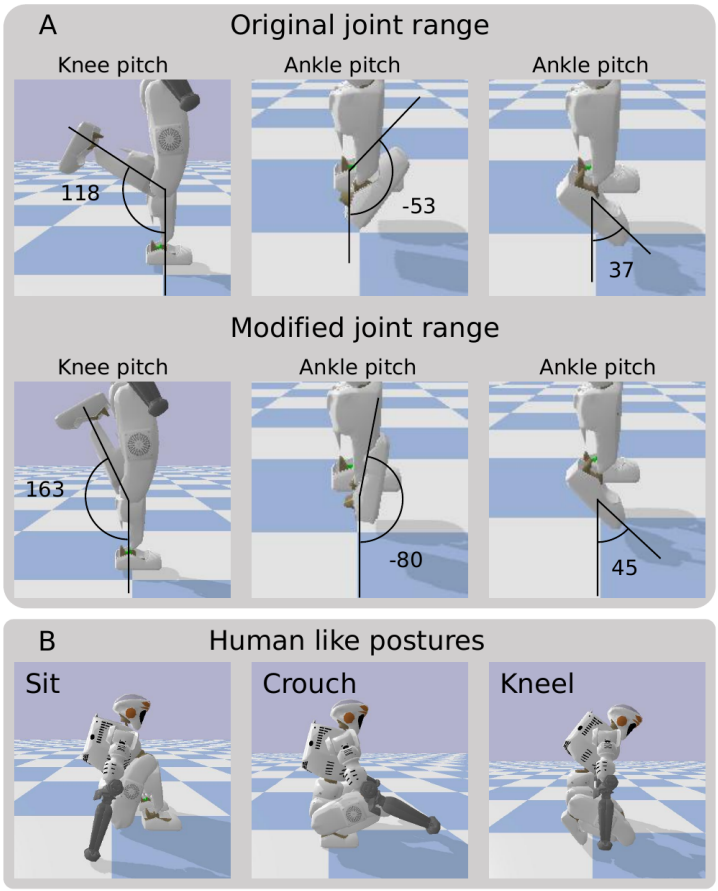


Figure 6.3: Joint configuration of the Valkyrie robot. (A) Original joint range (top row) and modified joint range (bottom row) of the Valkyrie robot. (B) Human-like key postures during standing up that are enabled by the modified joint range.

jumping maneuvers. Jueying Pro weighs 70kg and has 3 DOF per leg, with a maximum torque limit of 100 Nm for the motors in the hip and knee joints.

The Sigmaban robot was designed for the robocup soccer tournament, in which robots will constantly tip over due to collisions from other robots [103]. Therefore, the ability to recover from a fall was already taken into consideration during the robot hardware design process. Valkyrie is a 44-DOF robot that was designed by NASA in 2013 to operate in harsh environments that would be too dangerous for humans. It stands 1.87m tall, weighs 129kg, and is the byproduct of years of NASA research on humanoid robotics.

Simulations for Jueying Pro, Spotmicro, and Sigmaban use the original joint ranges found on the real robots, while the joint range for the Valkyrie robot has to be modified in simulation. The original joint limit of the Valkyrie robot is too restrictive and does not allow for human-like standing-up behaviour. We modified the joint limit and collision mesh to allow more dexterity in the legs, increasing the resemblance to humans. The collision mesh has also been slightly altered to enable the increased range of motion without causing self-collisions. The modified

Table 6.1: This Table provides the basic definitions of the mathematical notation used in the equations for the reward terms shown in Table 6.2

Nomenclature	
φ_{base}	Orientation vector: the projection of the gravity vector in the robot base frame to represent the orientation
φ_{torso}	Orientation vector: the projection of the gravity vector in the robot torso frame to represent the orientation
h_{base}	The robot base height (z) in the world frame
h_{head}	The robot head height (z) in the world frame
v_{base}	The linear velocity of the robot base in the world frame
τ	The vector of all joint torques
q	The vector of all joint angles
$\dot{q}$	The vector of all joint velocities
$(\hat{\cdot})$	The desired quantity of selected property $(\cdot)$, where $(\cdot)$ serves as a placeholder
$p_{foot,n}$	The n-th foot horizontal placement in the world frame
p_{base}	The horizontal xy coordinates of the base in the world frame

Valkyrie robot is able to perform human-like squatting, crouching, kneeling, and sitting motions, as shown in Figure 6.3 , which are crucial joint configurations during the human standing-up sequence. Such actions could not have been achieved with its original joint limit and collision mesh.

6.3.3 Reward Design

The design of the reward function for fall recovery follows the same design rule as the balancing reward presented in previous work [29], [30], [37]. The fall recovery task is similar to a balancing task, as fall recovery focuses on recovering a stable standing configuration and maintaining balance while in that stable configuration.

We used a Radial Basis Function (RBF) to design the bounded reward function:

$$K(x, \hat{x}, \alpha) = e^{\alpha(\hat{x}-x)^2}, \tag{6.1}$$

where x is the physical quantity used for the evaluation, $\hat{x}$ is the desired value, and α is the parameter that controls the width of the RBF. All reward terms composed of continuous physical properties utilize this RBF. The exceptions are the three reward terms for discrete properties, i.e., foot-ground contact, body-ground contact, and ground contact for the imitation gait. Details of the individual reward terms using the nomenclature Table 6.1 are presented in Table 6.2.

Table 6.2 shows the list of reward terms designed for humanoid fall recovery. The base height and head height terms are the two main terms that are responsible for the standing-up motion. They encourage the robot to stand up and maintain the desired height. The upper torso and base poses are responsible for regulating the upper body posture of the robot. The base velocity terms encourages the robot to learn a smooth standing-up motion without jerky motions

Table 6.2: Detailed description of task reward terms for humanoids. The corresponding normalization factor and weight for the reward terms are α and w.

Task reward terms			
Base pose	$K(\varphi_{base},[0,0,-1],\alpha)$	$\alpha=-2.35$	$w=0.0625$
Base height	$K(h_{base},\hat{h}_{base},\alpha)$	$\alpha=-4.60$	$w=0.25$
Upper torso pose	$K(\varphi_{torso},[0,0,-1],\alpha)$	$\alpha=-2.35$	$w=0.0625$
Head height	$K(h_{head},\hat{h}_{head},\alpha)$	$\alpha=-1.59$	$w=0.250$
Base velocity	$K(v_{base}$,[0,0,0],$\alpha)$	$\alpha=-4.60$	$w=0.125$
Joint torque regularization	$K(\tau,0,\alpha)$	$\alpha=-5.11e-5$	$w=0.0625$
Joint velocity regularization	$K(\dot{q},0,\alpha)$	$\alpha=-0.046$	$w=0.0625$
Body-ground contact	$\begin{cases}0, & \text{upper body contact with ground}\\ 1,.\end{cases}$		$w=0.0625$
Left foot placement	$K(p_{foot,left},p_{base},\alpha_p)$	$\alpha=-14.10$	$w=0.03125$
Right foot placement	$K(p_{foot,right},p_{base},\alpha_p)$	$\alpha=-14.10$	$w=0.03125$

Table 6.3: Detailed description of task reward terms for quadrupeds. The corresponding normalization factor and weight for the reward terms are α and w.

Task reward terms			
Base pose	$K(\varphi_{base},[0,0,-1],\alpha)$	$\alpha=-2.35$	$w=0.278$
Base height	$K(h_{base},\hat{h}_{base},\alpha)$	$\alpha=-51.16$	$w=0.278$
Base velocity	$K(v_{base}$,[0,0,0],$\alpha)$	$\alpha=-18.42$	$w=0.167$
Joint torque regularization	$K(\tau,0,\alpha)$	$\alpha=-0.003$	$w=0.111$
Joint velocity regularization	$K(\dot{q},0,\alpha)$	$\alpha=-0.026$	$w=0.111$
Body-ground contact	$\begin{cases}0, & \text{upper body contact with ground}\\ 1,.\end{cases}$		$w=0.056$

by penalizing high velocity. The joint torque regularization and joint velocity regularization terms penalizes high torque in, and high velocity of, the joints, respectively. The body ground contact reward rewards the agent whenever upper body parts are not in contact with the ground. The foot placement reward encourages the left and right feet to be placed close to the projected pelvis position on the ground, resulting in a more human-like standing posture. Without this reward term, the agent will still be able to learn to stand up, but the resulting pose might look unnatural.

Table 6.3 shows the list of reward terms designed for quadruped fall recovery. The reward for quadrupeds is similar to that for humanoids, with a few differences to accommodate the different robot morphology. The head height term, upper torso pose term and the two foot placement terms are removed for the quadrupeds, while the other reward terms remain the same.

6.3.4 Deep Reinforcement Learning

6.3.4.1 Soft Actor Critic

The task of fall recovery is more challenging than the balancing and locomotion tasks, as it involves more degrees of freedom and more complex contact situations. For balancing and walking, only the lower body of the bipedal robot is needed, and the contact is restricted to that between the feet and the ground. Whereas, for fall recovery, the bipedal robot has to utilize its entire body fully.

In such a case, more samples will be required to obtain a reasonable policy. Therefore, off-policy DRL algorithms are more favorable than off-policy algorithms in these scenarios. We chose to use Soft Actor Critic (SAC), an off-policy DRL algorithm, due to its sample efficiency [17]. SAC optimizes an expected sum of rewards augmented with an additional maximum entropy objective as:

$$J_{SAC}(\pi) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}(r(s_t,a_t)) + \alpha H(\pi(\cdot|s_t)), \tag{6.2}$$

where $\sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}(r(s_t,a_t))$ is the expected sum of rewards, and $H(\pi(\cdot|s_t))$ is the expected entropy of the policy π over the sample distribution ρ. The temperature parameter affects the stochasticity, and thus the exploration capability, of the optimal policy by changing the importance of the entropy term, where a higher temperature leads to more stochastic policies and vice versa. SAC balances the well-known problem of exploitation and exploration by tuning the temperature parameter α automatically and thus governing exploration. The policy is expressed as a Gaussian distribution $\mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)^2)$ with mean $\mu_\theta(s_t)$ and covariance $\sigma_\theta(s_t)$ generated by a neural network. A tanh-squashing function is applied to the Gaussian samples to project an action distribution within $\hat{a}_{s_t} \in (-1, 1)$. The squashed action $\hat{a}_{s_t}$ is then scaled by the joint range $\bar{q}$ and formulates the target action $a_{s_t} \in \bar{q}$ as joint angle references for the robot.

6.3.5 Sample Distribution Augmentation

As described in various papers, the sample distribution affects the learning of the control policy [8], [37]. There are two approaches for augmenting the sample distribution: (i) initializing the starting state, and (ii) early termination. Both approaches can be combined to augment the sample distribution in a way that increases sample diversity.

6.3.5.1 Initializing the Starting State

A disadvantage of fixed-state initialization lies in the time required for the agent to learn to encounter high-value states. Furthermore, the collected samples will suffer from a lack of diversity since the collection will be dominated by states close to the fixed initialization.

The key postures within the handcrafted contact transitional graph shown in Fig. 6.1 will be sampled randomly for starting states to initialize the robot during simulation. The postures shown in Figure 6.1A will be used to initialize the training of the Valkyrie robot. The same initial postures are used for the Sigmaban robot after slight adjustments to the height and joint angles to accommodate its different size and joint configuration. The postures shown in Figure 6.1B will be used to initialize the training of the Spotmicro robot. The same initial postures are used for Jueying Pro after slight adjustments to the height and joint angles to accommodate its different size and joint configuration.

6.3.5.2 Early Termination

Due to the existence of gravity, the humanoid and quadruped robots are naturally inclined to fall towards the ground. During the early training iterations, the distribution of the samples will be dominated by samples in which the robot is struggling to get up from the ground due to the existence of gravity. Having many samples with the same falling configuration is not necessarily a good way for the network to learn the standing-up behaviours necessary for achieving fall recovery due to the lack of diversity. Therefore, we set a time limit of 10s for the early termination of the episode and initialize the episode in different states to ensure the diversity of the experienced samples.

6.3.6 Action Filtering

A common phenomenon encountered with RL in simulation is that the learning policy takes advantage of the pure torque source, which has unlimited control bandwidth, and learns to generate abrupt, jerky motions in simulation [104], [105]. We implement a first-order Butterworth filter with a cut-off frequency of 5Hz on the output action to smooth out the action, i.e., restrict high-frequency actions and prevent the policy from overly exploiting of such risky motions. The cutoff frequency of 5Hz is set according to our experience with human motion analysis and robot control. We have observed that the frequencies of human (animal) motion normally do not exceed 5Hz in common cases; therefore, 5Hz is a reasonable choice for the cutoff frequency.

6.3.7 Smoothing Loss

The previously proposed action filter is not sufficient for generating smooth motion. As an action filter only limits the frequency of the action but not the amplitude, the policy may still explore and utilize low-frequency, large-amplitude motions with large torques to achieve a task. Large joint torques lead to large interaction forces with the environment that can cause abrupt motion and lead to instability. Hence, to encourage the policy to generate motions with

Table 6.4: PD gains for Spotmicro and Jueying Pro.

	Spotmicro		Jueying Pro	
	K_p (Nm/rad)	K_d (Nms/rad)	K_p (Nm/rad)	K_d (Nms/rad)
Hip roll	10	0.1	700	10
Hip pitch	10	0.1	700	10
Knee pitch	10	0.1	700	10

smooth torque profiles, we designed a special loss function, called a smoothing loss function $J_{smoothing}$:

$$J_{smoothing}(\mu(s_t)) = \left\| \mu(s_t) - q \right\|, \tag{6.3}$$

where the $\mu(s_t)$ are the deterministic mean outputs of the stochastic policy that are used as joint references, and the q are the measured joint angles. The smoothing loss $J_{smoothing}$ is the objective function that minimizes the difference in joint angles between the target $\mu(s_t)$ and the current measurement q. As the joint commands are the inputs for PD control, this minimization leads to minimal joint torques, thus encouraging the policy to learn smoother motion with less effort.

The smoothing loss $J_{smoothing}$ is added to the standard SAC training loss $J_{SAC}(\pi)$, and is used for backpropagation of the neural network. Instead adding it to the reward function, the smoothing loss term is added to the neural network loss function, which allows the information to backpropagate directly through the neural network and bypass the reward bootstrap and Q-function approximation process, thus removing the "wait time" needed for the Q-function to obtain a valid approximation of the expected return.

6.3.8 Control Framework

The control framework consists of two layers, a high-level neural network policy that generates position references for all joints at 25 Hz, and a low-level PD controller that generates torque at 500Hz. An update frequency of around 25 30 Hz is a common setting for a high-level motion planning layer in robot control [6],[8],[106],[54]. The PD gains are tuned differently for the four robot models used in this work (Table 6.4 and Table 6.5).

6.3.8.1 State Representation

The state representations selected for the fall recovery control policy are: (i) gravity vector, (ii) base angular velocity, and (iii) joint position. The base angular velocities and all the joint positions are measured directly using an Inertial Measurement Unit (IMU) and joint encoders. The gravity vector is a 3D unit vector pointing along the direction of gravity in a local frame of

Table 6.5: PD gains for Sigmaban and Valkyrie. Valkyrie consists of more joints compared to Sigmaban

	Sigmaban		Valkyrie	
	K_p (Nm/rad)	K_d (Nms/rad)	K_p (Nm/rad)	K_d (Nms/rad)
Torso yaw	1900	38	N/A	N/A
Torso pitch	1500	30	N/A	N/A
Torso roll	1500	30	N/A	N/A
Shoulder pitch	1900	19	23.54	0.48
Shoulder roll	1900	38	23.54	0.48
Shoulder yaw	650	6.5	N/A	N/A
Elbow pitch	650	13	23.54	0.48
Hip yaw	1900	38	47.08	0.96
Hip roll	3500	70	47.08	0.96
Hip pitch	3500	70	47.08	0.96
Knee pitch	3500	70	47.08	0.96
Ankle pitch	2050	20.5	47.08	0.96
Ankle roll	2050	10.25	47.08	0.96

the robot base. Other similar learning-based approaches include the linear velocity and height of the robot in the state input. However, considering that the height and velocity cannot be measured directly, we omit them from the input. We anticipate that height and linear velocity are not necessary for learning a successful fall recovery policy. The state input dimension can be seen in 6.6.

Table 6.6: State input dimension.

Physical quantity	Input dimension			
	Spotmicro	Jueying Pro	Sigmaban	Valkyrie
Gravity vector	3	3	3	3
Base angular velocity	3	3	3	3
Joint position	12	12	18	20
Total	18	18	24	26

6.4 Results

We first present the results from the relatively simple task of fall recovery for quadrupeds and then proceed to present the results from the more challenging fall recovery task for humanoids. We then finish up the section by presenting the implementation results on a real Jueying Pro

quadruped platform. The learning curve for the traiining of the fall recovery policies can be seen in Fig. 6.4.

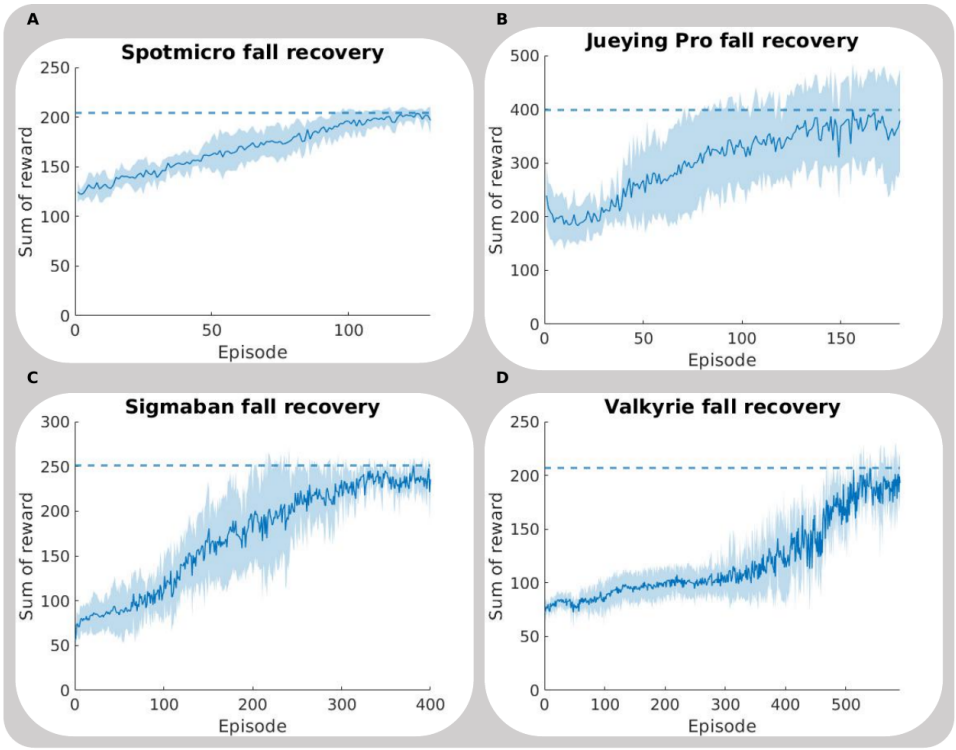


Figure 6.4: Learning curve for the fall recovery policies of (A) Spotmicro, (B) Jueying Pro, (C) Sigmaban, (D) Valkyrie. The results are averaged over 5 trials, each with a different random seed. All 5 trials are able to obtain a successful fall recovery policy.

6.4.1 Fall Recovery on Quadrupeds

Compared to fall recovery for humanoids, fall recovery for quadrupeds is relatively simple due to their lower Center of Mass (COM) heights and larger support polygons. We have trained a policy to successfully perform fall recovery motions on the Spotmicro robot and Jueying Pro robot, as shown in Fig. 6.5 and 6.7. In particular, the fall recovery policy for Spotmicro is able to recover from supine, prone, and left and right lateral postures. Due to the curved geometry of the Lidar sensor on the top of its body, the Jueying Pro is unable to lie in a supine posture, therefore the fall recovery policy for Jueying Pro is only trained to recover from a lateral posture.

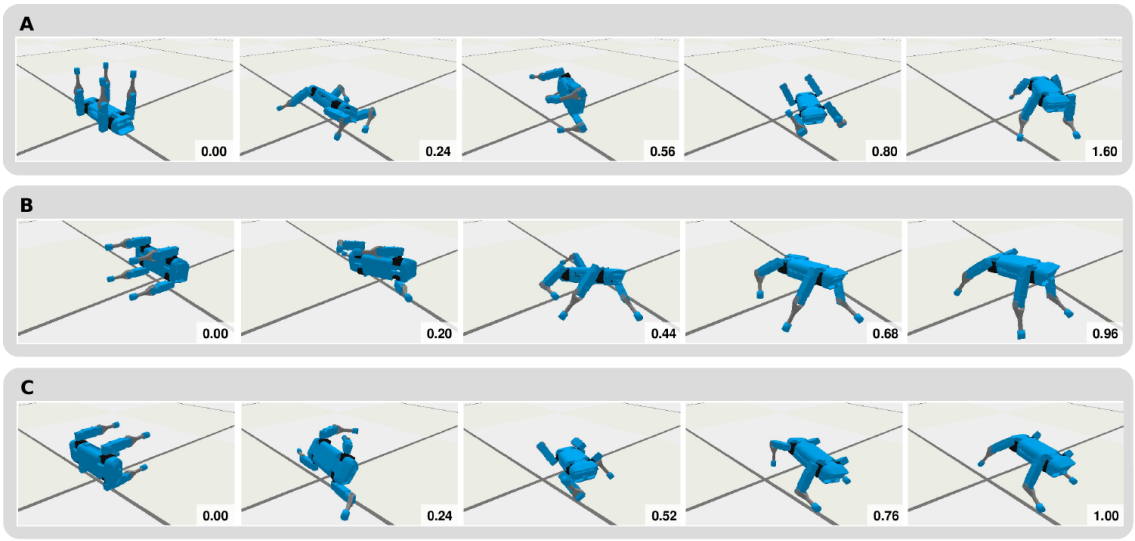


Figure 6.5: Snapshots of Spotmicro performing fall recovery maneuvers in simulation. (A) Supine. (B) Left lateral. (C) Right lateral.

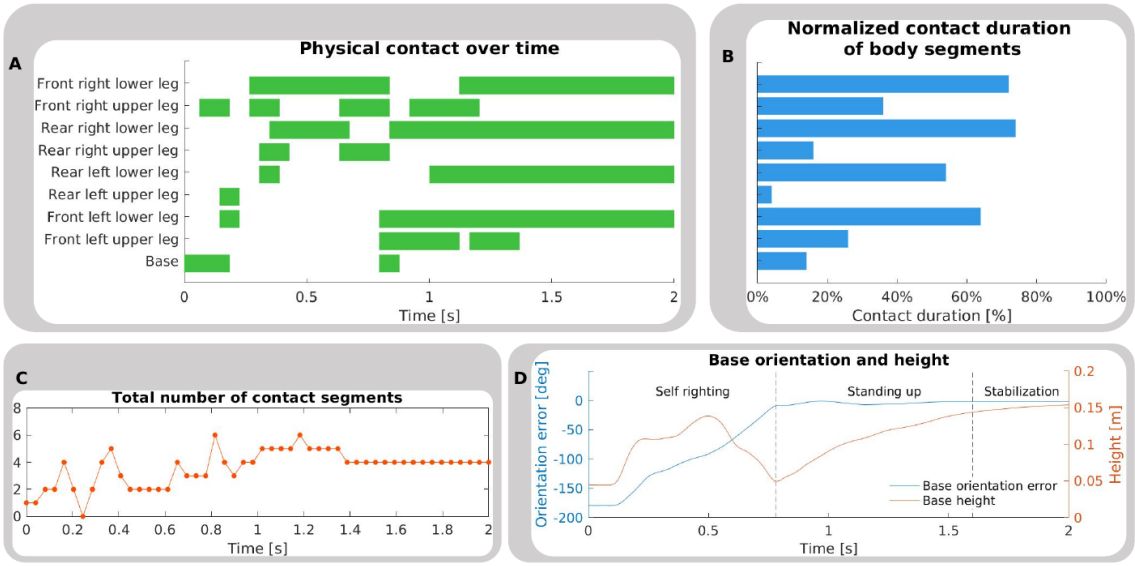


Figure 6.6: Detailed analysis of the contact status of the body segments of Spotmicro during fall recovery. (A) Contact status of body segments over time. (B) Normalized contact duratioin of body segments. (C) Total number of body segments in contact during each timestep. (D) Orientation error and Height of robot base. The orientation error is defined by the angle between the local z axis of the robot base frame and the global z axis of the world frame which points towards the opposite side of the gravity vector. When the robot stand in its nominal posture, the local z axis is aligned with the global z axis and thus the orientation error is 0.

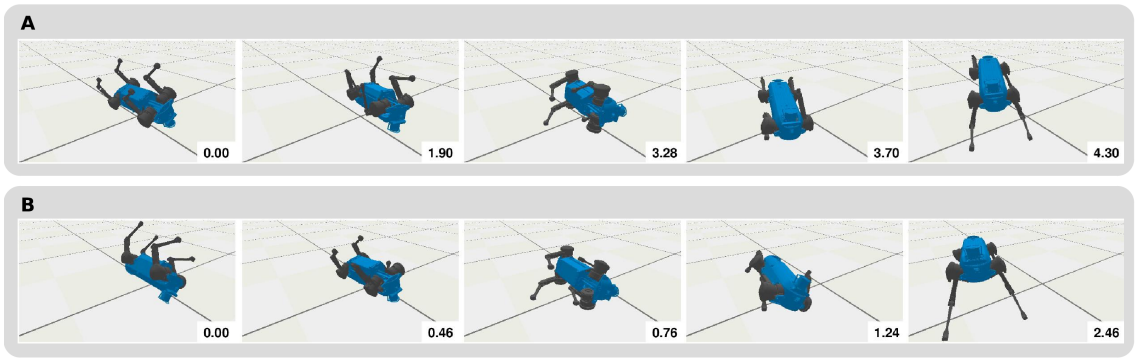


Figure 6.7: Snapshots of Jueying Pro performing fall recovery maneuvers in simulation. Due to the rounded curvature of the Lidar sensor unit at its front, the Jueying Pro is unable to lie on its back and will roll over to its side. Therefore, only the left lateral (A) and right lateral (B) postures are shown.

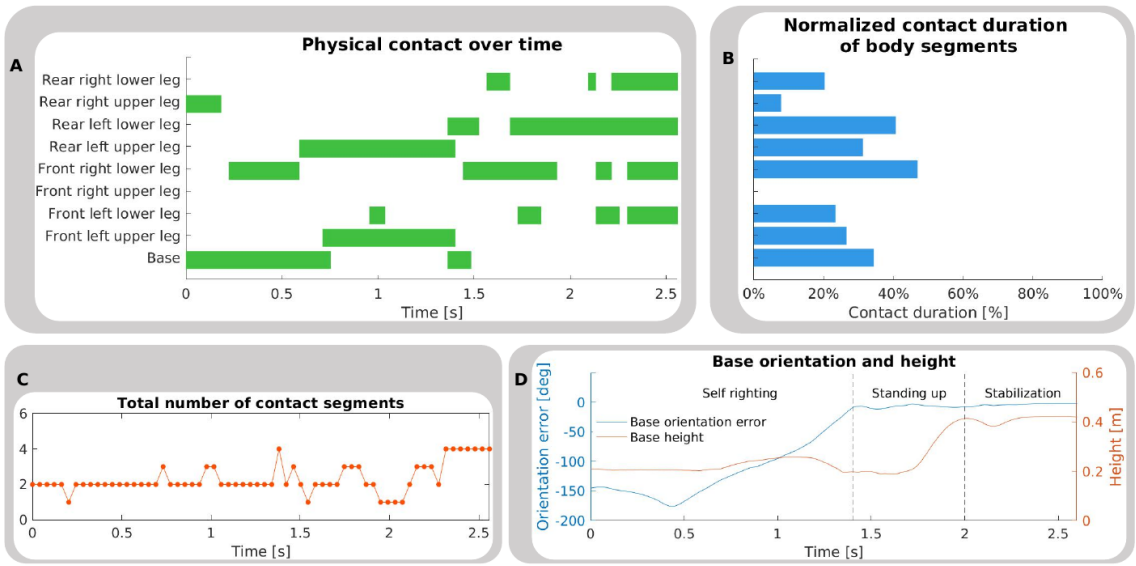


Figure 6.8: Detailed analysis of the contact status of the body segments of Jueying Pro during fall recovery. (A) Contact status of body segments over time. (B) Normalized contact duratioin of body segments. (C) Total number of body segments in contact during each timestep. (D) Orientation error and Height of robot base.

Spotmicro is able to perform swift recovery maneuvers in under 2 seconds, while it takes the Jueying Pro around $5 \sim 7$ seconds to complete a recovery maneuver. This difference is due to their differences in mass, i.e., Spotmicro weights 2.3kg and Jueying Pro weighs 70kg. Its low inertia allows Spotmicro to gain the momentum needed to right itself very quickly.

From Fig. 6.6 and Fig. 6.8 we can observe that fall recovery can be roughly divided into three phases: (i) Self righting, (ii) standing up, and (iii) stabilization. In the first phase, the robot reorients itself to minimize the error between the nominal standing posture. In the second phase, the robot pushes against the ground and increases its height. In the final phase, the robot performs minor adjustments to the body posture and stabilizes itself.

6.4.2 Fall Recovery on Humanoids

The principle for fall recovery for humanoids is the same as that for quadrupeds. The proposed learning framework is not limited to quadruped robots and can be used to learn fall recovery policies for different humanoid robots with different morphologies, demonstrating the versatility and generalizability of our framework. Compared to quadruped robots, humanoid robots have higher centre of mass height and smaller support polygon, which makes fall recovery behaviours for humanoids more challenging.

We have successfully trained a policy to perform fall recovery motions on the Valkyrie robot and Sigmaban robot, as shown in Fig. 6.9 and 6.11. The fall recovery policy is able to recover from supine, prone, and left and right lateral postures. From Fig. 6.10 and Fig. 6.12 we can observe that humanoid fall recovery has a three phase process similar to that of quadrupeds: (i) Self righting, (ii) standing up, and (iii) stabilization.

Despite the differences in the morphologies of the Valkyrie and Sigmaban robots, when recovering from supine and lateral postures, the Sigmaban and Valkyrie robots both learn a policy that involves rolling and adjusting into a prone posture with the abdomen facing down that provides enough clearance for the arms before executing the standing-up motion using all four limbs. The motion of using the arms to support the upper body while standing up is human-like and natural.

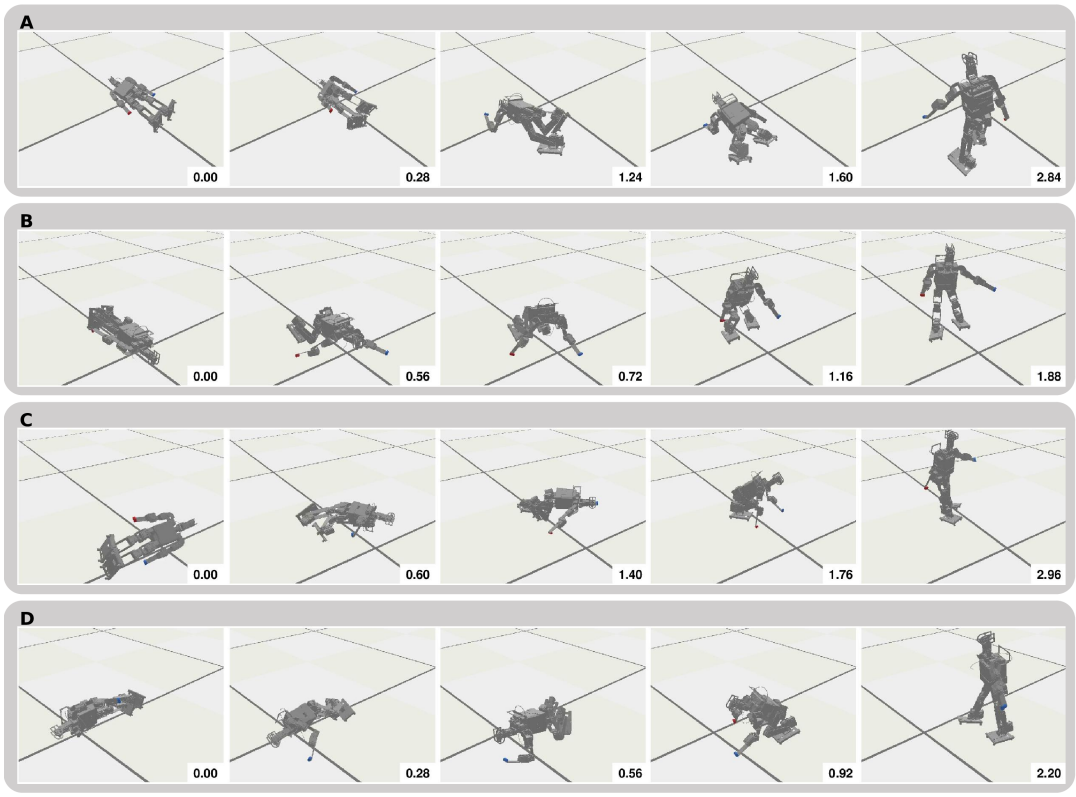


Figure 6.9: Snapshots of Sigmaban robot performing fall recovery.

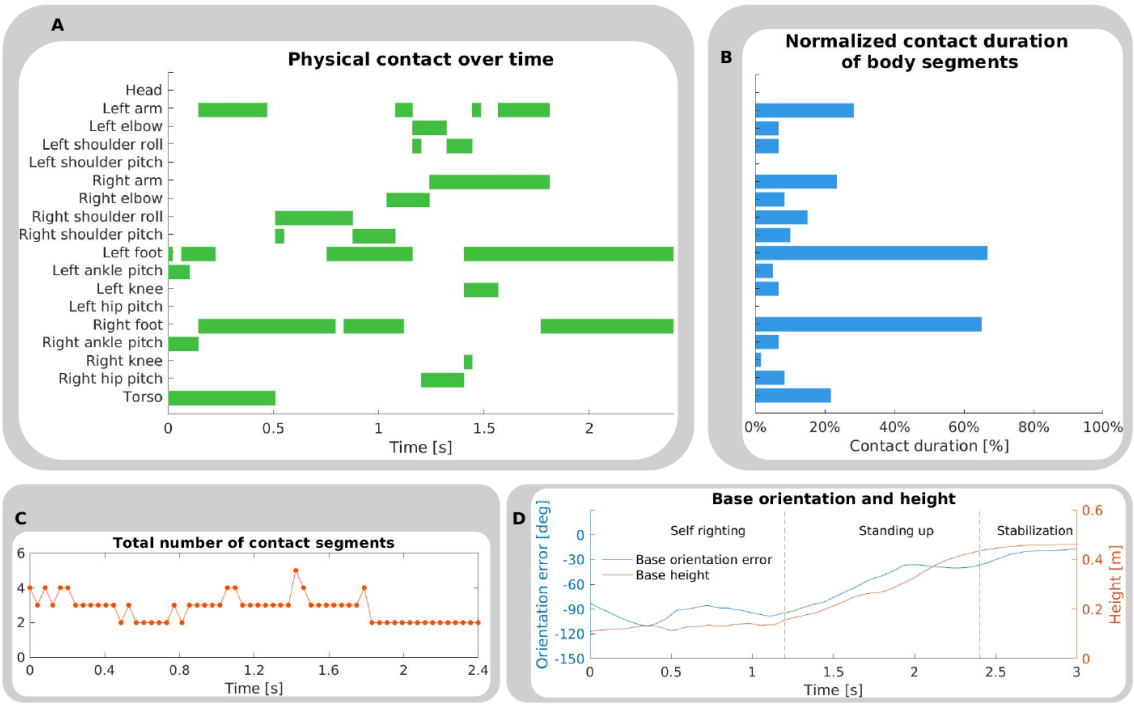


Figure 6.10: Detailed analysis of the contact status of the body segments of Sigmaban during fall recovery. (A) Contact status of body segments over time. (B) Normalized contact duratioin of body segments. (C) Total number of body segments in contact during each timestep. (D) Orientation error and Height of robot base.

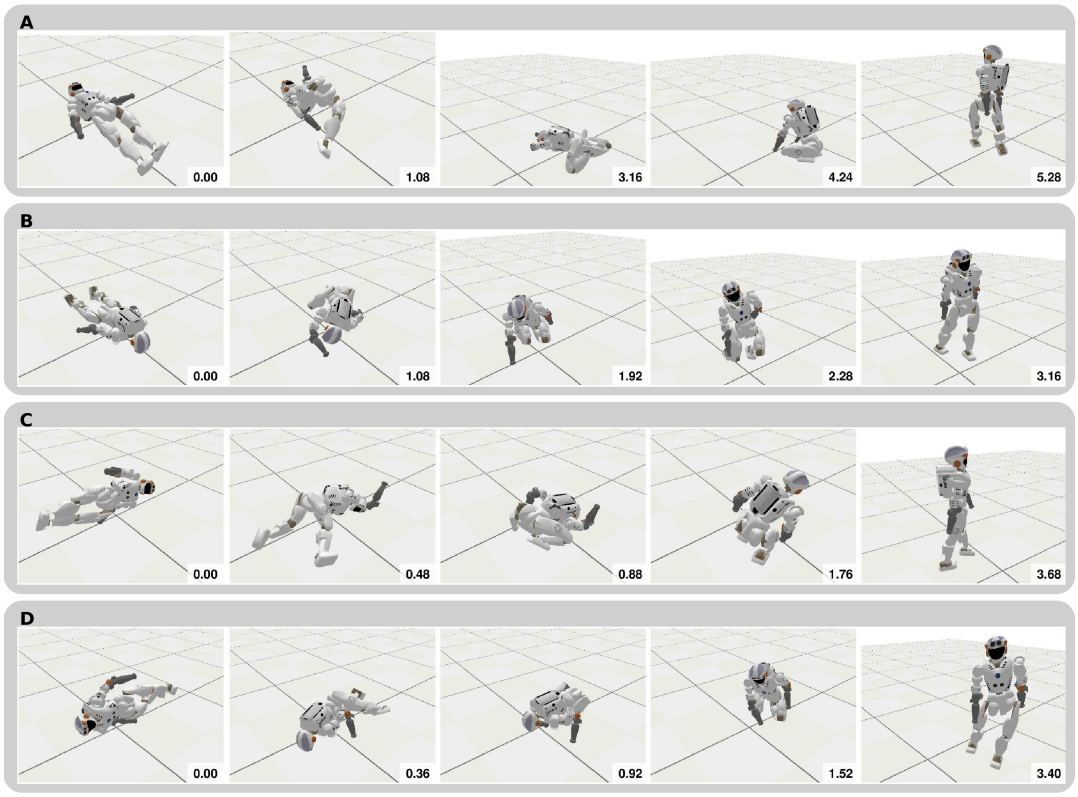


Figure 6.11: Snapshots of Valkyrie robot performing fall recovery.

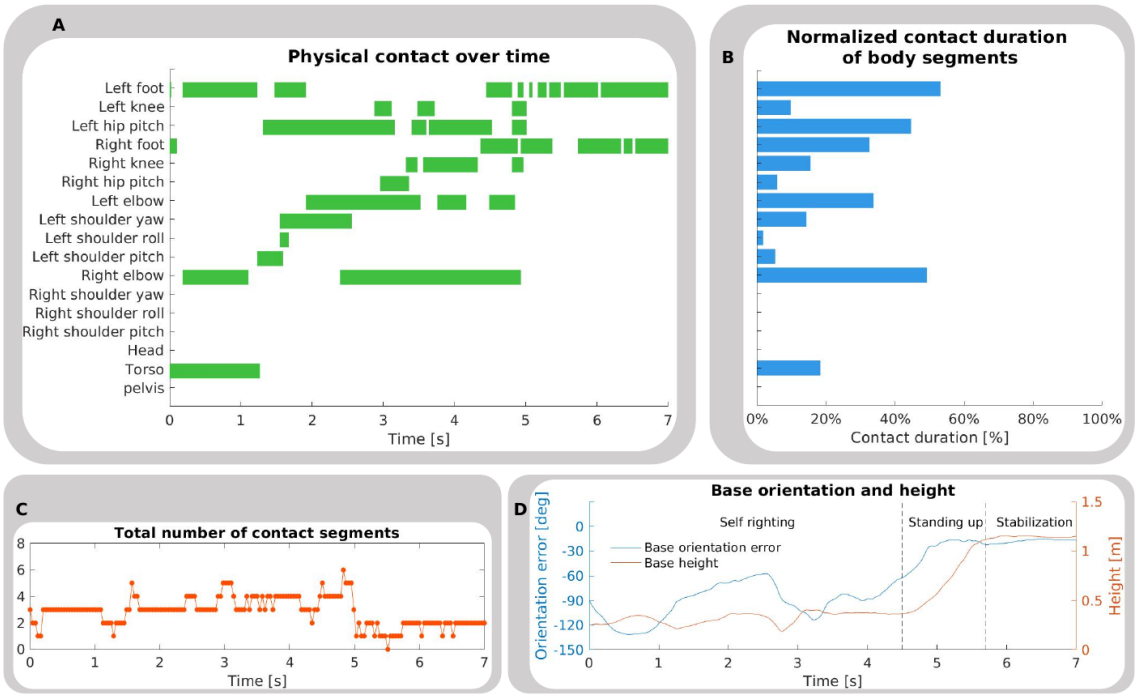


Figure 6.12: Detailed analysis of the contact status of the body segments of Valkyrie during fall recovery. (A) Contact status of body segments over time. (B) Normalized contact duratioin of body segments. (C) Total number of body segments in contact during each timestep. (D) Orientation error and Height of robot base.

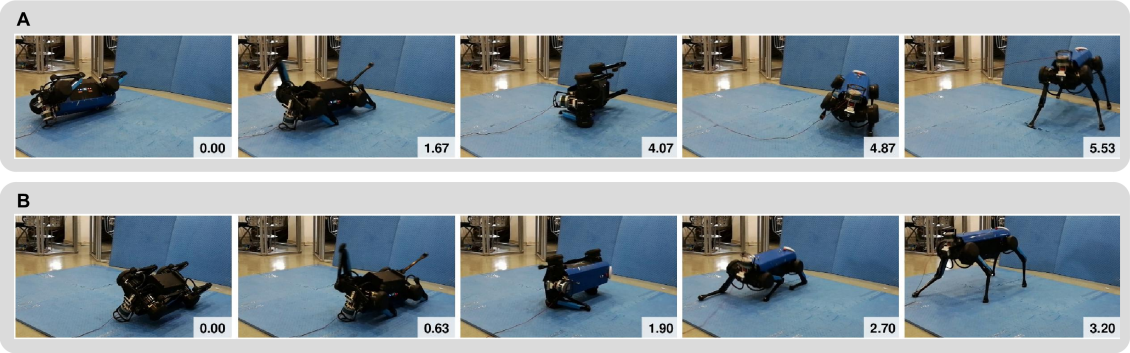


Figure 6.13: Snapshots of real-world experiments showing the Jueying Pro robot performing fall recovery.

6.4.3 Real-World Implementation

The fall recovery policy for the Jueying Pro learned in simulation can be deployed directly on the real robot, as shown in Fig. 6.13. The policy is transferred to the real robot without additional measures being used to bridge the simulation and reality gap, indicating that the policy is robust enough to be able to generalize to the discrepancy between the simulation and real world. Other reasons for the successful implementation without sim-to-real transfer is that the robot hardware is fairly robust with accurate sensor readings and good actuators that matches the model in simulation close enough. This might not apply to other robots, further research has to be done to investigate in the situation sim-to-real transfer where sim-to-real is not necessary.

6.5 Conclusion

Previous studies have demonstrated that fall recovery motions involved in standing up in humanoid and quadruped robots can be achieved using deterministic and analytical engineering approaches. In this chapter, we have shown that it is possible to produce similar or better fall recovery behaviour using DRL.

In this chapter we propose a DRL framework which is able to learn a versatile fall recovery policy for humanoid robots. The obtained fall recovery policy looks natural and human-like. The results demonstrated the feasibility and realizability of using DRL to learn human-like standing-up behaviours for humanoid robots. Furthermore, the proposed learning framework can generalize over different robot morphologies, as shown by the successful fall recovery policies for four different robot models; two humanoids and two quadrupeds.

The current limitation of the proposed method is that it has only been trained and tested on a flat surface with high stiffness and friction, and might fail in scenarios including steep inclinations, rough terrain, or low friction. To overcome these limitations, we need to randomize

the physical settings of the environment in simulation to allow the agent to learn a policy that is generalizable to a broader range of extreme terrain cases.

Chapter 7

Multi-Expert Learning of Adaptive Locomotion Behaviours

In Chapters 3,4, and 5, we used deep reinforcement learning to achieve balancing and walking for bipedal robots. In Chapter 6, we obtained fall recovery policies for both humanoid robots and quadruped robots in simulation and validated the policy on a real quadruped. Since we have successfully obtained a wide range of different motor skills, including balancing, walking, and fall recovery, we have now gathered the necessary components for the design of a unified learning framework capable of fusing multiple skills. In this chapter, we transferred the learning framework designed in previous chapters to obtain locomotion and fall recovery policies for quadruped robots. Moreover, we extended the multi-expert framework described in Chapter 4 and used it as a basis to design a multi-skill control policy that combines various trotting, steering, and fall recovery skills into a unified neural network control policy.

Achieving versatile robot locomotion requires adaptive motor skills in various new scenarios. We proposed a Multi-Expert Learning Architecture (MELA) that learns to synthesise adaptive skills from a group of representative expert skills. MELA first learns the core skills by separate deep neural networks (DNNs) for distinct tasks, and then refines all DNNs to acquire more diversified skills across various locomotion modes, including all the dynamic transitions in between. During runtime, MELA constantly blends multiple DNNs and dynamically fuses a new synthesised DNN to generate adaptive behaviours in response to changing situations. This approach leverages the advantages of the trained expert skills and the fast online synthesis of adaptive policies to achieve responsive reactions to unexpected perturbations. Using one unified MELA framework, we demonstrated successful multi-skill locomotion on a quadruped robot that performed coherent trotting, steering, and fall recovery autonomously, and showed the merit of multi-expert learning that generated behaviours adaptive to new scenarios.

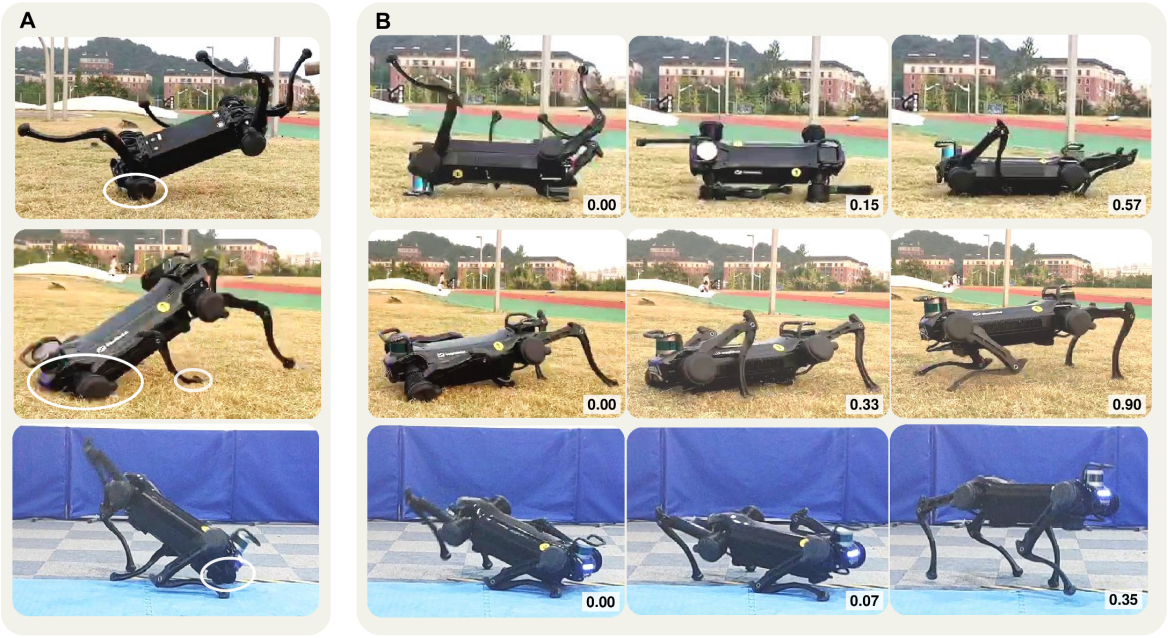


Figure 7.1: Challenging locomotion scenarios and agile manoeuvres of a quadruped robot. (A) Three challenging scenarios of the Jueying robot during various tests: unexpected body contacts with the environment and unpredictable robot states. The white circled regions highlight unusual contact that can occur at any body areas. (B) Different adaptive behaviours from our proposed learning framework that generated dynamic motions and complex coordinations of legs for immediate recovery from failures. (Time in snapshots is in second).

7.1 Introduction

Adaptive motor skills enable living organisms to accomplish novel motor tasks and offer them better chances to survive in nature. Among vast sensorimotor skills, locomotion is essential for most animals to move in the environment. Therefore, to understand and create adaptive locomotion behaviours is a long-standing scientific theme for biologists and roboticists. From a neurological perspective, it is worth understanding how sensorimotor control processes various sensory information and produces adaptive reactions in new situations that were not seen before [107], [108], [109], [110]. From a robotics perspective, it is interesting to take a bio-inspired approach and transfer biological principles, such as primitive neural circuits, to produce robot behaviours similar to that of animals [107]. Since the underlying mechanisms of the motor cortex are not yet fully replicated [110], [111], we take the latter approach by drawing inspirations from biological motor control to develop learning algorithms to achieve skill adaptation for robot locomotion [112].

This study is to investigate how an artificial agent can learn to generate multiple motor skills from a set of existing skills, particularly for critical locomotion tasks that require immediate responses. As an example of learning a complex physical task, playing soccer consists of a

number of sub-skills, such as dribbling, passing, and shooting. During training, players first practise the most important sub-skills separately. Once mastered, all different sub-skills are used in a flexible combination to improve all these techniques coherently. Our research of multi-skill learning has studied such skill-adaptive capabilities, and we also draw inspirations from animal motor control for designing the learning and control architecture. Using a quadruped robot as the testbed, we aim to produce versatile skills and adaptive behaviours to succeed in unexpected situations and unseen scenarios in a responsive manner.

The DARPA Robotics Challenge (DRC) from 2012 to 2015 fostered the development of semi-autonomous robots for dangerous missions: disaster response in unstructured environments [113]. Most DRC robots had different forms of legged design for the dexterity to traverse irregular surfaces and recover from falls. Despite the tremendous engineering efforts, no robot could recover from falls autonomously [114]. To date, most legged robots still lack such an autonomous ability to generate adaptive actions to deal with unexpected situations.

Due to the uncertainties in unseen situations, locomotion failures are likely to happen and are devastating for real robots, resulting in hardware damage. We illustrate representative challenges in robot locomotion from real field tests (see Fig. 7.1A). Typically, falling occurs within a second and the time window for fall prevention is about 0.2 - 0.5 second. Therefore, it is critical to react immediately to coordinate different locomotion modes, mitigate perturbations, and prevent or recover from failures. In comparison to man-made robots, biological systems, e.g., cats, dogs, and humans, exhibit higher versatility [115], and the performance gap lies in the motion intelligence to handle changing and complex situations [115], [116], [117].

Our research studies a machine learning approach that learns reactive locomotion skills and generates adaptive behaviours by reusing and recombining trained skills. Here, we investigate the motor skills in the form of feedback control policies to address the reactive adaptation to multimodalities during robot locomotion, leading to increased robustness against failures.

7.1.1 Related Work

Unspecified body contacts due to uncertain interactions with the environment impose major challenges in finding control solutions. The main approach in the legged locomotion community uses model-based mathematical optimisation to solve these multi-contact problems, such as Model-Predictive Control (MPC), whole-body Quadratic Programming (QP), and Trajectory Optimisation (TO). To achieve fast online computation, MPC utilises simplified models and short predictive horizons to plan task-space motions for walking [118], running [119] and pacing [120]. QP methods are used for whole-body control to map task-space motions (e.g., those from the MPC) to joint-space actions, considering the whole robot model and physical constraints [121], [2]. A unified but computationally expensive technique is to optimise all models and constraints together (whole robot model, contact model, environment constraints), i.e., through nonlinear MPC (NMPC) [122] and whole-body optimisation [123], [124], [125].

In the optimisation scheme, all physical contacts between the robot and the environment need to be defined as constraints in the formulation. The contact sequence, such as the contact location, timing, and duration, needs to be specified either by manual design or by an additional planner [126], [127]. Furthermore, the explicit properties of the robot and the environment need to be modelled [128], but are expensive to compute [129] and thus difficult to run in real-time in complex settings even with exhaustive computing [130]. This fundamental principle suffers from the curse of dimensionality, and therefore limits the scalability to real-time solutions in more complex and challenging problems [131].

To this end, Deep Reinforcement Learning (DRL) becomes attractive for acquiring task-level skills: through rewarding intended outcomes and penalising undesired ones, an artificial agent can learn desirable behaviours [126], [127]. Using DRL offers several advantages: the training process can be realised by using physics engines to perform a large number of iterations in simulations without risks of hardware damage; the agent can explore freely and learn effective policies that are difficult for humans to manually design; and the computation of readily trained neural networks can be real-time. For legged locomotion, many DRL results have been achieved in simulation [6], [8] and on hardware from recent studies [132], [133], [134], e.g., demonstration of learning-based control on a real robot using separate policies for fall recovery and walking [54]. However, similar to other DRL approaches, the learning policies in [54] were specialised in separate tasks, instead of being a unified policy across different tasks. This is a common feature due to the learning structure that only trains a single DRL agent for solving one specific task, which results in a narrowly skilled policy.

Hierarchical Reinforcement Learning (HRL) solves complex tasks at different levels of temporal abstraction using the existing knowledge in experts [135]: experts are trained to encode low-level motor primitives, while a high-level policy selects the appropriate expert [136], [36]. However, generating new skills cannot be achieved in the standard HRL framework since only one expert is selected at a time. This problem can be addressed by learning to continuously blend high-dimensional latent variables of all experts [137]. One related approach is the Mixture of Experts (MoE) that synthesises the outputs of individual experts specialised on sub-problems using a gating function [138], which has been used in robotics [139], computer vision [140], and computer graphics [141]. However, compared to blending the experts' high-dimensional latent variables, MoE has known limitations in scaling to high degree-of-freedom systems [141], because the limited expressivity of the low-dimensional latent space causes expert imbalance problems, i.e., favouring certain experts while degrading others [90].

We draw inspirations from biological motor control to design our control and learning framework. Biological studies suggest that motor behaviours are controlled by the Central Nervous System (CNS) that resets the reference position of body segments, and the difference between the reference and the actual position excites the muscular activities for generating appropriate forces [142]. This precludes the need of computing the inverse dynamics, simplifies

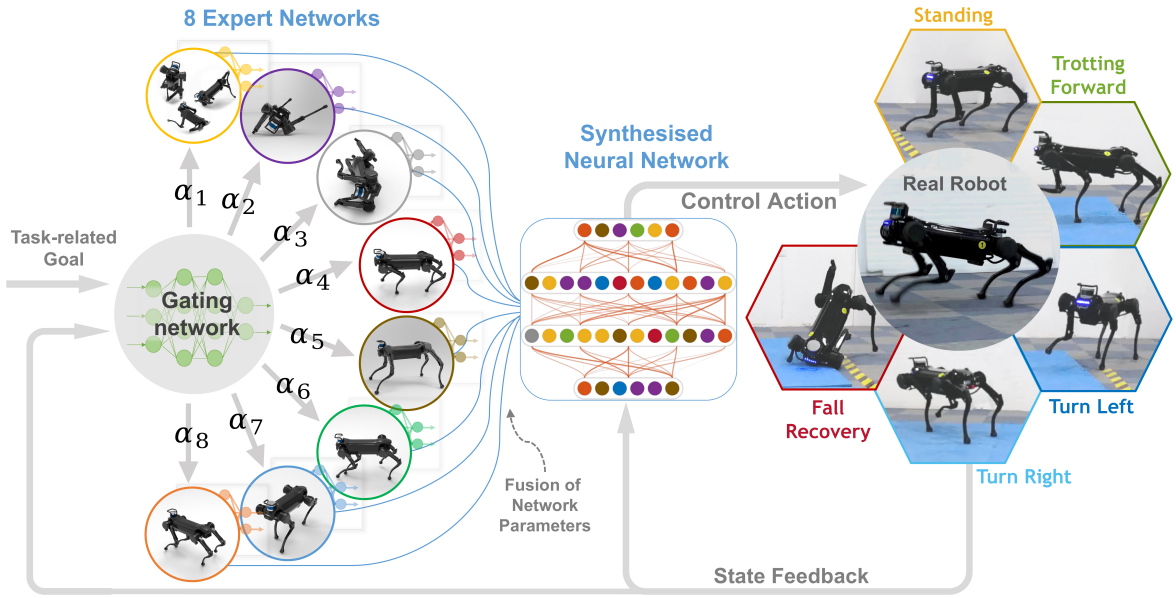


Figure 7.2: Multi-Expert Learning Architecture (MELA): a hierarchical deep reinforcement learning framework that synthesises multiple deep neural networks (DNNs) together to produce versatile locomotive skills. The Gating Neural Network (GNN) generates variable weights (α) to fuse the parameters of all eight expert networks (each expert is illustrated by its primary motor skill), such that newly synthesised motor skills are adapted to different locomotion modes by blending useful learned behaviours collectively from the collection of experts.

control and minimises the computation [143]. Since the spring-damper property provided by the impedance control resembles the elasticity of biological muscles, we applied the hypothesis of Equilibrium-Point (EP) control to generate joint torques by offsetting the equilibrium point.

Inspired by the biomechanical control of muscular systems and the EP-hypothesis, we distribute the robot motor control in two layers: (i) at the bottom layer, we use torque control to configure the joint impedance for the robot; and (ii) at the top layer, we designate deep neural networks (DNNs) to produce set-points for all joints to modulate posture and joint torques, establishing suitable force interactions with the environment. By doing so, we can focus on developing the learning algorithms at the top layer to achieve motor intelligence.

7.1.2 Contribution

This work aims to achieve a breadth of adaptive behaviours for contact-rich locomotion and present research of hierarchical motor control using deep reinforcement learning. To this end, we propose a Multi-Expert Learning Architecture (MELA), which contains multiple expert neural networks (each with unique motor skills) and a gating neural network that fuses expert networks dynamically into a more versatile and adaptive neural network (see Fig. 7.2).

Compared to the approach of using kinematic primitives [144], [145], the proposed MELA policy indirectly modulates the joint torques by changing the references of joint angles, where

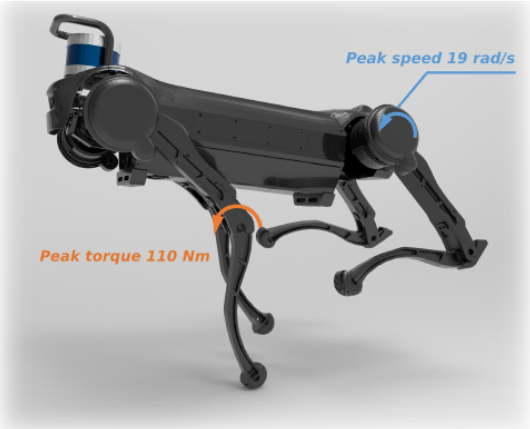
the resulting motions are the natural outcomes of the dynamic interactions with the environment. In contrast to other hierarchical learning approaches that select one policy representing one skill at a time [136], [36], MELA continuously combines network parameters of all experts seamlessly, and therefore is responsive without wait time from disjoined switching. Additionally, since MELA synthesises the experts in a high-dimensional feature space, i.e., weighted average of the network parameters (weights and biases of the neural networks), and therefore, it does not suffer from the expert imbalance problem as in MoE [90]. Similar multi-expert structures that blend experts in high-dimensional feature space were studied in computer graphics for kinematic animation [90], [89], [146], but have not yet been developed as feedback policies for the control of dynamical systems such as robots.

Our presented work contributes to a learning framework that (i) effectively generates multiple distinctive skills, (ii) diversifies expert skills through co-training, and (iii) synthesises multi-skill policies with adaptive behaviours in new situations, which are supported by the proof-of-concept experiments on a real robot with validations of adaptive behaviours, as well as various extreme test scenarios in the physics simulation. By synthesising multiple expert skills, the collective expertise of MELA is more versatile compared to that of each single expert, thanks to the dynamic structure of integrating all experts based on the online state feedback. Such multi-expert learning allows each expert to specialise in unique locomotive skills, i.e., some prioritise postural control and failure recovery while others acquire strategies for maximising task performance. As a result, MELA can perform a broad range of adaptive motor skills in a holistic manner, and is more versatile because of the diversification among experts.

The proposed framework is effective to achieve reactive and adaptive motor behaviours to changing situations consisting of unseen scenarios, unexpected disturbances, and different locomotion modes, which have not been addressed well by a unified framework in the literature. The results in Fig. 7.1B shed light on the advantages of the proposed work that accomplish a variety of strategies for the task success, for example, quick responses to counterbalance perturbations at different unexpected postures and the ability to produce stable dynamical transitions. This is an indicative level of machine intelligence – the capability of autonomous locomotive skills that do require significant intelligence to design if these need to be programmed by humans. In the following sections, we report technical details of our learning framework and the results of adaptive behaviours and robust locomotion.

7.2 Methodology

In this section, we will first introduce the robot platform and the control framework, then explain the core designs of the learning framework, including reward terms, state observations, and action space .etc. Particularly, we will present an emulation of the frequency response of



	Joint range (°)	τ (Nm) [*Peak*]	ω (rad/s) [*Peak*]
Hip roll	-25 – 25	25 [*75*]	13 [*19*]
Hip pitch	-159 – 29	25 [*75*]	13 [*19*]
Knee	37 – 164	45 [*110*]	13 [*19*]
Body size (LWH)		0.85m×0.30m×0.30m	
Leg length (upper, lower)		0.33m, 0.34m	

Figure 7.3: Specification of the Jueying quadruped robot

actuators and a loss function design for producing smooth and feasible actions. Finally, we elaborate on the MELA framework and the 2-stage MELA training procedure.

7.2.1 Robot Platform

We implemented our learning algorithms on the *Jueying* quadruped robot [147](52) to validate the adaptive behaviours with feasible and safe tests on the real hardware. *Jueying* has 3 degrees of freedom (DoF) per leg (12 DoFs in total) which is actuated by brushless electric motors with low gear ratio (i.e., 7:1) and high-fidelity joint torque control (see specifications in Fig. 7.3).

7.2.2 Control Framework

Joint torque control of the real *Jueying* robot is used to create an impedance mode for all joints because having mechanical impedance is known to be robust during the physical contact and interactions [148], [149]. Hence, the synthesised DNN plays a role similar to a CNS that produces trajectory attractors constantly pulling and pushing all joints in the impedance mode to generate joint torques similar to a spring-damper system.

The neural network policy generates joint position references at 25 Hz (see Fig. 7.7A). A policy with an update frequency of around 25 Hz is a common setting as a high-level motion planning layer in robot control [6], [8], [134], [30], [37]. The standard joint-level trajectory interpolation and speed limit were implemented to generate smooth position references at 1000 Hz for the low-level impedance control.

The Proportional-Derivative (PD) gains used for the impedance control are shown in Table 7.1. The feedback gains were 700Nm/rad and 10Nms/rad for stiffness and damping respectively and were sufficient for the robot to have good control over its swing leg and placement of stance

Table 7.1: Proportional-Derivative parameters for joint-level PD controller.

	Hip roll	Hip pitch	Knee pitch
$K_p(Nm/rad)$	700	700	700
$K_d(Nms/rad)$	10	10	10

foot. Based on our proposed smoothing loss, the policy has learned an active compliance behaviour in the simulation as well as in the experiments by adjusting the position references.

During the dynamic response, the stiffness produced by the active control can be lower than the original PD gains set in the impedance mode, as it is regulated the same way as for series elastic actuators (SEA) [150]. Therefore, it is desirable to have medium PD gains in the low-level joint control, and render the desired or a lower impedance by adjusting the high-level position set-point, i.e., a deliberate motion to buffer an impact.

The MELA policy learns how to regulate the set-point for the impedance controller to achieve compliant behaviour, which is similar to active compliance found in the control of robotic arms. Robot manipulators are usually controlled by high PD gains and are thus very stiff, but soft and compliant behaviours can still be achieved by limiting the amount of torque applied on joints [151], [152]. Since the neural network receives feedback of the actual joint positions q^m and has direct control of the desired joint positions q^d, according to the SEA principle $\tau = K_p(q^d - q^m)$, actively changing the set-point q^d with respect to the measured position q^m can restrict the amount of torque, lower the stiffness, and thus increase the compliance.

7.2.3 Soft Actor Critic

For our experiment, the computation bottle neck is the time needed for simulation to obtain the samples. We chose to use Soft Actor Critic (SAC) an off-policy DRL algorithm due to its sample efficiency [17]. SAC optimizes over an expected sum of rewards augmented with an additional maximum entropy objective as:

$$J_{SAC}(\pi) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}(r(s_t,a_t)) + \alpha H(\pi(\cdot|s_t)), \tag{7.1}$$

where $\sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}(r(s_t,a_t))$ is the expected sum of rewards, $H(\pi(\cdot|s_t))$ is the expected entropy of policy over the sample distribution. The temperature parameter affects the stochasticity and thus exploration capability of the optimal policy by changing the importance of the entropy term, where a higher leads to more stochastic policies and vice versa. SAC balances the well-known problem of exploitation and exploration by automatically tuning the temperature parameter α.

The policy is expressed as a Gaussian distribution $\mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)^2)$ with mean $\mu_\theta(s_t)$ and covariance $\sigma_\theta(s_t)$ generated by a neural network. A tanh-squashing function is applied to the Gaussian samples to project the action distribution to be within $\hat{a}_{s_t} \in (-1,1)$. The squashed action $\hat{a}_{s_t}$ is then scaled by the joint range $\bar{q}$ and formulate the target action $a_{s_t} \in \bar{q}$ as joint

Table 7.2: Hyperparameters for SAC algorithm.

SAC hyperparameters	
Smoothing loss coefficient	2.0
Learning rate	3e-4
Weight decay	1e-6
Discount factor	0.987
Polyak averaging weight	0.001
Replay buffer size	1e6
Steps per epoch	5e3

Table 7.3: This Table describes the basic definitions of mathematical notations to help explain the equations of the reward terms in Table 7.4.

Nomenclature	
φ^{base}	Orientation vector: the projection of the gravity vector in the robot base frame to represent the orientation
h_{world}	The robot base height (z) in the world frame
v_{base}^{world}	The linear velocity of the robot base in the world frame
v_{base}^{local}	The linear velocity of the robot base in the robot's local heading frame: $R^T(\theta_{yaw}^{world}) \times v_{base}^{world}$
θ_{yaw}^{world}	The yaw orientation of the robot body in the world frame
ω_{yaw}^{world}	The yaw angular velocity of the robot body in the world frame
τ	The vector of all joint torques
q	The vector of all joint angle
$\dot{q}$	The vector of all joint velocity
$(\dot{\cdot})$	The desired quantity of selected property $(\cdot)$, where $(\cdot)$ serves as a placeholder
$h_{foot,n}^{world}$	The n-th foot height (z) in the world frame
$v_{foot,n}^{world}$	The n-th foot horizontal linear velocity $(\dot{x}, \dot{y})$ in the world frame
$p_{foot,n}^{world}$	The n-th foot horizontal placement in the world frame
p_{goal}^{world}	The horizontal component of the goal position (x, y) in the world frame
p_{robot}^{world}	The horizontal component of the robot base position in the world frame
$u_{goal,base}^{base}$	The unit vector pointing from the robot base to goal in the base frame

angle references for the robot. The hyperparameters of the SAC algorithm can be found in Table 7.2.

7.2.4 Reward Design

For training individual tasks, such as fall recovery, trotting and target-following, we designed a specific reward function with corresponding weights for reward terms that represent different physical quantities. The full list of reward terms is: (i) base pose, (ii) base height, (iii) base velocity, (iv) joint torque regularisation, (v) joint velocity regularisation, (vi) body ground contact, (vii) foot ground contact, (viiii) yaw velocity, (ix) swing and stance, (x) average foot placement, (xi) reference joint position, (xii) reference foot contact, (xiii) robot's heading to the goal, and

(xiv) the goal position. Different tasks require a specific subset of reward components, i.e., fall recovery requires reward (i) to (vii), locomotion requires reward (i) to (xii), and multimodal target-following requires all 14 reward terms. In summary, the first 7 reward terms are common physical quantities across all tasks to ensure stable robot motion, while the other terms are task-specific. The mathematical formulation of all reward terms and the task-specific weights are in Table 7.4 and Table 7.5, respectively.

We used a Radial Basis Function (RBF) to design the bounded reward function:

$$K(x, \hat{x}, \alpha) = e^{\alpha(\hat{x}-x)^2}, \tag{7.2}$$

where x is the physical quantity for the evaluation, $\hat{x}$ is the desired value, and is the parameter that controls the width of the RBF. All reward terms composed of continuous physical properties utilize this RBF. The exceptions are the three reward terms of discrete properties, i.e. foot-ground contact, body-ground contact and ground contact from the imitation gait. Details of the individual reward terms using the nomenclature Table 7.3 are presented in Table 7.4.

The reward design follows a similar design rule as in [29], [30], and [37]. In Table 7.4, the first twelve terms are straightforward and task-related since each evaluates a physical quantity directly related to the physical movements. The *reference joint position reward* and *reference foot contact reward* provide reference trajectories from an existing trotting gait for the agent to imitate. By providing such a reference gait, the agent is able to learn stable trotting more efficiently and effectively through imitation. We also clarify the purpose of the last two remaining reward terms in Table 7.4 as follows. The *Swing and stance reward* reflects the contact constraint that encourages a higher velocity at a smaller height error to encourage the swing motion, and a lower velocity at a larger height error while the feet deviate from the nominal height $\hat{h}$. By discouraging the stance foot to move, the Swing and stance reward is able to prevent slippage as well. The *foot placement reward* encourages the feet to place around the robot body averagely, guiding the policy to perform more symmetric and stable foot placement during locomotion.

Table 7.4: Detailed description of task reward terms. The terms are combined to construct the task reward.

Task reward terms	
Base pose	$K(\varphi,[0,0,-1],\alpha), \quad \alpha=-2.35$
Base height	$K(h,\hat{h},\alpha), \quad \alpha=-51.16$
Base velocity	$K(v_{base}^{world},\hat{v}_{base}^{world},\alpha), \quad \alpha=-18.42$
Joint torque regularization	$K(\tau,0,\alpha), \quad \alpha=-0.003$
Joint velocity regularization	$K(\dot{q},0,\alpha_y), \quad \alpha=-0.026$
Foot ground contact	$\begin{cases}0, & \text{foot not in contact with ground}\\ 1, & \text{foot in contact with ground.}\end{cases}$
Body ground contact	$\begin{cases}0, & \text{main body in contact with ground}\\ 1, & \end{cases}$
Yaw velocity	$K(\omega,0,\alpha), \quad \alpha=-7.47$
Reference joint position	$K(q,\hat{q},\alpha), \hat{q}$ is the joint reference, $\quad \alpha=-29.88$
Reference foot contact	$\begin{cases}0, & \\ 1, & \text{match desired foot contact}\end{cases}$
Robot heading	$K(u_{goal,base}^{base},[1,0,0],\alpha), \quad \alpha=-2.35$
Goal position	$K(p_{goal}^{world},p_{base}^{world},\alpha), \quad \alpha=-0.74$
Foot clearance	$K(1/4\sum_4^{n=1}(h_{foot,n}^{world}-\hat{h}_{foot,n}^{world}),0,\alpha), \quad \alpha=-460.50$
Average foot placement	$K(1/4\sum_4^{n=1}(p_{foot,n}^{world}),p_{base}^{world},\alpha), \quad \alpha=-18.42$

Table 7.5: Weights of the reward terms for different tasks. Trotting and fall recovery used a subset of the reward terms, and the multimodal MELA locomotion used all the reward terms.

Weights of reward terms			
Physical quantities	Trotting	Fall Recovery	MELA
Base pose	0.071	0.333	0.100
Base height	0.036	0.333	0.100
Base velocity	0.178	0.067	0.071
Joint torque regularization	0.018	0.067	0.020
Joint velocity regularization	0.018	0.067	0.020
Foot ground contact	0.018	0.067	0.020
Body ground contact	0.018	0.067	0.020
Yaw velocity	0.071	0.000	0.020
Foot clearance	0.036	0.000	0.036
Reference joint position	0.416	0.000	0.167
Reference foot contact	0.083	0.000	0.033
Average foot placement	0.036	0.000	0.036
Robot heading	0.000	0.000	0.143
Goal position	0.000	0.000	0.214

Table 7.6: Selection of states for different tasks and neural networks.

Physical quantities	Locomotion	Fall recovery	MELA gating network	MELA expert network
Joint position	✓	✓	✗	✓
Gravity vector	✓	✓	✓	✓
Angular velocity of the robot	✓	✓	✓	✓
Linear velocity of the robot agnostic to the heading direction	✓	✗	✓	✓
Phase vector	✓	✗	✗	✓
Goal position	✗	✗	✓	✓

7.2.5 State Observation

We used the following state observations that are essential and minimalistic to train successful policies: (i) base (robot body) orientation, (ii) angular velocity of the robot base, (iii) linear velocity of the robot base, (iv) joint positions, (v) phase vector, and (vi) goal position. The body orientation is represented as a normalised gravity vector projected in the robot local frame using the measurements from the Inertial Measurement Unit (IMU). The angular velocities of the body and all the joint positions were measured by the IMU and joint encoders, respectively. The linear velocity was estimated from IMU as a strap-down inertial navigation system and then transformed to the heading coordinate, so the resulting velocity is agnostic to the heading direction. The 2D phase vector was designed to clock along the unit-circle to describe the phase of the periodic trotting (see Fig. 7.4). Lastly, the target position is represented by a relative 3D vector with respect to the robot's local frame, and only the horizontal components are used as the state inputs. The detailed combination of the state observations for different tasks and networks are in Table 7.6.

7.2.6 Action Space

The benchmark of DRL-based locomotion [54], [25] showed a suitable configuration for the action space that yields better performance and faster learning due to the compliant interaction: a DRL agent provides joint references and an impedance mode for controlling the joint. The related work using this setting produced successful motions [54], [25], and hence we adopted the same design of the action space here. To guarantee smooth and feasible actions for the real robot, we developed two important techniques: (i) emulation of the characteristics of the actuators' frequency response using low-pass filters, which enforces physically realisable reference motions, and (ii) design of a special loss function to generate smooth and non-jerky joint references and torques. We named these two techniques action filtering and smoothing loss, respectively.

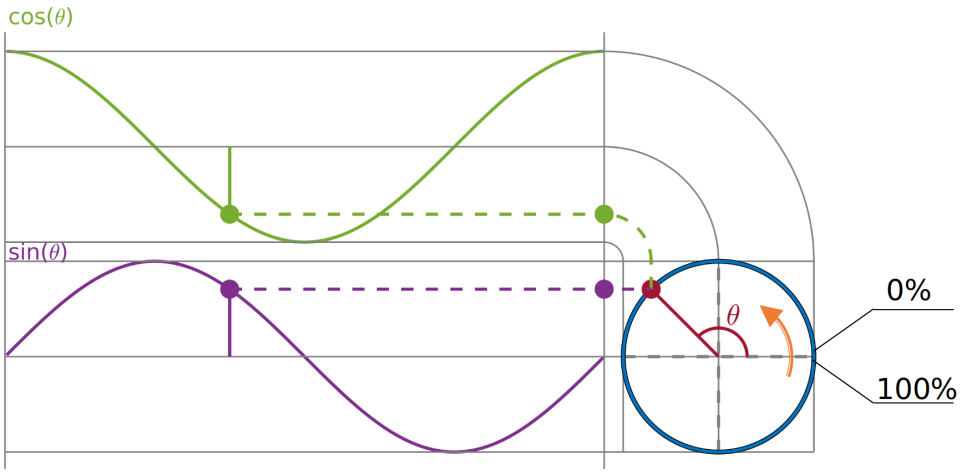


Figure 7.4: Illustration of the 2D phase vector for training the locomotion policy. The sine and cosine functions are used to represent the time-varying phase variable in a continuous manner, and the resulting phase vector contains temporal information to describe the phase (0-100%) of a periodic gait.

7.2.7 Action Filtering

Real actuators have limited control bandwidth and hence the references with frequencies higher than the bandwidth cannot be tracked. However, a common issue in simulation is that the learning policy takes advantage of the pure torque source with unlimited control bandwidth: exploitation of abrupt and jerky motions that are only possible in simulation to maximise the reward but infeasible on real systems. The difference between ideal actuators (pure torque source) in simulation and real actuators (restricted bandwidth, torque, speed and power) needs to be addressed appropriately in the learning framework. In addition to the basic position and velocity limit [105], we performed action filtering using a first-order Butterworth filter to emulate the frequency response of real motors and to guide the policy to learn a smoother and more feasible behaviour.

For the *Jueying* robot, we found that emulating the frequency response and setting the speed limit are sufficient to represent realistic characteristics of actuators. The properties of *Jueying*'s actuators, such as good torque tracking control and low gear ratio which results in decoupled inertia and minimal gear friction, avoid the need of modelling detailed actuator properties and simplify the simulation setting. During the simulation training, we emulated the limited frequency response of actuators by applying action filtering on the output action with the cut-off frequency of 5 Hz, which was higher than the 1.67 Hz trotting gait (see Fig. 7.21). This provides realistic restriction of high-frequency actions and prevents the policy from over-exploiting risky motions while still permitting necessary movements for dynamic tasks. For safety reasons in real experiments, we applied a more conservative cut-off frequency of 3 Hz in case of unexpected jerky references. As a result, we obtained all policies with smooth motions

within the bandwidth (see Fig. 7.21) that can be executed on the real robot directly and safely.

7.2.8 Smoothing Loss

The action filtering mentioned above alone may not always guarantee smooth or feasible motions, because it only limits the frequency of the DNN output but not the magnitude, so the learning process may still explore and exploit low-frequency but large-amplitude motions regardless. Therefore, we further designed the smoothing loss based on the principle of minimal interaction to minimize the applied torques [142].

Biological studies show that when the CNS resets a new equilibrium, the displacement between the equilibrium and the actual position will activate a neuromuscular response that tries to reduce the muscular activity (torque). This principle of minimal interaction serves as a biological foundation of studying the proposed smoothing loss, which is effective for smoothing the exerted torque, i.e., $\tau = K_p(q^d - q^m)$. To guide the policy and generate actions following the minimal interaction principle, we designed a smoothing loss function $J_{smoothing}$ as:

$$J_{smoothing}(\mu(s_t)) = \left\| \mu(s_t) - q \right\|, \tag{7.3}$$

where $\mu(s_t)$ are the deterministic mean outputs of the stochastic policy used as the target joint references, and q are the measured joint positions. The smoothing loss $J_{smoothing}$ is the objective function that minimises the differences between the the target $\mu(s_t)$ and the current measurement q. As the joint references are the inputs for impedance control, this minimisation leads to more gentle torque profiles, thus encouraging the learning of strategies with the least effort as possible.

The proposed smoothing loss $J_{smoothing}$ is incorporated into the SAC training loss $J_{SAC}(\pi)$ and is used for the backpropagation of neural networks, instead of being part of the reward function. Adding the smoothing loss term to $J_{SAC}(\pi)$ allows the information (the causality of actions) to backpropagate directly through the neural network and to bypass the process of reward bootstrap and Q-function approximation. Since for training the policy, the Q-function requires iterations to obtain a valid and accurate enough approximation of the expected return, our approach of bypassing the Q-function avoids the wait time and permits the information to backpropagate within the first few iterations.

7.2.9 Sample Collection Procedure

The distribution of samples collected by the agent during training will affect the learning outcome of the policy. In order to obtain a sample distribution containing a variety of robot states for better generalisation, we used two techniques to augment the standard sample collection: reference state initialisation and early termination.

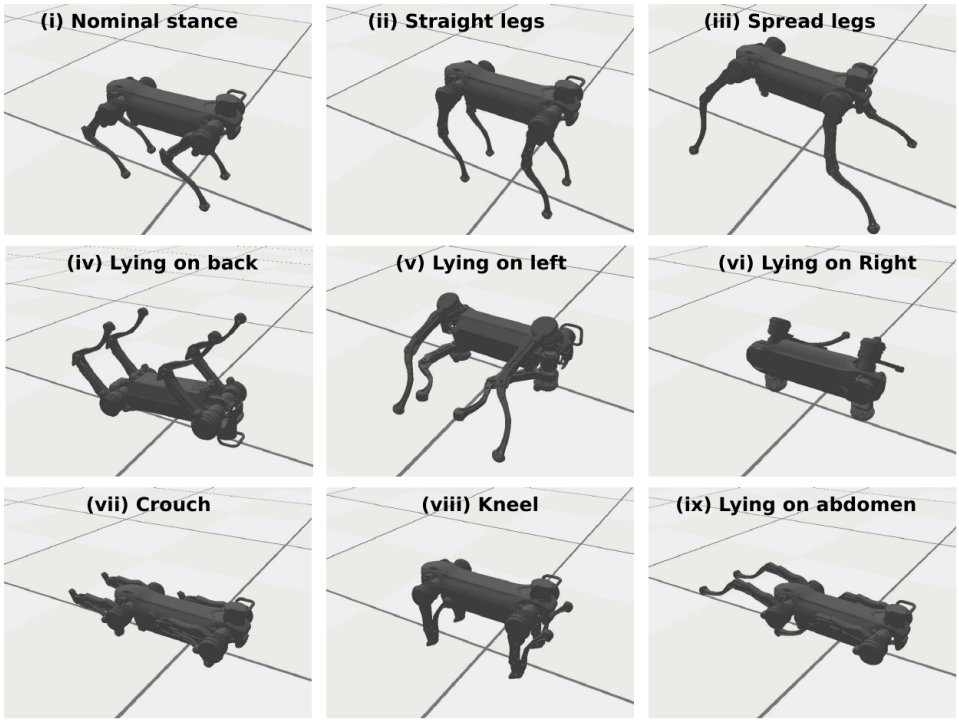


Figure 7.5: Nine distinct configurations used as the initialisation for training fall recovery policies in simulation. Snapshots are taken from the physics-based simulator using the PyBullet engine [1]

First, to increase the diversity among collected samples, we initialised each simulation episode by a random selection from a set of reference configurations. The diversity in the sample distribution allows the agent to learn a policy of good generalisability for a range of different states. Furthermore, by initialising the robot in difficult configurations, the agent can experience challenging cases more often. Second, we applied early termination to the episode during training when the robot encountered an undesirable state, such as a failure state in which the robot was unable to recover. Early termination prevents irreversible failures from skewing the sample distribution.

1. *Reference State Initialization*

   We designed a set of initial reference states to initialise episodes for each task. As shown in Fig. 7.5, for fall recovery, we designed 9 distinct poses to ensure the diversity within the collected samples: (i) standing at nominal height, (ii) standing at maximal height with straight knees, (iii) leg sprawling posture, (iv) lying on the back, (v) lying on the left side, (vi) lying on the right side, (vii) crouching, (viii) kneeling, and (ix) lying on the abdomen. Posture (i) - (vi) are common configurations during standing and fall recovery, while posture (vii) - (ix) are unusual contact configurations that are difficult to recover. For learning to trot, a trotting gait sample from the robot's factory setting was used as

reference states for imitation. We combined the reference state initialisation datasets of fall recovery and trotting to create a new set of initialisation states for the multimodal locomotion, since the target-following using MELA would involve all modes and their transitions. For training the MELA policy, the goal position was initialised within a circular area of 6 m radius around the robot at the beginning of each episode (see Fig. 7.6).

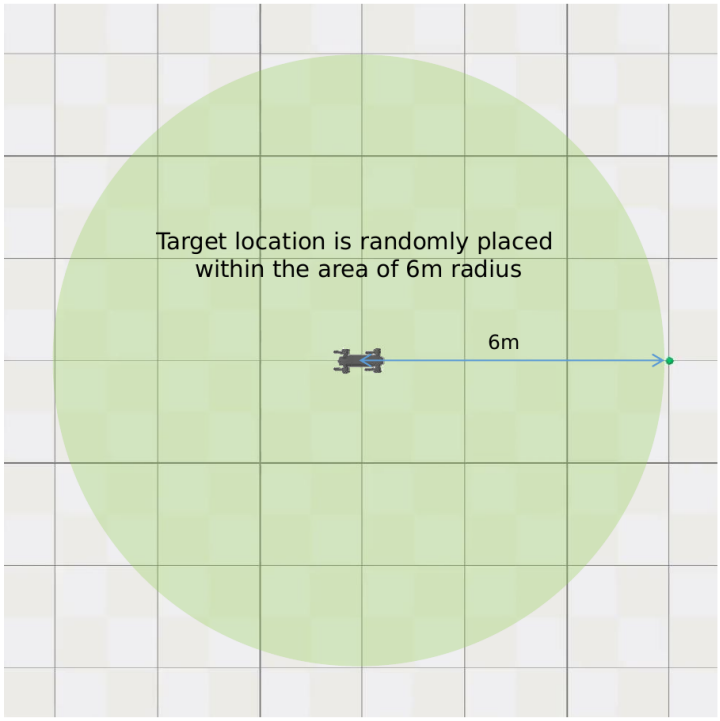


Figure 7.6: Setting of the target location for training MELA policies in simulation. During the initialisation of each sample collection episode, the target location (the green ball) is randomly placed within the area of 6 m radius around the robot, and remains fixed within the same episode.

2. *Early Termination*

We specified three termination criteria for locomotion related tasks: (i) any body part other than feet are in contact with the ground, (ii) body orientation exceeding a threshold of 90° (The body orientation is represented as a normalised gravity vector projected in the robot local frame using the measurements from the Inertial Measurement Unit (IMU)), and (iii) reaching the time limit of the episode. For the fall recovery task, only criterion 3 was used because the robot had to undergo the states specified in criteria (i) and (ii) in order to learn how to recover from failures autonomously.

7.2.10 MELA Training Procedure

Figure 7.7 depicts both the network architecture and training procedure of our MELA framework. The MELA network consists of one gating network and eight expert networks (see Fig. 7.7B). The gating network has 2 hidden layers with 128 neurons each, using a ReLU activation function. All expert networks have 2 hidden layers with 256 neurons each and use a ReLU activation function, which has the same network structure as that of the pre-trained expert policy networks shown in Fig. 7.7A.

Here, we elaborate on MELA's two-stage training procedure. In stage 1, as shown in Fig. 7.7A, successful trotting and fall recovery policies were pre-trained to warm-start the expert networks in the second stage. In stage 2, MELA first initialised eight expert networks as two subgroups by copying the weights and bias from the pre-trained experts (see Fig. 7.7B) and randomly initialised the weights and bias of the gating network. Then MELA embedded all the experts together with the gating network, and co-trained all of them with diverse samples. During the stage 2 co-training, the gating network needed to learn how to compute correct weights for all experts and synthesise a new skill-adaptive network, as illustrated in Fig. 7.7B. The robot feedback states were the input of the synthesised network for generating motor actions.

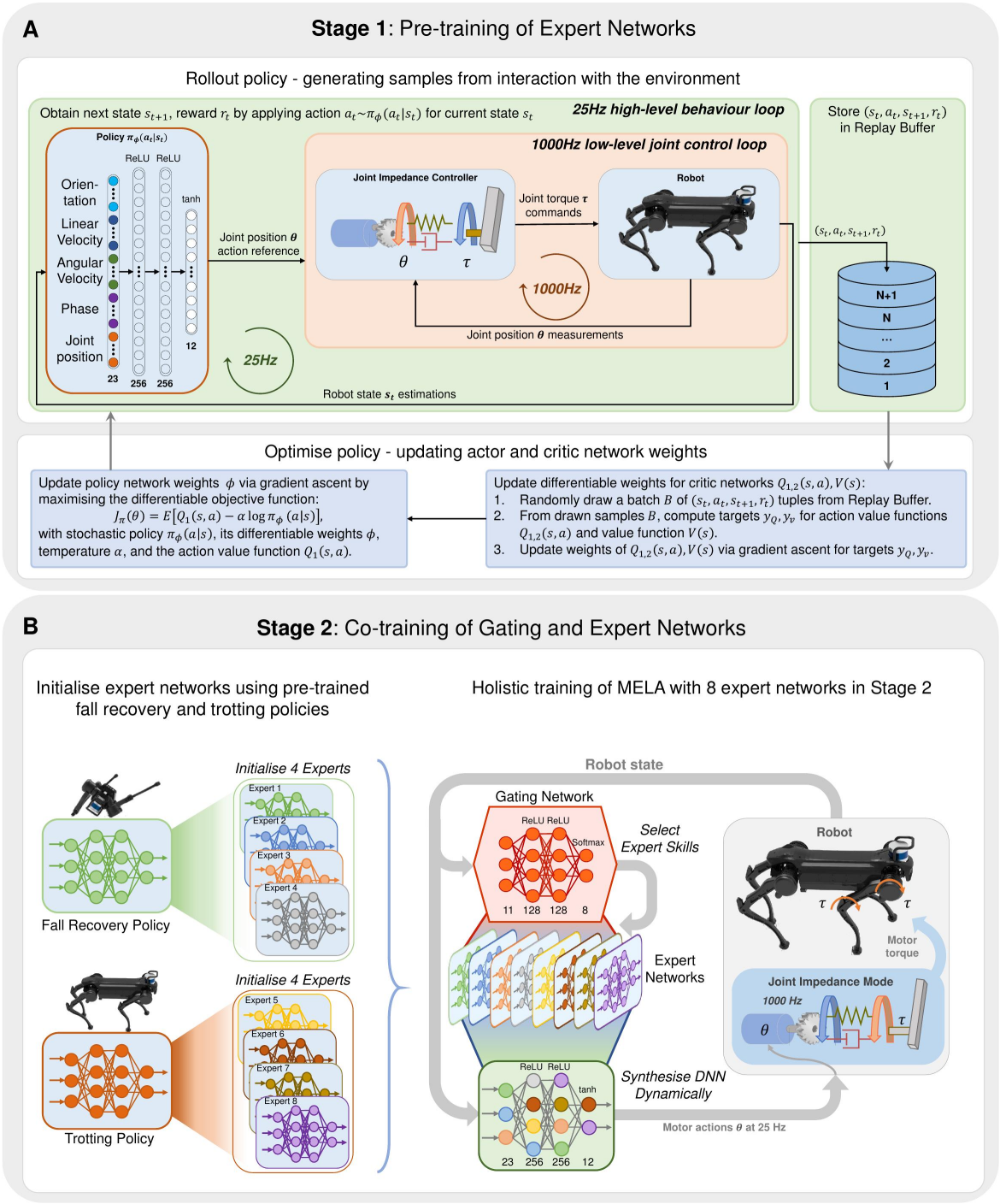


Figure 7.7: Two-stage training of MELA. (A) In stage 1, the fall recovery and trotting policies are individually trained. (B) In stage 2, the pre-trained trotting and fall recovery policies from stage 1 are used to initialise two evenly distributed groups of experts, each containing 4 experts. All these expert networks are co-trained together with the gating network.

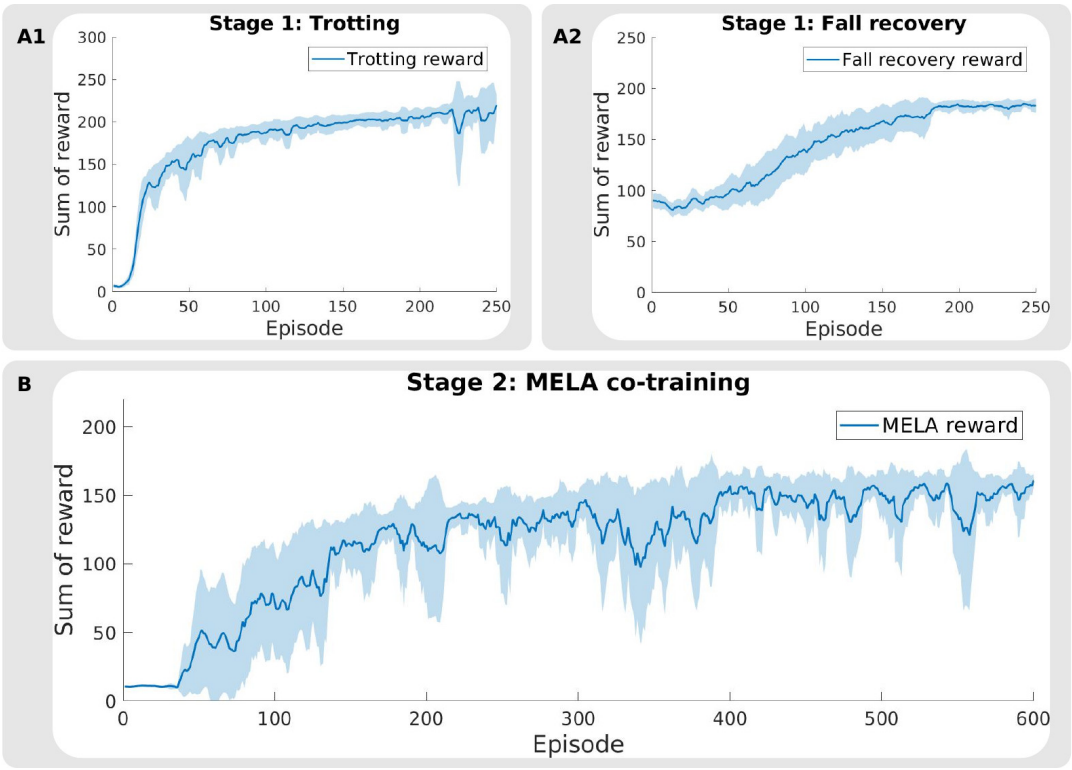


Figure 7.8: Learning curves during the first and second stage of MELA training. For training experts in the first stage of MELA, 250 episodes were required for both fall recovery and trotting tasks. For co-training in the second stage, 400 episodes were required. One episode consists of 5000 samples that were collected at 25 Hz.

Following the framework in Fig. 7.7, both stages were trained using the SAC algorithm, and the samples were collected at 25 Hz frequency while the actions were executed through the impedance control at 1000 Hz frequency. Both training stages initialised the robot in diverse configurations during the simulation episodes so as to increase the diversity of the collected samples. The learning curves during stage1 and stage 2 of the MELA training can be found in Fig. 7.8.

All neuron connections within MELA are differentiable, including those between the gating network and expert networks. This allows every network weight and bias to update through backpropagation simultaneously. Thus, all the MELA networks can be trained with the same backpropagation techniques used for Fully Connected Neural Networks. The actor for SAC was encoded using a MELA network, while the critic consisting of two Q functions was encoded as Fully Connected Neural Networks which were adopted from Double Q-learning to prevent overestimation [153],[154].

Let x, y, h denote the dimensions of the input, output, and the hidden layer respectively; let W and B be the weights and bias of the network. The parameter-set of the skill-adaptive

network is:

$$\Psi_{synth} = \left\{W_0 \in \mathbb{R}^{x\times h}, W_1 \in \mathbb{R}^{h\times h}, W_2 \in \mathbb{R}^{h\times y}, B_0 \in \mathbb{R}^{h}, B_1 \in \mathbb{R}^{h}, B_2 \in \mathbb{R}^{y}, \right\} \tag{7.4}$$

and the parameter-set of each individual expert network is:

$$\Psi_{synth} = \left\{W_0^n \in \mathbb{R}^{x\times h}, W_1^n \in \mathbb{R}^{h\times h}, W_2^n \in \mathbb{R}^{h\times y}, B_0^n \in \mathbb{R}^{h}, B_1^n \in \mathbb{R}^{h}, B_2^n \in \mathbb{R}^{y}, \right\} \tag{7.5}$$

During runtime, the weights W and bias B of skill-adaptive network are fused by the weighted sum formulation as:

$$W_i = \sum_{n=1}^{8} \alpha_n W_i^n,\ B_i = \sum_{n=1}^{8} \alpha_n B_i^n \tag{7.6}$$

where $n = 1,...,8$ is the index of experts, $i = 1,2,3$ is the corresponding layer index, and $\alpha_n \in [0,1]$ are the variable weights generated by the gating network.

The fused W and biases B are used to construct the synthesised network dynamically during runtime using the equation as follows.

$$\Phi_{synth} = Tanh(W_2 ReLU(W_1 ReLU(W_0 X + B_0) + B_1) + B_2) \tag{7.7}$$

Where $X \in \mathbb{R}^x$ is the input parameter, and $Tanh(*)$ and $ReLU(*)$ are the nonlinear activation functions. The sum of the 8 variable weights $\alpha_n, (n = 1,...,8)$ is normalized to one using a Softmax function. There are several nonlinear features in the blending process: each expert DNN is a nonlinear control policy by nature, and each blending weight is produced by nonlinear rescaling of the output of the gating network with a Softmax function to normalise different values of the original sum. Therefore, the resulting synthesised expert is a highly nonlinear control policy – a nonlinear mapping between the feedback states and actions that is required to deal with challenging scenarios.

7.3 Results

7.3.1 Multi-Expert Learning Framework

We first define key terminology to explain better the concepts referred to in this article: motor skill or skill in short, expert, and locomotion mode. *Motor skill*: a feedback policy that generates coordinated actions to complete a specific type of tasks, which serves as a building block for composing more complex manoeuvres. *Expert*: a deep neural network with specialised motor skills. *Locomotion mode*: a type of movements occurring in quadrupedal locomotion, such as standing, fall recovery, turning on the spot, steering left/right, and trotting forward/backward.

To complete tasks in new scenarios, an artificial agent needs the ability to adapt appropriate skills during runtime, for which MELA is proposed: a hierarchical reinforcement learning structure constituting a collection of Deep Neural Networks (DNNs) and a Gating Neural Network (GNN). As shown in Fig. 7.2, the GNN continuously fuses expert DNNs into a

synthesised neural network at every time step by computing the weighted average of all experts' network parameters. The synthesised policy fully encodes the motor skills of the experts in a high-dimensional feature space. Through repurposing and synthesising existing skills, MELA acquires a wide array of adaptive behaviours and achieves versatile locomotion in new scenarios.

The multi-expert policy of MELA is trained in a two-step process. In the first stage, we train individual policies to accomplish representative and distinct tasks. Specifically, we use the trained experts, such as fall recovery and trotting experts, to initialise all experts by two sub-groups. In the second stage, we use a gating neural network as the merging mechanism to fuse all network-parameters and train all experts together, so that their collective specialisations can be fully utilised by the gating network. Meanwhile, the gating network is trained to learn the continuous and variable activation of different experts to generate optimal policies at each control loop. The MELA policy was trained in the physics simulation and evaluated on a real robot system.

For constructing MELA, the number of experts is determined by the relation between the desired locomotion tasks and the required skills. A particular motor skill corresponds to a distinct control strategy (e.g., to roll the body by pushing the ground), and thus one locomotion task would require a set of different skills. In our study, we focus on five locomotion tasks (Fig. 7.2): fall recovery, standing, turning left, turning right, and trotting. All these movements need a variety of motor skills for interacting with the environment.

According to the configurations that the robot would encounter during these five tasks, a basic number of experts can be determined. There are 7 *distinct situations* each requiring at least one motor skill, namely: (i) fall recovery from a supine pose (lying on the back), (ii) fall recovery from lateral decubitus poses (lying on the left/right side), (iii) balance control during stance, (iv) body postural control, (v) trotting forward, (vi) left steering, and (vii) right steering. Hence, a minimal number of 7 DNNs can be determined to represent these skills. We also introduced a redundant expert that can represent nonlinear features and additional skills which are difficult to anticipate. As a result, we constructed the MELA collection using 8 experts in total. In addition, our further comparison found that using more than 8 experts had no further performance but more training time (see Fig. 7.9). It shall be noted that the 7 situations are only used as a guideline to determine the number of experts at the initial design process, and does not need to match the numeration of MELA's learned motor skills in the latter section.

7.3.2 Learning Individual Motor Skills

In challenging physical tasks, it is difficult to directly train control policies where a variety of skills are needed, such as recover falling poses and resume walking gaits. Prior studies show that pre-learning skills allow the experts to learn task representations, otherwise the policies were not able to learn to solve more difficult composite tasks [137], [141]. Like training in sports, it is

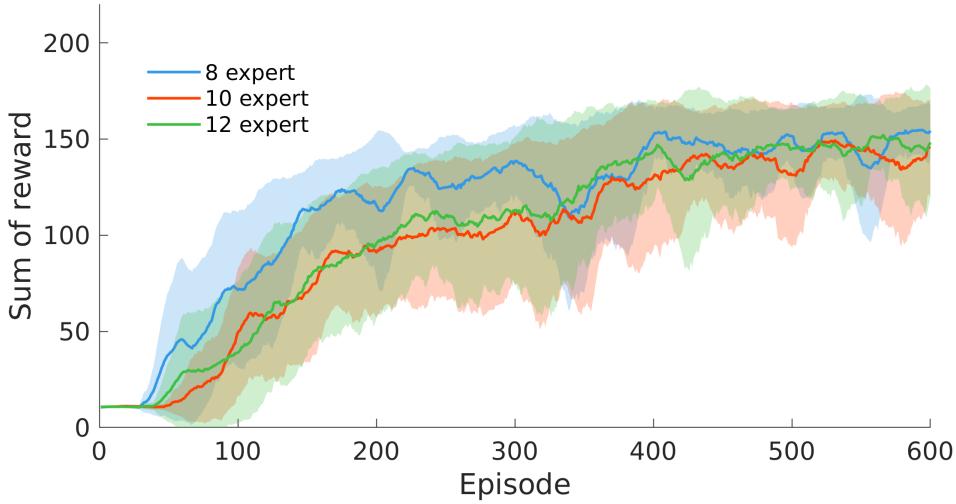


Figure 7.9: Comparison of MELA's learning curves using different numbers of expert networks. It can be seen that using more than 8 experts does not improve the task performance, and has a slower convergence instead.

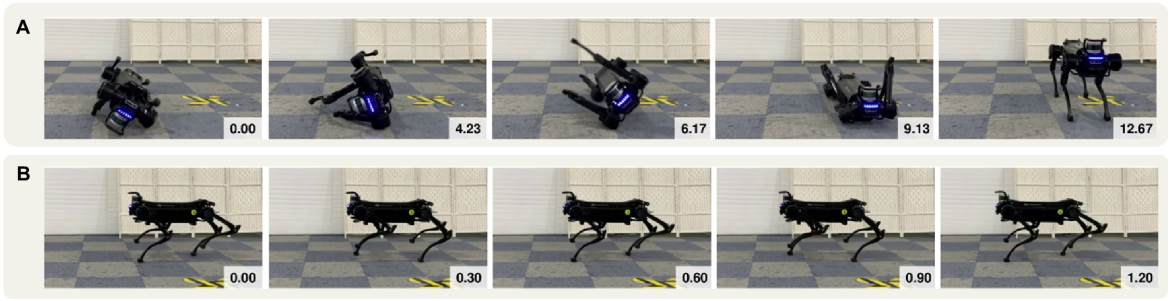


Figure 7.10: Baseline experiments of fall recovery and trotting from engineered controllers. (A) The fall recovery from the engineered controller has a fixed sequence of motions and takes more than 12 s to stand up. (B) The trotting gait sample from the robot's control suite.

essential to practise individual skills that are distinctly different. Similarly, MELA has a two-step training procedure of experts, in which two distinct, separate modes are specialised first: fall recovery and trotting. In the following, we will show the experimental results of fall recovery and trotting using individually trained neural networks, and then present the multimodal locomotion experiments using MELA.

7.3.2.1 Fall Recovery

For quadrupeds, a canonically stable configuration is a standing posture with four feet forming a support polygon close to the body's length and width. For fall recovery, the DRL agent is rewarded for feedback policies that restore such stable postures from various failures. We

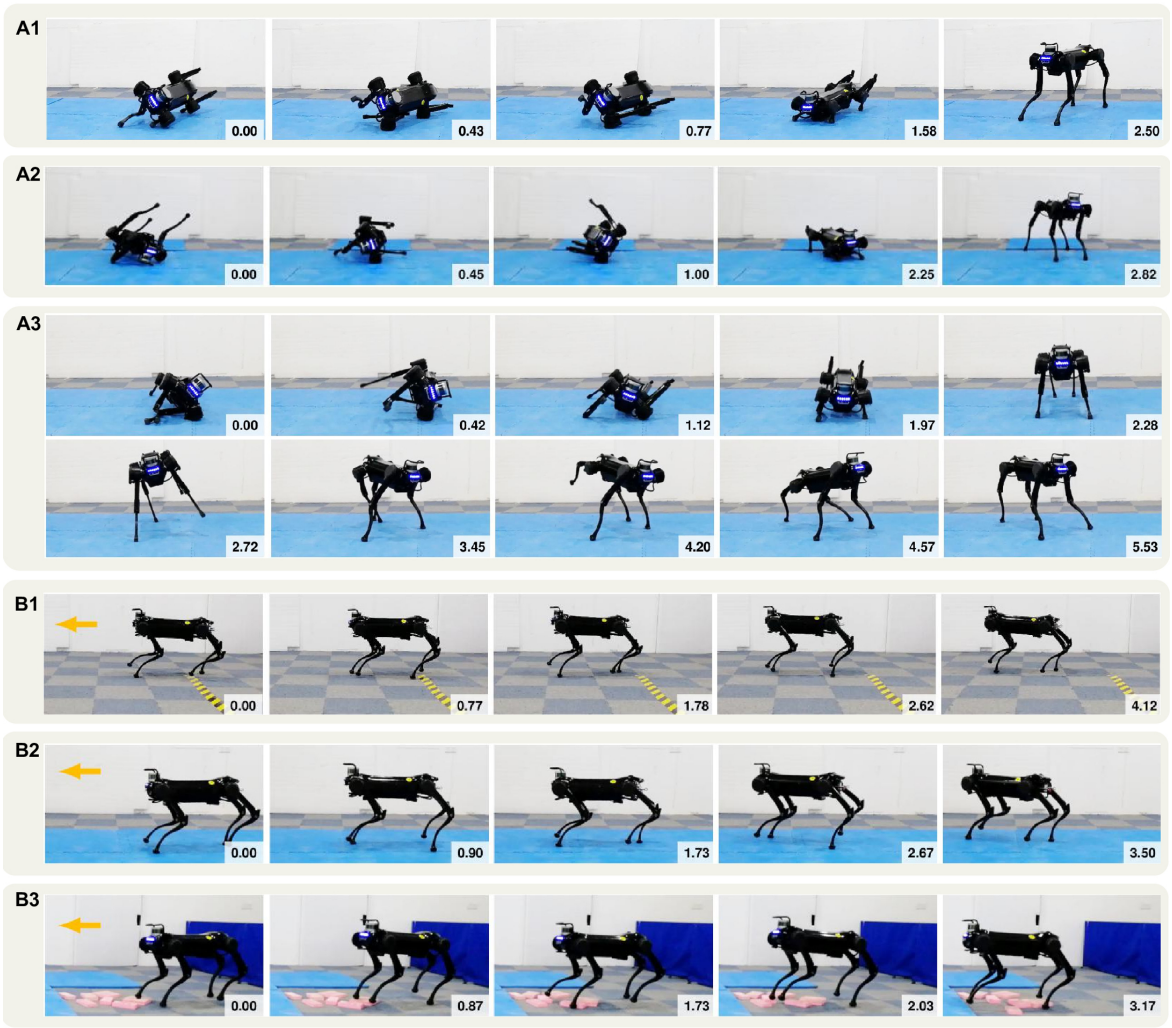


Figure 7.11: Individual motor skills for the fall recovery and trotting respectively. (A1) A configuration between prone and lateral decubitus positions where legs were stuck underneath the body: the robot first pushed the ground to lift up the body for ground clearance, and then retrieved legs to a prone posture for standing up. (A2) The robot actively used elbow-push to generate a large momentum to self-right to a prone position. (A3) A stepping behaviour was learned and performed naturally to keep balance. (B1) Stable trotting on a hard floor. (B2) Stable trotting on soft slippery foam mats. (B3) Stable trotting over scattered obstacles, showing the compliant interaction and robustness learned by the trotting expert. (Time in snapshots is in second).

applied random initial configurations to explore diverse robot states and facilitated the agent's ability to generalise policies for various fall poses (see Fig. 7.5).

We evaluated the robustness of recovery policies and Fig. 7.11A shows the learned reactive behaviours by four strategies: (i) natural rolling exploiting semi-passive movements, (ii) active righting and flipping, (iii) standing up from prone positions, and (iv) stepping. *Natural rolling*

describes the behaviour that the robot exploits the natural dynamics and gravity to roll over. This is activated when the robot is in a prone and/or lateral decubitus position, as shown in Fig. 7.11A1. *Active righting* is the strategy that the robot pushes by leg and elbow, creating momentum to actively flip itself to a prone position, as shown in Fig. 7.11A2. *Stepping* emerges when necessary during standing to regain balance, involving coordination and switching of support legs. An example of stepping is shown in Fig. 7.11A3, and such multi-contact switching was all generated naturally using the learned policy based on the online state feedback.

From all fall recovery experiments (see Fig. 7.11), we can see the responsive and versatile reactions, including the emerged stepping behaviour. Compared to the baseline fall recovery which is manually engineered with a fixed pattern (see Fig. 7.10), our learning policy is able to recover from various different fall scenarios, because it can respond to dynamic changes using online feedback, while the handcraft controller only addresses a narrow range of situations.

7.3.2.2 Trotting

Based on our proposed deep reinforcement learning framework, we applied imitation learning to train trotting skills using a reference trajectory (Fig. 7.10). By providing reference trajectories from an existing trotting gait to explore upon, the agent was able to learn stable trotting more efficiently and effectively through imitation.

Jueying can robustly trot under three ground conditions with different stiffness, friction, and obstacles (Fig. 7.11B). In Fig. 7.11B1, the concrete ground was covered by thin carpets with high friction, while in Fig. 7.11B2, 2cm thick foam mats were laid, creating a softer and more slippery surface. In Fig. 7.11B3, 5cm thick bricks were scattered as small obstacles. The learned motor skills were robust under different ground conditions, and the Jueying robot was able to continue trotting steadily in all three scenarios.

All these trained policies have exhibited behaviours of compliant interaction to handle physical interactions and impacts. The joint impedance mode offers the ability to indirectly regulate joint torques by the deviation between the desired joint position q^d and actual joint position q^m via the principle similar to the series elastic actuators, i.e., $\tau = K_p(q^d - q^m)$ [150]. The expert has indirectly learned active compliance control by regulating the references based on feedback of the current joint positions to minimise joint torques.

7.3.3 Multi-Expert Learning Structure

7.3.3.1 Analysis of learned MELA policy

After the first stage of the two-stage training process, the network parameters of fall recovery and trotting policies are transferred to the expert networks in MELA. In the second stage, all experts are co-trained with the gating network, and MELA repurposes the initial experts to learn adaptive behaviours necessary for multimodal locomotion, while the gating network learns

how to blend the acquired skills according to the changing tasks. As a result, MELA is able to achieve non-cyclic and asymmetric motions (fall recovery), rhythmic movements (trotting), goal-oriented tasks (target-following), as well as all dynamical transitions between different modes. These key adaptive behaviours of the MELA policy were realised on a real robot, which demonstrated the capabilities of achieving a diversity of locomotion tasks, adapting to external environmental changes responsively, and meanwhile following variable user commands.

To analyse the inner workings of MELA, we performed systematic tests of the trained MELA networks in the physics simulation, so as to fully stimulate a wide range of sensory inputs to maximum. Without hardware risks, we operated the robot in extremely dynamic motions to activate different experts, allowing the analysis of all distinct motor skills. This provides data of variable weights that reflects the activation level of all experts over each motor skill. The analysis in Fig. 7.12 was based on the simulation tests using the policy from a single training run, which was representative because the training process can consistently reproduce policies with very similar characteristics of skill specialisation. Fig. 7.12A shows the correlations and patterns between the activation of experts and the motor skills, and reveals that each motor skill has a dominant expert, suggesting the primary specialisation of each MELA expert.

As shown in Fig. 7.12A, 8 *fundamental motor skills* are acquired: (i) back righting, e.g., push elbow to roll over from supine positions (lying on the back); (ii) lateral rolling, e.g., retrieve legs and roll the body to a prone position (lying on the abdomen); (iii) postural control, e.g., maintain a nominal body posture; (iv) standing balancing, e.g., maintain stable stance and take steps when necessary; (v) turning left; (vi) turning right; (vii) trotting at small steps; and (viii) trotting at larger steps. These 8 *fundamental motor skills* are the building blocks for MELA to compose variable skills and produce adaptive behaviours.

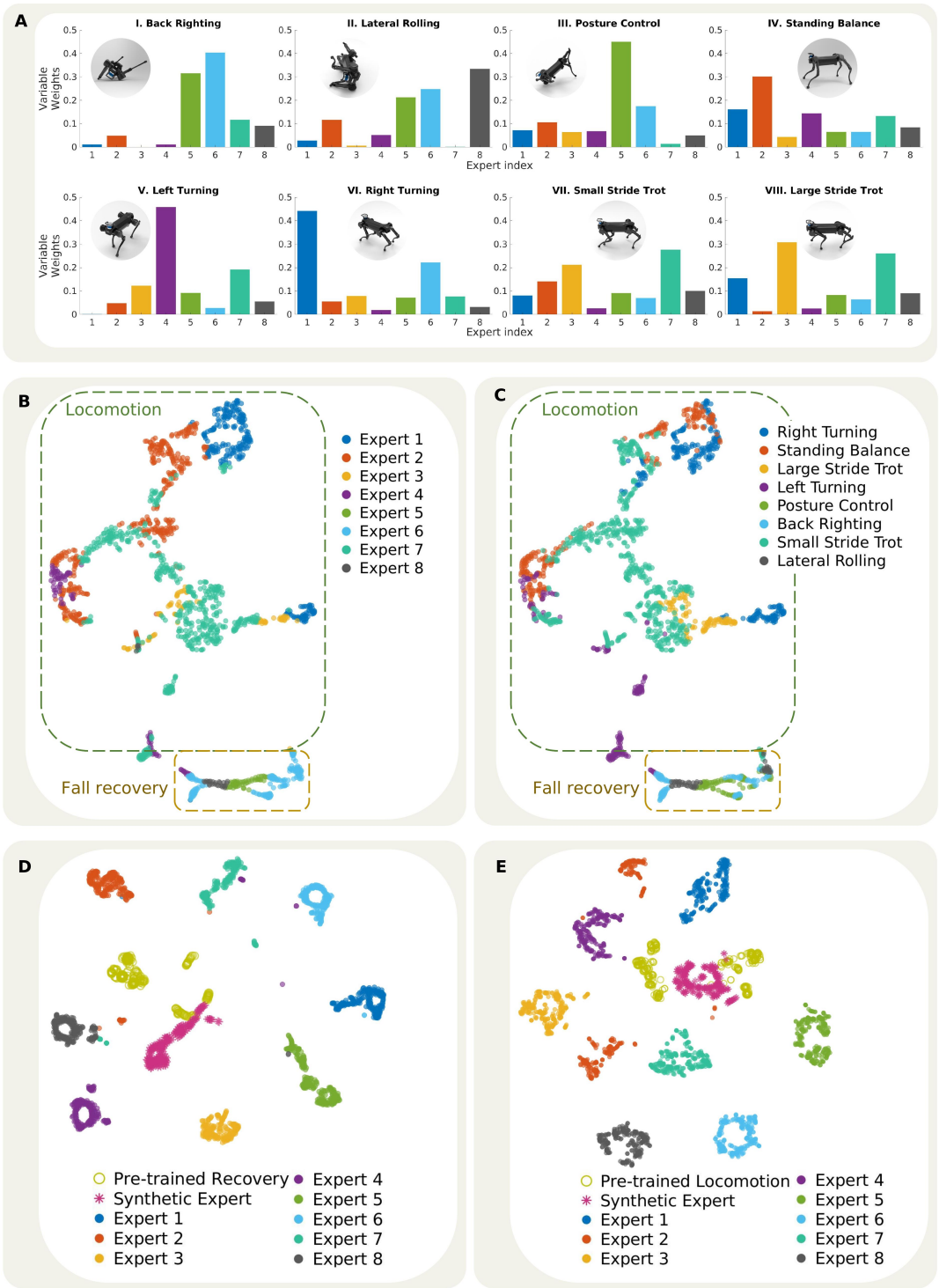


Figure 7.12: Analysis on the specialisation of experts, and the patterns of the gating network and the expert networks using the t-distributed Stochastic Neighbour Embedding (t-SNE). (A) Specialised activation of eight experts across different motor skills. (B-C) The 2D projection of the gating network's activation pattern by t-SNE. (B) Classified by the index of the dominant expert (see Fig. 7.12A). (C) Classified by the physical states during a distinct locomotion mode, e.g., trotting, balancing, turning left/right. (D-E) The 2D projection of the actions from the pre-trained, co-trained, and synthesised expert policies during fall recovery (D) and trotting (E) tasks.

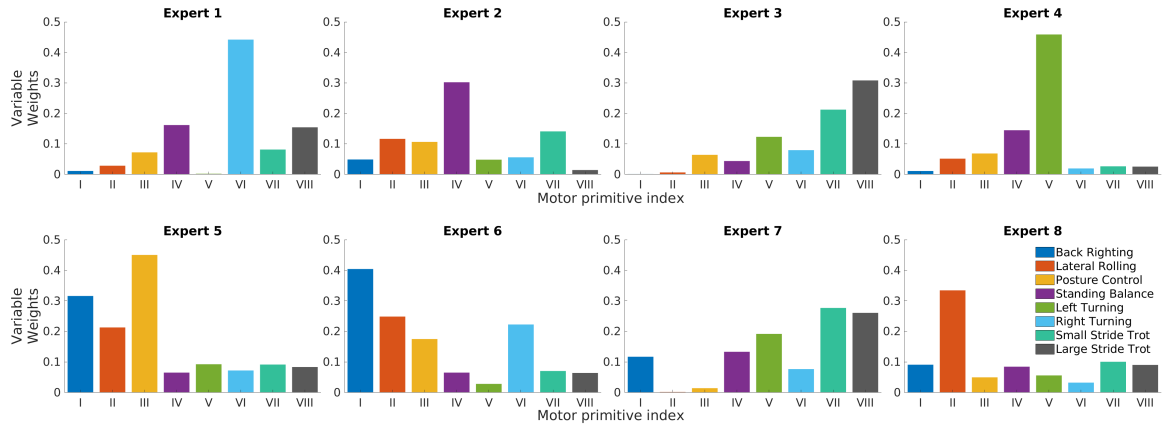


Figure 7.13: Activation patterns of experts across all motor skills. The unique activation pattern of each expert, in which the specialisation is indicated by the highest activation of a motor skill numerated by roman numbers. The specialised motor skills of expert 1-8 are: (i) right turning, (ii) balance stabilisation, (iii) large-step trotting, (iv) left turning, (v) posture control, (vi) back righting, (vii) small-step trotting, and (viii) lateral rolling, respectively. The data used for visualising the activation patterns are obtained from simulation tests of the trained MELA policy.

The skill specialisation and distribution among experts emerge naturally through the MELA co-training. Therefore, the order of the experts does not need to follow the numeration of the specialised motor skills. The primary specialisation of expert 1 to 8 over the motor skills are: turning right (skill vi), standing balance (skill iv), large-step trotting (skill viii), turning left (skill v), body posture control (skill iii), back righting (skill i), small-step trotting (skill vii) and lateral rolling (skill ii). As the result of introduced redundancy, expert 7 was exploited and trained as a complementary role in conjunction with expert 3 for trotting forward, i.e., expert 7 and 3 were specialised in trotting at small and large steps, respectively. An alternative visualisation can also be found in Fig. 7.13 by the activation patterns of experts across different motor skills.

7.3.3.2 Analysis of skill adaptation and transfer

Apart from the skill specialisation (Fig. 7.12A), we studied how skills are adapted and transferred in the MELA networks by using t-distributed Stochastic Neighbour Embedding (t-SNE) to analyse coactions of the gating network and the expert networks, respectively. The t-SNE algorithm is a dimensionality reduction technique to embed and visualise high-dimensional data in a low-dimensional space. It first computes a conditional probability distribution, representing the similarities of samples in the original high-dimensional space based on a distance metric, and then projects samples to a low-dimensional space in a probabilistic manner. Therefore, similar output actions from the networks will appear with high probability in the same neighbourhood as clustered points (Fig. 7.12B-E), and vice versa.

In Fig. 7.12B-C, the t-SNE analysis on the outputs of the gating network (the variable weights for all experts) reveals the relationship between experts and the resulting motor skills, and how the gating network synthesises experts after the MELA learning process. There are two maps of clusters in both Fig. 7.12B-C, which are grouped by dashed lines corresponding to locomotion (green) and fall recovery (bronze) separately, suggesting that the gating network perceives these two as different modalities. The t-SNE analysis is labelled according to the experts (Fig. 7.12B) and motor skills (Fig. 7.12C), respectively, where the clustered samples have matching distributions mostly between these two maps, indicating each expert's primary motor skill is in agreement with the activation patterns shown in Fig. 7.12A.

We also compared actions generated by all experts using t-SNE and revealed how multiple skills evolved and diversified after co-training. As shown in Fig. 7.12D-E, the actions generated from 8 experts are distant from each other, meaning that experts have been specialised towards more unique skills. The limited intersection between the clusters of the trained experts in stage 2 and the initial experts from stage 1 also implies that: the trained experts have diversified from the original warm-start skills and further acquired more profound and newly emerged skills during MELA's co-training stage. The data in Fig. 7.12D-E show interesting results: the cluster of actions from the synthesised network intersects with those from the pre-trained expert policies, meaning newly emerged behaviours of fall recovery and locomotion share some similarities with the original ones; the dynamically synthesised expert preserves partially the original skills, which are reconstructed by fusing 8 distinctive experts.

7.3.3.3 Multi-skill Locomotion

To validate the performance of the MELA policy, we designed experiments that were safe to execute on the real robot with an increasing number of locomotion modes: (i) single-mode fall recovery (Fig. 7.14A); (ii) double-mode left-right steering on the spot (Fig. 7.14B); (iii) triple-mode of simultaneous left-right steering and trotting (Fig. 7.14C); and (iv) target-following locomotion involving all modes, i.e., standing, left-right steering, trotting and fall recovery (Fig. 7.14D).

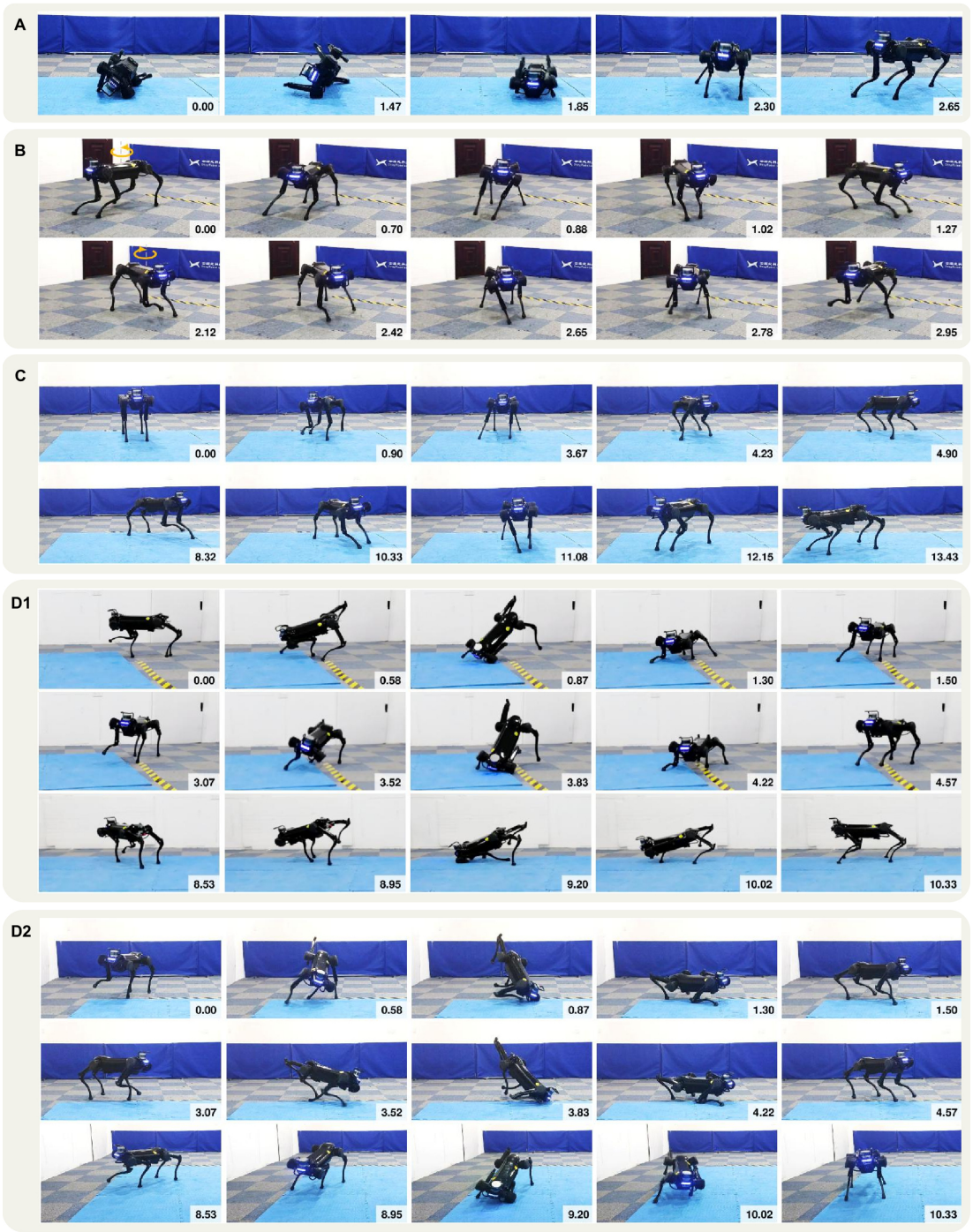


Figure 7.14: Dynamically synthesised MELA policy running on a real quadruped robot. (A) Successful fall recovery performed by the MELA expert, inheriting original skills from the pre-trained expert. (B) Newly emerged skills of dynamic steering on the spot naturally learned through the MELA framework (first to the right then to the left). (C) Target-following experiment with simultaneous trotting and steering. (D1) Target-following experiment showing the capability of failure-resilient trotting and critical recovery within one second (averagely 0.5 second for restoring body posture and 0.4 seconds for returning to the trotting mode). (D2) Elapsed-time snapshots of the same experiment as in (D1) from the front view. (Time in snapshots is in second).

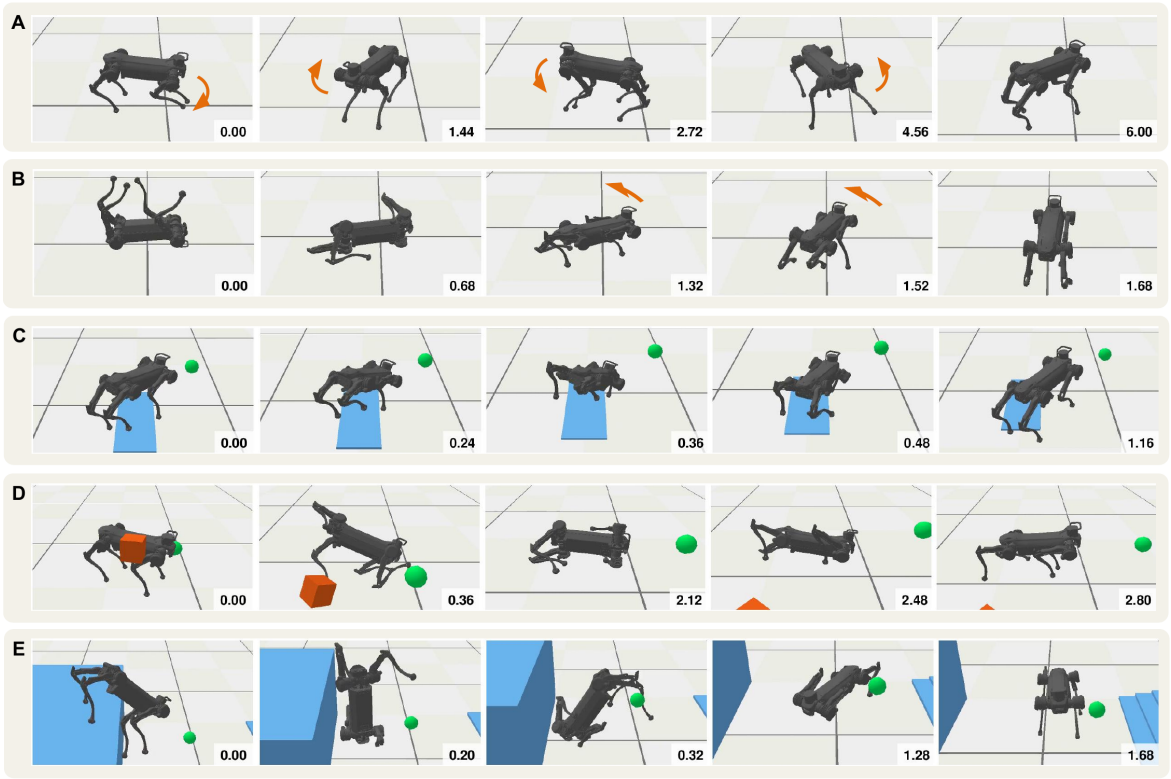


Figure 7.15: Five representative cases showing adaptive behaviours of the MELA expert under new situations in simulation. (A) An emerged behaviour of left and right steering on the spot. (B) An emerged behaviour of simultaneous steering and standing up while recovering from fall to trotting. (C) A tripping case caused by a slippery ground with a low friction coefficient of 0.1. The tripping and recovery behaviour was similar to that in the real experiment (see Fig. 7.14D1-D2). (D) A large impact disturbance caused by a 20 kg box hitting the robot at 8 m/s velocity. (E) An extreme crash test of blind locomotion over a cliff of 1 m height. (Time in snapshots is in second).

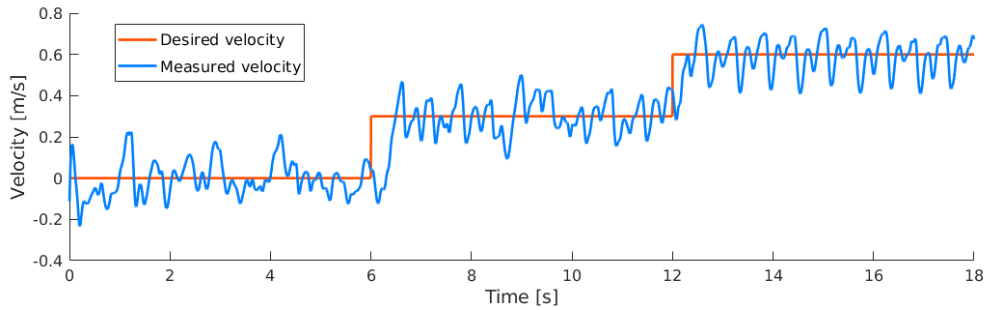


Figure 7.16: Forward trotting velocity during the variable speed trotting simulation. The robot adapted its trotting speed and followed the moving target.

In our study, adaptive behaviours refer to the online synthesised skills that adapt reactively

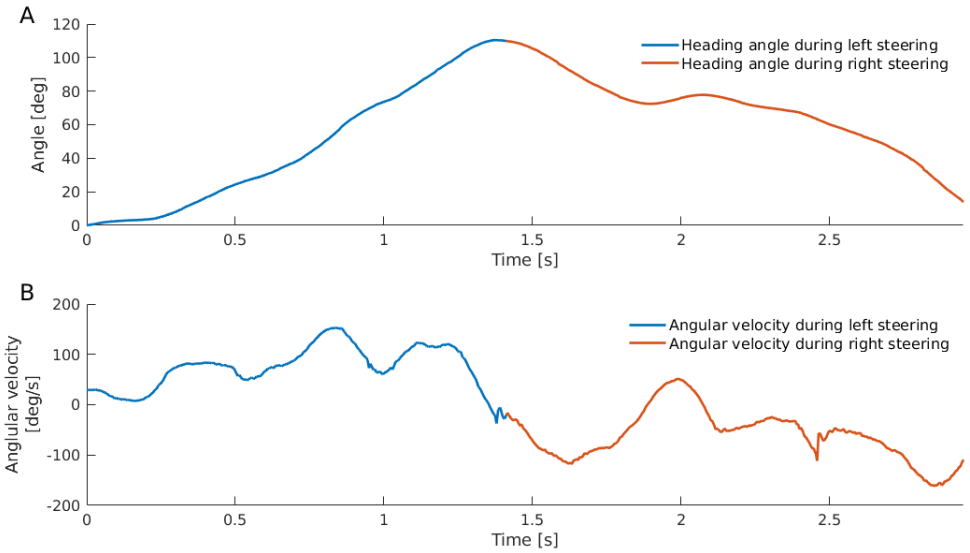


Figure 7.17: Heading angle and angular velocity during the steering experiment on the real robot. The heading data here corresponds to the experiment in Fig. 7.14B. (A) The robot first steered counter-clockwise towards the left and then clockwise towards the right. (B) The average yawing velocities were $1.6 rad/s$ $(92.0 deg/s)$ and $-1.1 rad/s$ $(-61.7 deg/s)$ during left and right steering, respectively, while the peak velocities reached $2.7 rad/s$ $(156.8 deg/s)$ and $-2.7 rad/s$ $(-156.9 deg/s)$.

to new situations. We summarise the adaptive behaviours achieved by MELA in two categories: (i) *Emerged skills* that are newly acquired during training in stage 2 of MELA, i.e., skills for steering and turning (see Fig. 7.14 and Fig. 7.15) and variable speed trotting (see Fig. 7.16); (ii) *Transitional skills* that coordinate dynamical transitions smoothly between different locomotion modes, e.g., transition from various failure poses to trotting (see Fig. 7.14 and Fig. 7.15B-E). Five representative cases of the adaptive behaviours from the MELA policy can also be found in Fig. 7.15.

Figure 7.14A shows successful fall recovery performed by the MELA policy, and the similarity with those in Fig. 7.11A indicates that the MELA policy has reused some pre-trained skills. Figure 7.14B shows that the MELA policy was able to infer the heading direction from the target location, and learned how to perform swift turning to track the changing target. During the left and right steering experiments (see Fig. 7.14B), the average turning velocities were 1.6 rad/s (92.0 deg/s) and -1.1 rad/s (-61.7 deg/s) with peak values at 2.7 rad/s (156.8 deg/s) and -2.7 rad/s (-156.9 deg/s), respectively (see Fig. 7.17). Although the experts initialised in MELA were only for trotting and fall recovery, MELA was able to reshape the existing experts for the steering tasks as one of the newly emerged skills.

Figure 7.14C-D show more challenging target-following tasks requiring simultaneous trotting and steering on the real robot. The task was to chase a virtual target given by the user

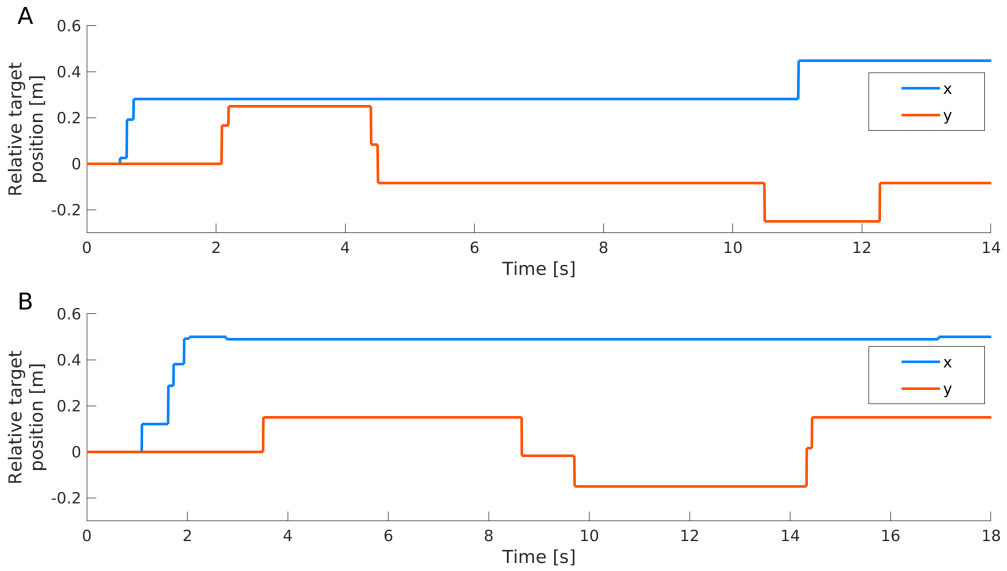


Figure 7.18: Relative target positions with respect to the robot from the user command as the input to the MELA networks during the real multimodal locomotion experiment. (A) The changing target position (x, y) during the real target-following experiment (Fig. 7.14C), and runtime was 14 seconds. (B) The changing target position (x, y) during the fall-resilient experiment (Fig. 7.14D1-D2), and runtime was 18 seconds.

command, i.e., a variable position vector with respect to the robot. In Fig. 7.14C, a smaller target position ahead of the robot was provided, e.g., 0.28 m in the heading direction (see Fig. 7.18A), and the robot performed left/right turning while trotting forward. In the next locomotion experiment (Fig. 7.14D1- D2), a farther target position of 0.48 m was commanded (see Fig. 7.18B). In this case, the robot chased such a distant target by trotting at larger steps, and the torque saturation of motors occurred more often (see Fig. 7.19), which naturally induced three tripping incidents. Key snapshots around the falling and reactive responses are presented in Fig. 7.14D. Once anomalous robot states were sensed, MELA was able to produce immediate reactions to restore balance within a second (see the body orientation in Fig. 7.20), and the robot recovered from tripping and continued locomotion without human intervention. The synthesised MELA expert demonstrated flexible transitions to resume locomotion in all these three successive incidents, and such reactive response using feedback is crucial for autonomous and resilient locomotion.

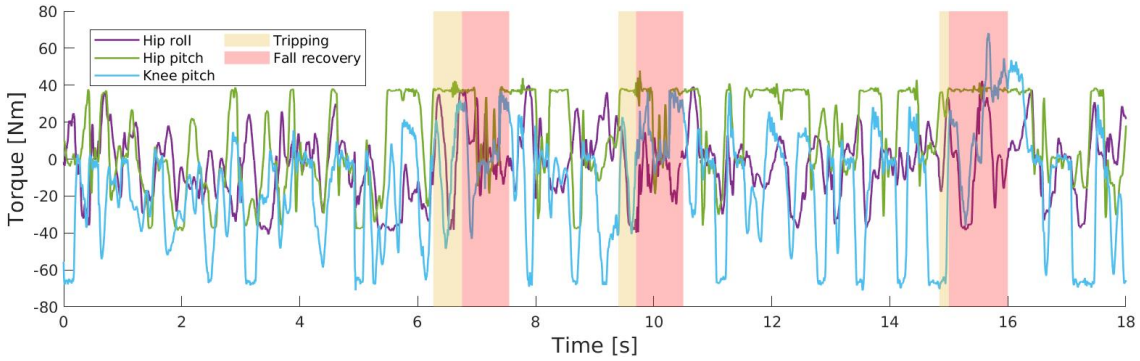


Figure 7.19: Measured torques of the front left leg during the real multimodal locomotion experiment (Fig. 7.14D1-D2). During this experiment, the robot trotted at large steps (see the larger commanded (x, y) target positions in Fig. 7.18B) and saturated the motor torques at times, e.g., the hip pitch joint. In this case, the torque-saturated leg was not able to move as intended and the robot stumbled and tripped (yellow regions), leading to falling motions (red regions).

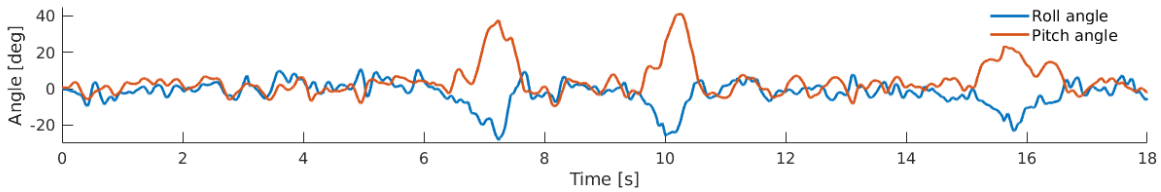


Figure 7.20: Roll and pitch angles during the real multimodal locomotion experiment. The measurements correspond to the experiment presented in Fig. 7.14D1-D2. Significant changes were observed in the body orientation by the roll and pitch angles during the tripping moments. The peaks in roll and pitch angles were up to 0.47 rad (26.7 deg) and 0.7 rad (39.8 deg), respectively. The recovery was accomplished within 1 second time.

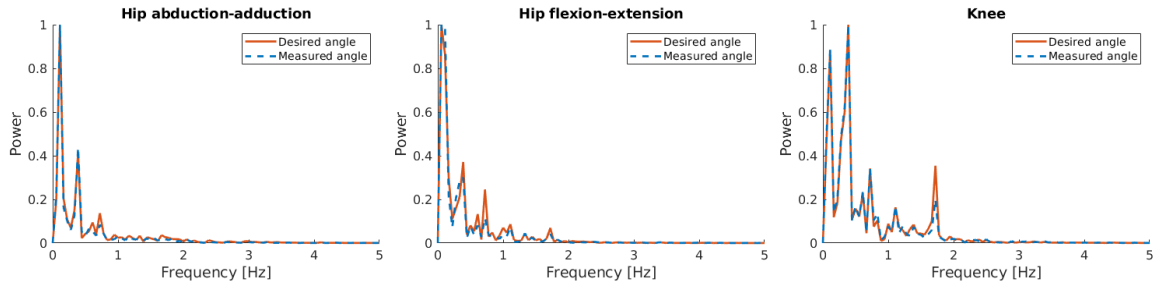


Figure 7.21: Normalised power spectrum analysis of motions during the real multimodal locomotion experiment (without the DC component). The data were collected from the experiment shown in Fig. 7.14D1-D2. The majority of the frequency components were below 1Hz, and some small components were around 1.67Hz corresponding to the trotting motions, which indicated that all useful motion components were unaffected by the action filters.

As shown in Fig. 7.14, the hierarchical MELA enabled the robot to complete four types of validations successfully, and demonstrated dynamic fall-resilient locomotion. The gating network in MELA has learned how to generate variable weights of all experts in response to the state feedback and provides smooth transitions across all modalities, and more data analysis of the representative case shown in Fig. 7.14D1-D2 can be found in Fig. 7.22. Meanwhile, all the trained experts were activated coherently to collaborate with each other under the regulation of the gating network, in order to synthesise an optimal skill suited for the situation.

From Fig. 7.22A, the changing weights around the boundaries of locomotion modes indicate that MELA produced smooth and quick transitions across successive modalities. The data in Fig. 7.22C-E shows a clear correlation between the sum of the experts' weights and the locomotion mode, suggesting that MELA trained the experts to be activated in a collaborative manner. Figure 7.19C delineates the sum of the weights of expert 3 and 7, which is high throughout the trotting mode but low during standing and fall recovery. Similar patterns of activation can also be observed in Fig. 7.22D, where the sum of expert 5, 6 and 8 has particularly high peaks during fall recoveries but constantly low in other modes. Figure 7.22E shows the differential weight between expert 1 and 4 (weight of expert 1 deducted by that of expert 4), and the complementary activation during left and right trotting, i.e., activation of expert 1 and inhibition of expert 4 during right trotting, and vice versa.

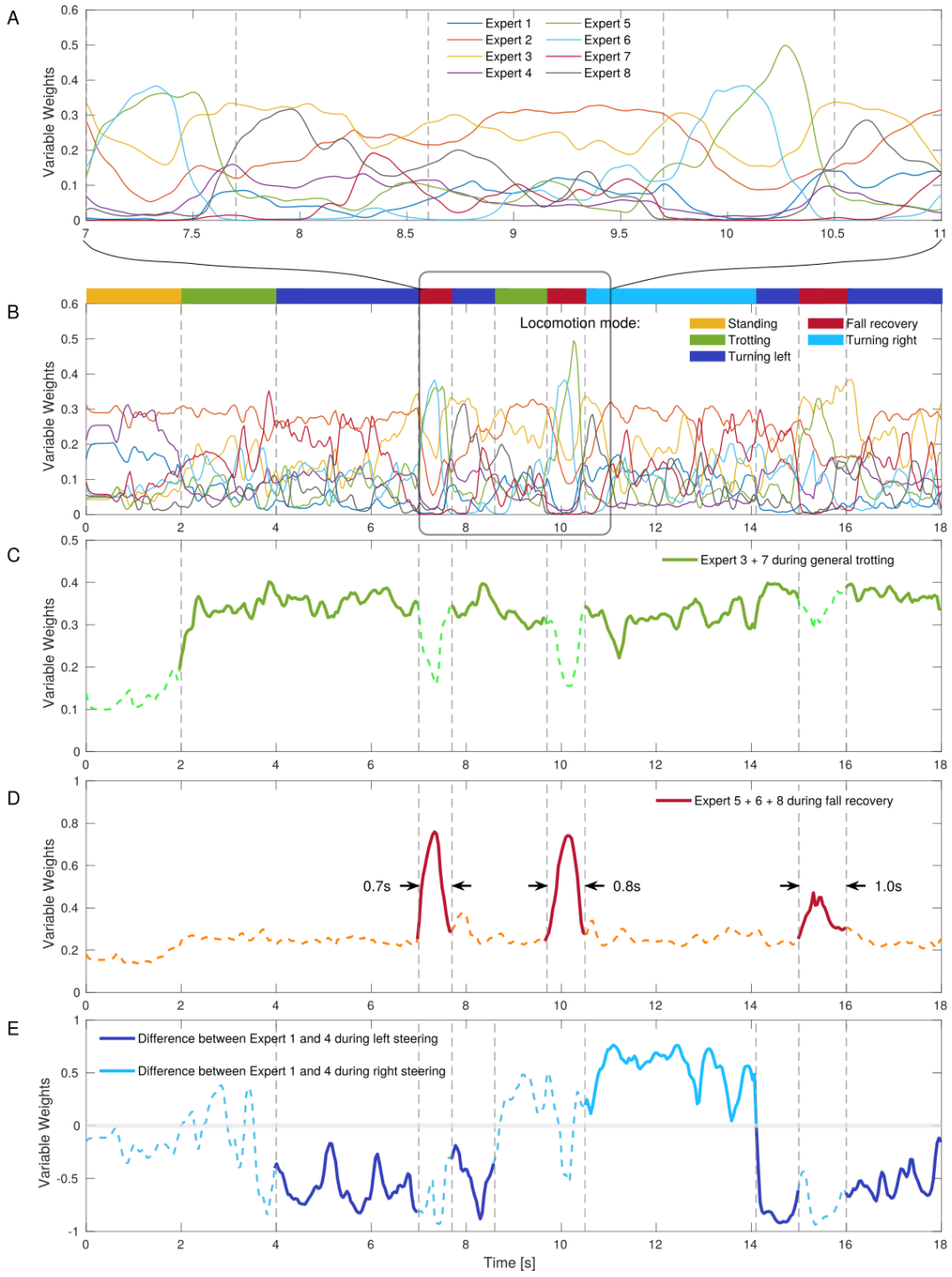


Figure 7.22: Continuous and variable weights of all experts during the real multimodal MELA experiment (Fig. 7.14D1-D2). (A) The variable activations within a zoomed period to show the transition of weights between multiple experts. (B) The variable activations of the entire multimodal locomotion with trotting, turning and fall recovery during a target-following task. (C-E) The activation levels of paired weights from collaborating experts, where the expert groups (3, 7), (5, 6, 8), (1, 4) cooperated together in trotting (forward, left, right), fall recovery, and turning (left, right), respectively.

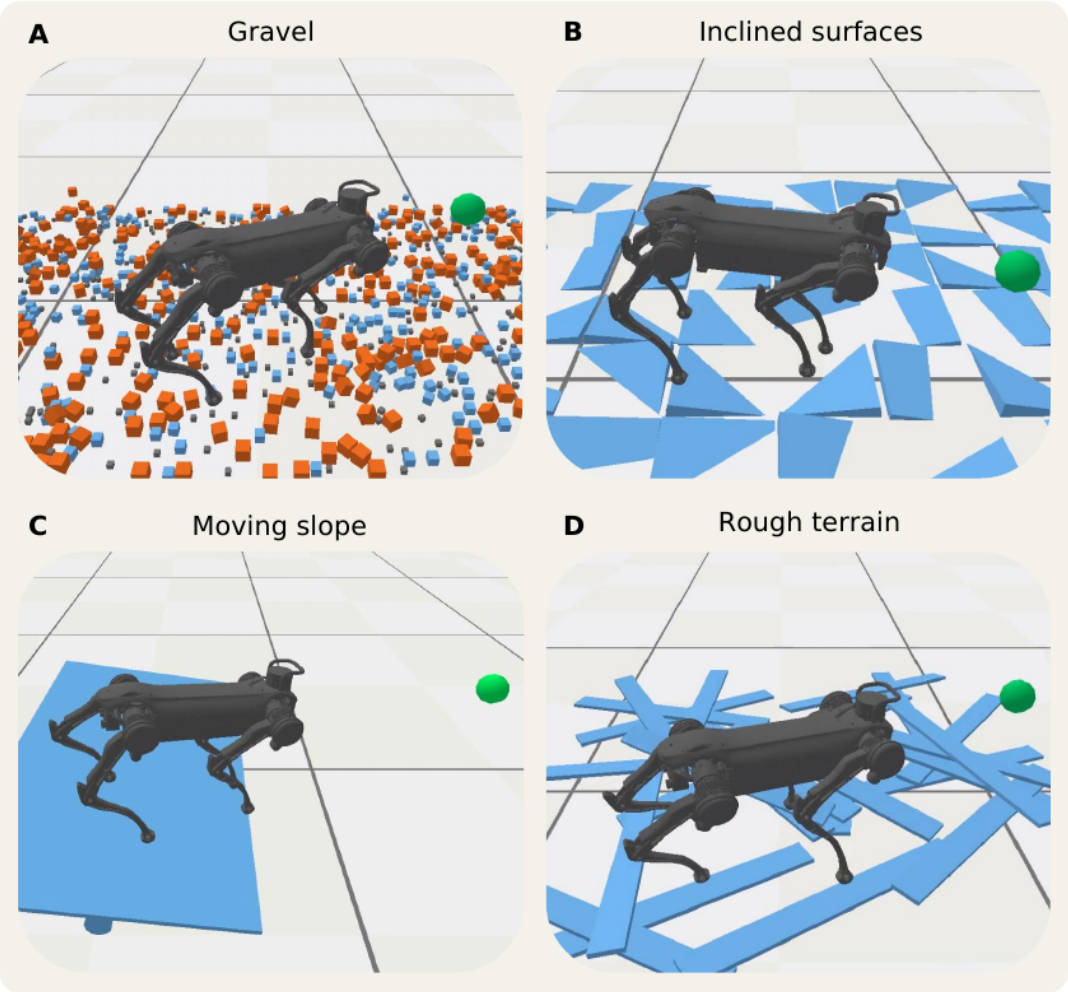


Figure 7.23: Four types of new terrains for testing the multi-skill MELA policy in the simulation. (A) The gravel is constructed by a variety of freely moving cubes with dimensions of 0.02m, 0.035m, and 0.05m. (B) The inclined surfaces consist of rectangular slabs (0.4 m x 0.4 m x 0.2 m), which are statically placed with random orientations on the ground. (C) The moving slope has a changing inclination created by a seesaw with a maximum inclination of 0.17 rad (10 deg). (D) The rough terrain created by planks with the mass of 2.5kg and a size of 1.2 m x 0.12 m x 0.02 m randomly distributed on the ground.

To further evaluate the performance, the MELA policy was validated by additional test scenarios in simulation that were not encountered during training, including gravel, inclined surfaces, moving slope, rough terrain (Fig. 7.23) as well as robustness tests with variations of masses and motor failures (Fig. 7.24). During successful locomotion in these unseen scenarios, the MELA policy performed versatile adaptations to new situations. We note that the MELA framework has learned how to deal with transitions at various gait phases (see analysis in Fig. 7.25-7.29), and the synthesised policy is different from the eight basic motor skills which indicates a nonlinear interpolated behaviour among expert skills (see Fig. 7.30). All these

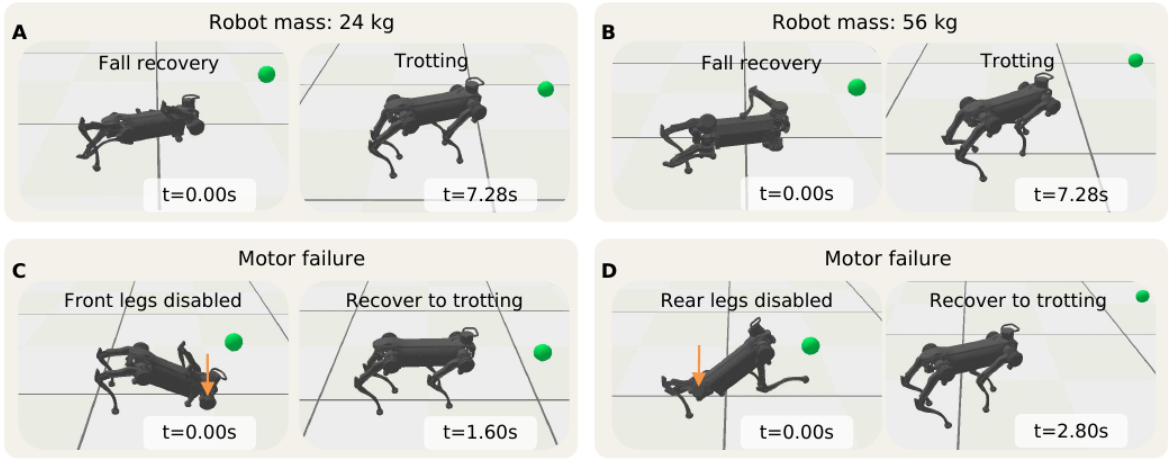


Figure 7.24: Simulated test scenarios for evaluating the robustness of the MELA policy. (A-B) Uncertainties in dynamic properties are simulated by modifying the robot model, i.e., robot mass with variations of 25%, 30% and 40% of the original value (40kg). We show snapshots with the mass variation of 40% as an extreme example. (A) Fall recovery and trotting with 60% of the original mass. (B) Fall recovery and trotting with 140% of the original mass. (C-D) Motor failures are emulated by disabling (zero torque) the front legs (C) and rear legs (D) respectively for one second. In both cases, the robot was able to recover from failures and accomplish the task.

experiments and simulations validate MELA's capability of producing flexible behaviours in a variety of unseen situations.

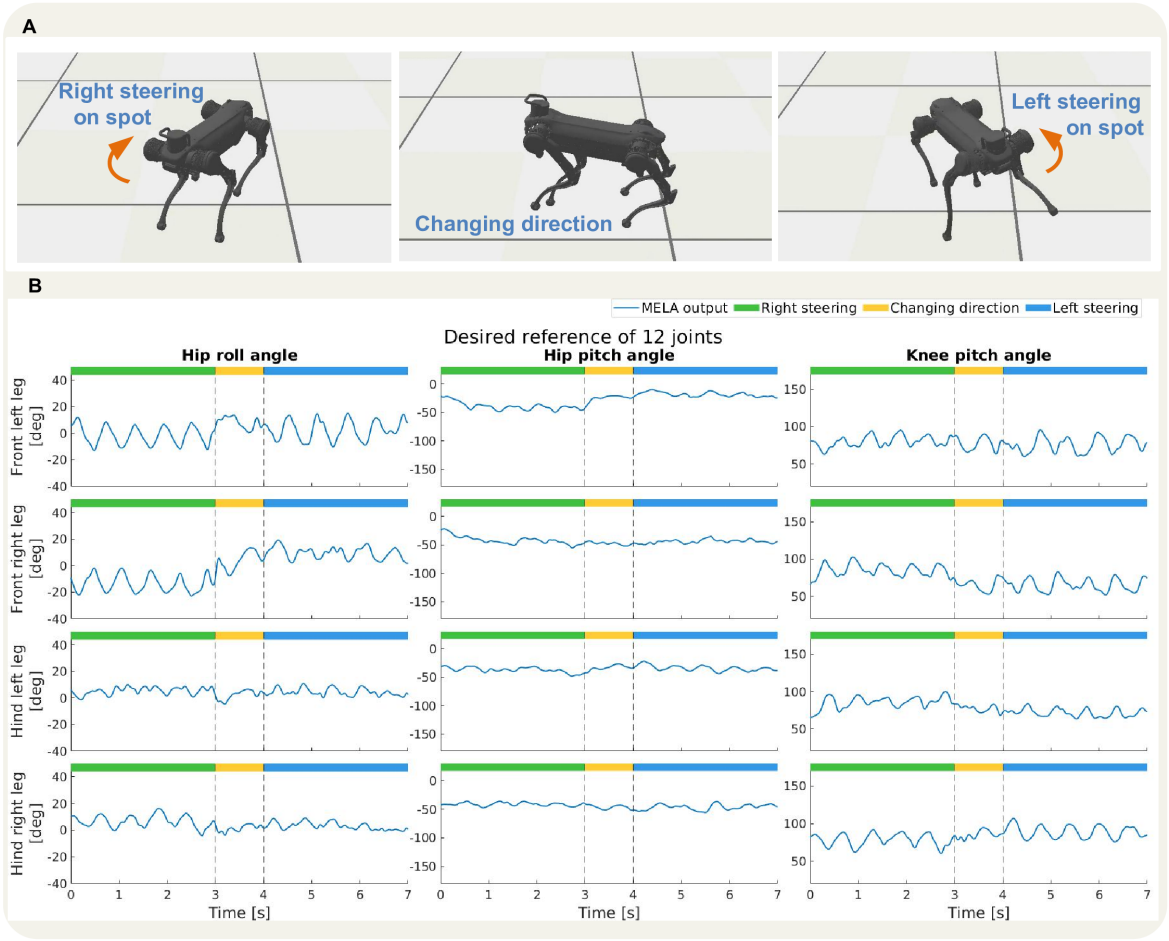


Figure 7.25: Representative adaptive behaviour from the simulated scenario of steering on spot (Fig. 7.15A). (A) Snapshots depicting the behaviours during left and right steering. (B) Position references of all the joints during left and right steering phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

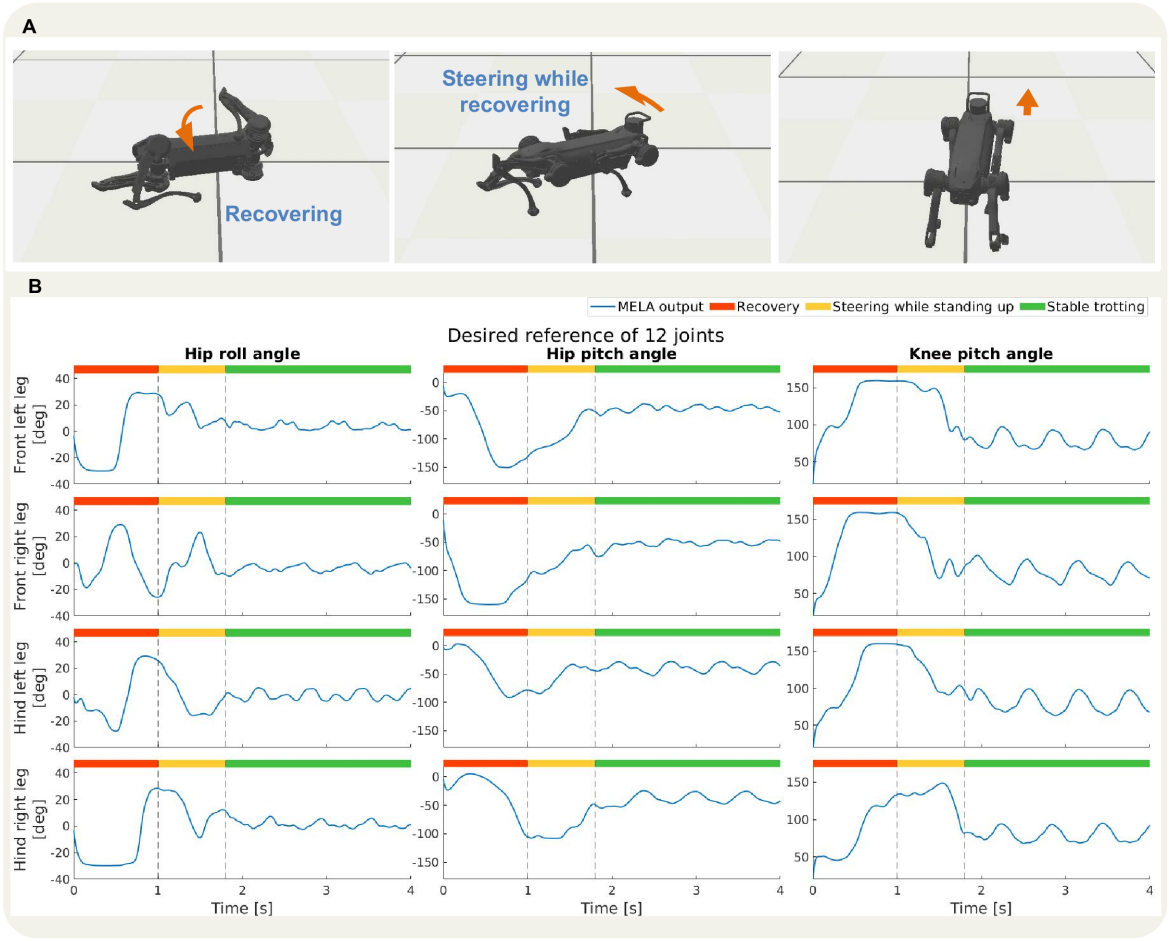


Figure 7.26: Representative adaptive behaviour from the simulated scenario of steering while recovering to trotting (Fig. 7.15B). (A) Snapshots depicting the behaviours during recovery and steering. (B) Position references of all the joints during recovery, steering, recovery, and trotting phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

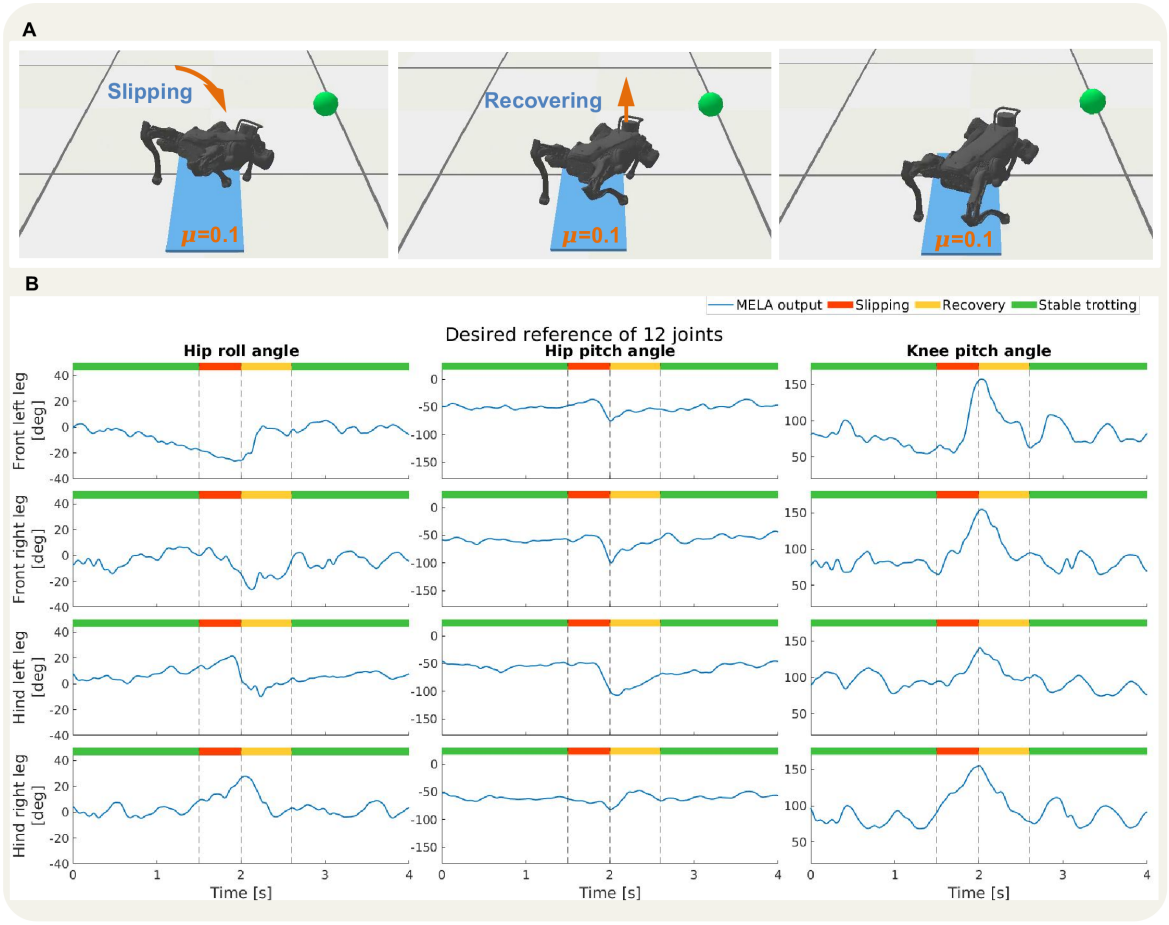


Figure 7.27: Representative adaptive behaviour from the simulated scenario of tripping (Fig. 7.15C). (A) Snapshots depicting the behaviours during slipping and recovery. (B) Position references of all joints during slipping, recovery, and trotting phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

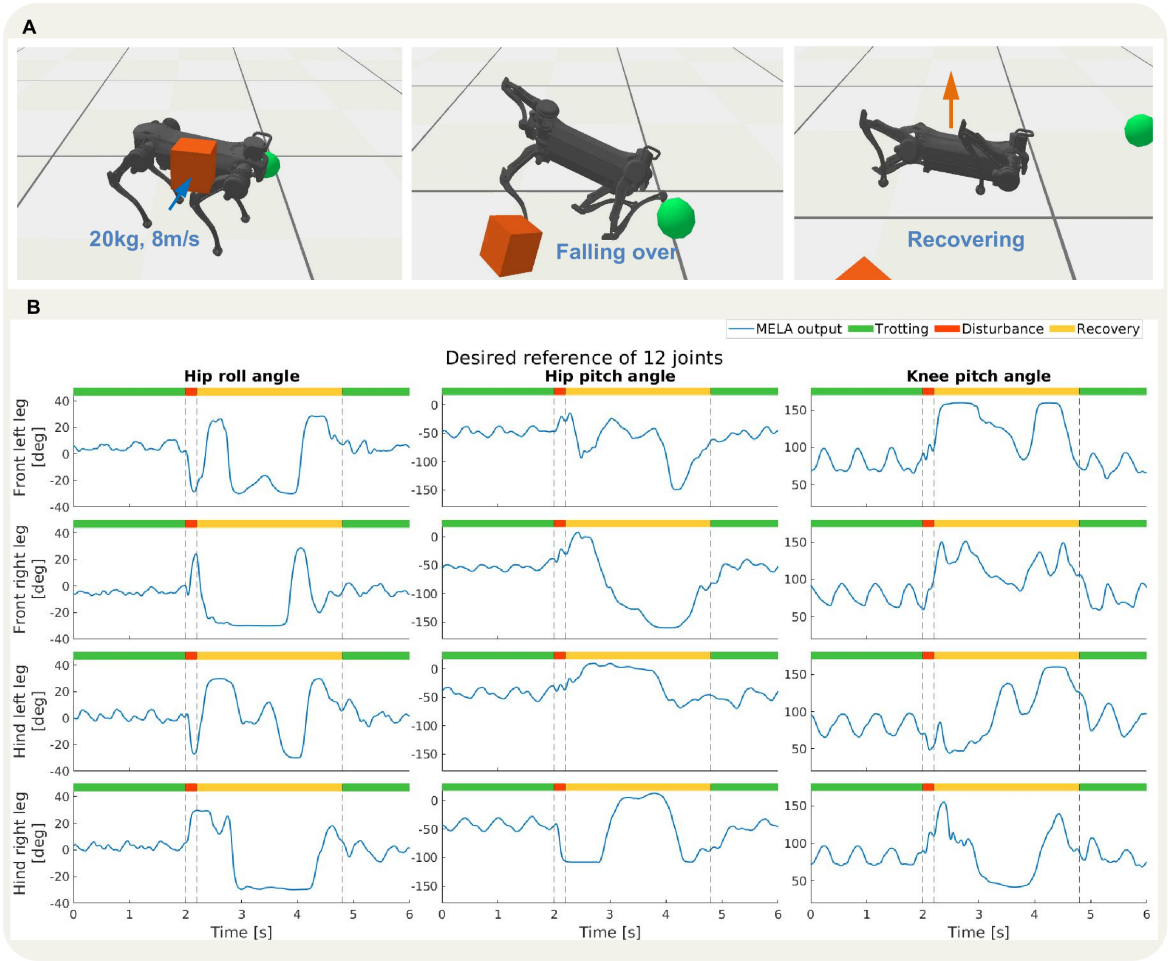


Figure 7.28: Representative adaptive behaviour from the simulated scenario of a large impact (Fig. 7.15D). (A) Snapshots depicting the behaviours during the moment of disturbance and recovery (B) Position references of all the joints during trotting, disturbance, and recovery phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

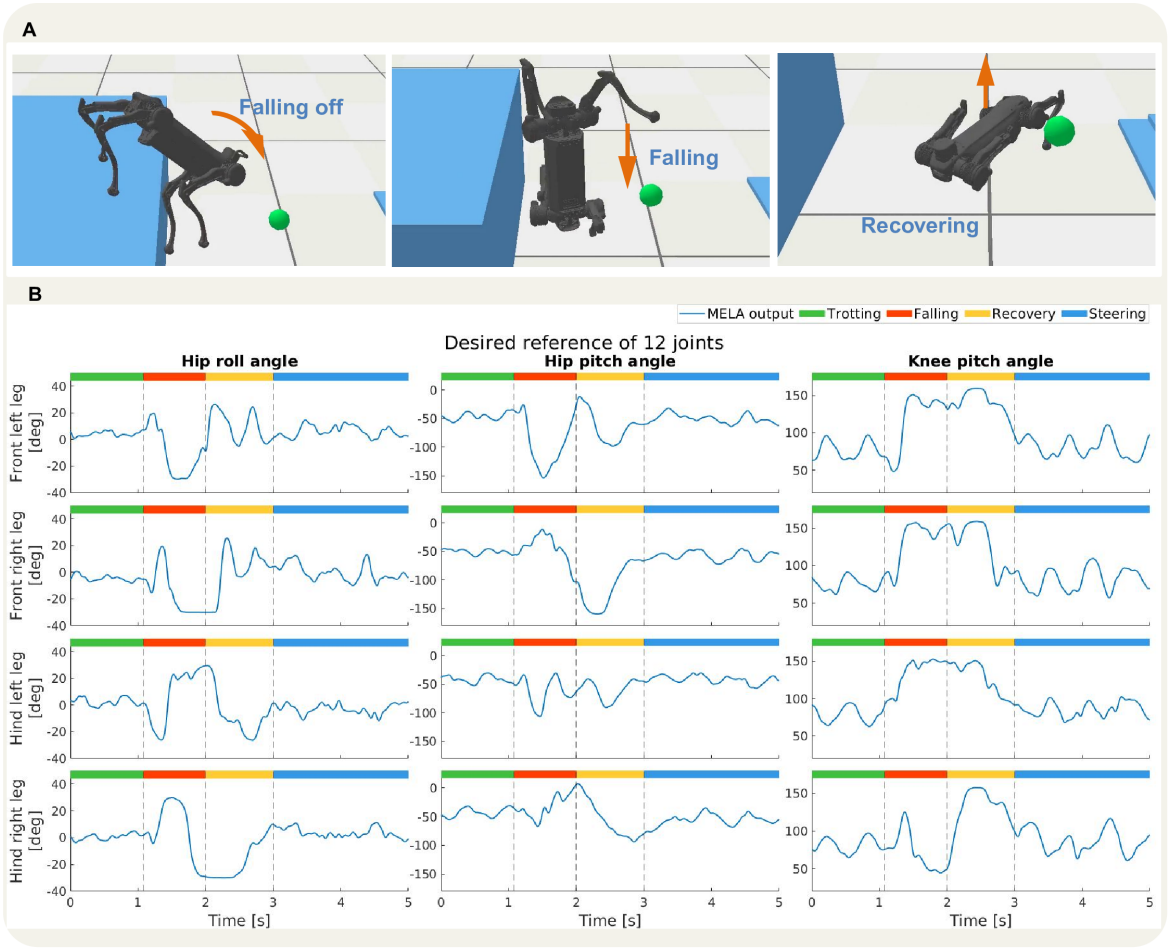


Figure 7.29: Representative adaptive behaviour from the simulated scenario of falling off a cliff (Fig. 7.15E). (A) Snapshots depicting the behaviours during falling and recovery. (B) Position references of all the joints during falling, recovery, and steering. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

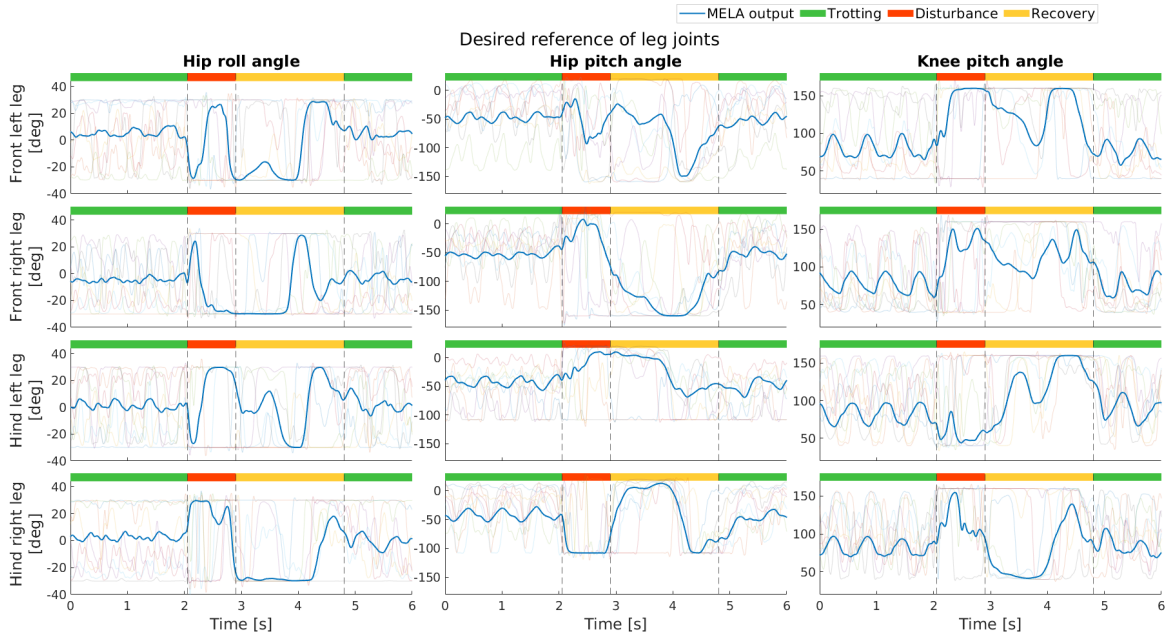


Figure 7.30: Analysis of responses from the MELA policy during the simulated scenario of a large external perturbation (Fig. 7.15D). We use the case of a flying box impact as a representative example to show how the policy actively reacts to perturbations with a smooth and seamless transition. The coloured bars at the top of each plot show the 3 phases during the cube impact scenario: the green, red, and yellow phases represent stable trotting, the time during the force impact by the high-speed cube, and the recovery process, respectively. In each subplot, the 8 semi-transparent lines are the outputs of each individual expert, and the blue solid line is the output of the synthesised MELA network. The outputs of the synthesised policy (solid blue lines) have very different characteristics from that of the 8 basic experts during all the phases, which suggests an interpolated behaviour and a nonlinear synthesis among the expert skills.

7.4 Discussion

This study aims to achieve versatile robot motor skills in contact-rich multimodal locomotion. In contrast to most solutions dedicated to separate narrow-skilled tasks, we approached this challenge by a hierarchical control architecture of multi-expert learning – MELA – which is able to generate adaptive motor skills and achieve a breadth of locomotion expertise. In particular, MELA learns to generate adaptive behaviours from trained expert skills by dynamically fusing a new synthesised neural network, i.e., a feedback policy that reacts quickly to new situations. This is essential for autonomous robots to respond rapidly in critical conditions and is more useful for the mission success in real-world applications.

Compared to MoE, MELA's approach of fusing network parameters prevents the expert imbalance problem while providing diversity among the expert skills. As a result, all experts are required to have the same neural network structure for implementing MELA. The training of MELA is a two-stage process with an initialisation of fall recovery and trotting policies at the first stage, and the multi-expert co-training with the gating network at the second stage. The t-SNE analysis of all expert networks and the gating network suggest that: the collection of multiple experts have expanded the initial pre-trained expert skills and acquired more distinct and diverse skillsets; the high-level gating network has learned to distinguish each expert and blend weights of each specialisation according to different conditions; and the composed synthesised MELA expert partially preserves some of the original skills.

The experimental and simulation results outline MELA's key contributions in learning a variety of adaptive behaviours from specialised experts, the adaptation to changing environments, and the robustness against uncertainties. The experimental results show that MELA achieved multimodal locomotion with agile adaptation and fast responses to different situations and perturbations, i.e., smooth transitions between standing balancing, trotting, turning, and fall recovery. As a learning-based approach, MELA leverages computational intelligence and shows the advantage of generating adaptive behaviours compared to traditional approaches that purely rely on explicit manual programming.

Though our current MELA scheme is able to generate adaptive policies, it has no visual and haptic perception which are critical for long-term motion planning [155], dynamic manoeuvres [156], and utilisation of affordance to coordinate whole-body support poses [101]. To acquire more advanced motion intelligence in unstructured environments, future research needs to integrate visual cues and haptic sensing to develop environment-aware locomotion.

While scaling up the number of modalities, training in physics simulation may impose some limitations. Though all policies were validated by the Jueying robot in five locomotion modes, the discrepancy between the simulation and the real world may accumulate and arise as an issue, when the number of tasks increases. Since the scope of this research is to achieve a diversity of reactive skills rather than sim-to-real transfer, we performed training in simulation and avoided potential damage to the real 40kg-robot during the exploration of the learning

algorithm. Based on the results of MELA, the future work will be on the learning algorithms that can refine motor skills safely on real hardware for more complex multimodal tasks.

Chapter 8

Conclusion

8.1 Conclusion

The contribution of the work presented in this thesis are threefold. (i) We have developed a versatile DRL framework that is able to balancing, walking, and fall recovery motor skills. (ii) We proposed a Multi-Expert Learning Architecture (MELA) for DRL that is able to produce dynamic multi-skilled locomotion behaviors. (iii) We have successfully implemented learned control policies on real-world robots.

In this thesis, the first major contribution is that we have presented a number of Deep Reinforcement Learning (DRL) based control frameworks for tackling three individual control policies for bipedal and humanoid robots in simulation, namely, balancing, walking, and fall recovery. Which are all essential skills for the successful locomotion of terrestrial legged robots. The learning frameworks presented share the same underlying design principles, with slight variations to solve the different problems of balancing, walking, fall recovery, and multi-skill locomotion. The shared common design principles are reflected in the following: (i) state representation, (ii) action representation, (iii) reward design, and (iv) sampling distribution.

The second major contribution is that we have proposed a novel multi expert learning architecture with a hierarchical structure to combine the learned balancing, walking and fall recovery policies to create a single unified multi-skill policy through a multi-expert network structure and a two stage pre-training and co-training procedure. We showed that these three motor skills can be merged into a single control policy to synthesize new motor skills within our novel hierarchical multi-expert learning architecture. The performance of the multi-skill policy is later validated on a real quadruped robot.

The final contribution is that we have demonstrated the feasibility of implementing DRL based control policies on real robot systems. However, while we have successfully deployed our learning framework on a real quadruped robot, we have also observed that the learned control policies tend to generate jerky motions with high frequency oscillations, which is a phenomenon that have been observed in other work [104]. We introduced two techniques

termed *smoothing loss* and *action filtering* to guide the policy to learn to generate smoother motions which can be executed on the robot and directly and safely. The proposed techniques are crucial components of the framework as they prevent the policy from generating abrupt and jerky motions that are infeasible to be deployed on the real system.

8.2 Limitations and Future Extensions

Though our proposed DRL scheme is able to learn flexible policies for both humanoid and quadruped robots, there are still several limitations. In this section, we discuss the limitations in the current work and also propose future research directions to improve and extend upon the methods presented within the thesis.

8.2.1 Limitations

8.2.1.1 Hardware Implementation on Humanoids

One limitation of the work is that the balancing, walking, and fall recovery policies have not been validated in the real world on real humanoid robots. Although the effectiveness of the approach has been validated by the successful implementation of multiple motor skills on real quadrupedal robots, it can still be argued that the control of humanoid robots is much more challenging and that success on quadrupeds does not guarantee that the learned control policies will function as intended on real humanoids. One interesting next step is to tackle the difficulties during hardware implementation and deploy the control policies on real humanoids. It will then be possible to investigate the performance of the learned policy while implemented in real-world environments.

8.2.1.2 Visual and Haptic Feedback

The research conducted in this thesis does not utilize the high dimensional visual feedback from cameras or the haptic feedback from artificial skin and thus have no means to obtain information from the external environment. Although the current selection of state feedback is sufficient for blind traversing over relatively flat terrains, the lack of visual and haptic perceptions may impose future limitations when dealing with more complex tasks. There are a few studies that have trained policies with visual feedback in the form of height maps, which were shown to be capable of navigating through obstacles and terrains in simulation [6], [31]. There are also attempts in training DRL policies with tactile feedback to perform manipulation tasks where the sense of touch plays a crucial role [157], [158]. Visual and haptic perceptions are critical for long-term planning and utilising affordances. To acquire more advanced motion intelligence in unstructured environments, more research needs to be done to utilize useful visual and haptic inputs and to incorporate visual and haptic feedback effectively.

8.2.1.3 Learning on Hardware

The fact that all the policies were trained in physics simulation may impose future limitations when scaling up the modality. Although all policies were validated successfully by the Jueying robot across five locomotion modes, discrepancies between the simulations and the real world may arise when the number of tasks increases. Since the scope of this research is to achieve a diversity of skills at runtime instead of sim-to-real transfer, we performed simulated training for the safety of the robot and lab personnel. During the exploration of the learning algorithm, the robot was made to progress through infeasible and aggressive actions, which led to inevitable crashes in all possible ways. Duplicating such learning with a real robot would have caused significant damage to, and incur repair costs for, the 40kg Jueying robot [42]. Based on the results of MELA, we suggest future work on learning refined motor skills on real hardware for more complex multi-modal tasks, which are hard to simulate accurately and feature accumulated discrepancies.

A major limitation of the work presented in this thesis is that the policy is not trained on real hardware and thus may suffer from discrepancies between the simulations and the real world. No matter how well the simulation resembles the real world, there will always be gaps between simulation and reality, which will lead to performance issues. One way of circumventing the issue is to use data gathered from the real environment to train the policy. Learning on hardware is a method that can be used to bypass the sim-to-real gap, as the policy is learned directly on the real hardware, avoiding any model discrepancies [159].

The most challenging aspect of learning directly on hardware is the sample efficiency. It takes time to gather the samples, and it is not possible to run the robot for too long due to possible hardware damage and fatigue. Some work pre-train the policy in simulation and use real-world data for fast adaptation to uncertainties in the environment in real time [160]. Other studies abandon simulation data completely and train the policy directly on real hardware [159]. For future work, meta-learning and model-based RL could be adopted to increase the sampling efficiency, thus making it feasible to learn on hardware with a minimum number of trials.

Additional safety measures must be considered when learning directly on hardware. Under such circumstances, safety constraints must be included in the learning framework to allow the robot to explore and operate in a safe manner in the real world to minimize physical damage[161].

8.2.2 Future Extensions

8.2.3 Exploring Diverse Skillsets

The MELA results presented in Chapter 7 are limited to a small number of motor skills, while, in principle, MELA's framework is highly scalable and could be extended to incorporate more motor skills by increasing the number of experts within the MELA network. The motor skills

investigated in this thesis were limited to balancing, trotting, and fall recovery. However, humans and animals exhibit many other motor skillsets. In the character animation field, researchers have trained human characters to punch and kick as well as perform backflips and various other motions in physics simulation by imitating real human data. In robotics, researchers have also designed controllers to perform dynamic motions, such as jumping, backflipping, and galloping, using real-world quadrupedal robots [54], [162], [9]. An interesting research direction would be to study unique and challenging motor skills and incorporate these new motor skills into the existing MELA framework, creating a highly versatile multi-expert control policy with the ability to adapt to a wide range of scenarios.

The proposed MELA has only been implemented to learn skills for quadruped locomotion. However, other challenging applications such as manipulation might also benefit from the multi-expert structure of MELA. It would be an interesting research direction to implement MELA for robotic arms and multi-fingered manipulators and see if MELA is capable of learning the necessary skillsets for solving the challenging task of dexterous manipulation of objects with complex geometries.

8.2.4 Policy Transfer

Training new policies demands a lot of computation. The efficiency can be increased if previously trained skills can be transferred across different domains to facilitate the acquisition of new skills. Transfer learning is a concept in machine learning that focuses on leveraging past knowledge to improve and accelerate the learning in order to solve new tasks. It has had notable successes within the RL paradigm [163].

The skill transfer presented in Chapter 7 for the MELA network is done by transferring the network parameters of the initial fall recovery and trotting policies directly. The proposed MELA was designed with the ease of policy transfer in mind. Since each expert encapsulates a single behaviour, the hierarchical structure of MELA allows new expert policies to be transferred into the framework by simply adding a new expert network. This addition can be achieved easily, as the source policy has state and action representations that are similar to those of the destination MELA policy. There is the added benefit that the policies are trained on the same robot, making the transfer easier.

However, in many cases, the goal is to transfer skills between robots with different morphologies and state and action representations [164]. Such cases demand more intricate approaches for transferring known skills. Policy transfer is an interesting research topic worth pursuing for future research, as the ability to transfer a policy easily and effectively between different tasks and robots will increase the versatility of the MELA network.

8.2.5 Neural Network Structures

We have experimented on 4 different neural network structures within the thesis: (i) Fully Connected Neural Network, (ii) Phased-Functioned Neural Network [89], (iii) Mode-Adaptive Neural Network [90], and (iv) our proposed MELA network. However, there exists many other neural network architectures with different and unique characteristics and advantages.

Researchers have explored the usage of various different neural network design within the deep reinforcement learning paradigm, such as Long Short Term Memory (LSTM) and Convolutional Neural Network (CNN). LSTM have been used for tasks that require the memorization of past experience such as maze navigation [165], while CNN have been used for tasks that involve visual information such as terrain traversing and object manipulation [166]. For future work, it might be worth exploring the strength and weaknesses of different neural network structures and investigating how to design the suitable neural network for the desired application.

8.2.6 Imitation Learning from Nature

Quadruped and humanoid robots have their equivalent counterparts in nature, which are dogs, cats and humans. It makes sense to use the motions of humans and dogs as an expert demonstration for the artificial learning agents to learn more diverse motor skills. There have been quite a few works that have successfully used real-world motion data of humans and animals to guide the agent to learn dynamic and natural looking motions [8], [9].

The first and most crucial step of Imitation Learning is to obtain the necessary expert demonstrations. A common approach is to capture the motions using a specialized motion capture device [89], [9]. The advantage of such methods is that they can provide accurate and dense information on the orientation and position of the limbs and joints. However, the disadvantage is also quite clear, the motion capture device are expensive and may sometimes restrict the movement of the subject wearing the device. Recognizing the limitation of mocap devices, some researchers have attempted to explore other sources of motion data that are more easily accessible. Peng et. al. proposed a framework that is capable of extracting 2D and 3D pose estimation from videos for the imitation of a broad range of dynamic skills [167],

In this thesis, we have only used imitation learning to achieve the motion of bipedal walking and quadrupedal trotting. For future work, we will further investigate the use of imitation learning to learn more motor skills such as bipedal running, quadrupedal bounding, pacing, and galloping etc.

Bibliography

[1] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.

[2] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[3] H. Van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE, 2007, pp. 272–279.

[4] H. Van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Reinforcement learning*. Springer, 2012, pp. 207–251.

[5] X. B. Peng, G. Berseth, and M. Van de Panne, "Dynamic terrain traversal skills using reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–11, 2015.

[6] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–13, 2017.

[7] X. B. Peng, G. Berseth, and M. Van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.

[8] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[9] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[10] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 1988.

[11] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[13] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[14] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, 2016, pp. 2829–2838.

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

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

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

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

[19] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 239–246.

[20] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. center of pressure-zero moment point," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 5, pp. 630–637, 2004.

[21] M. Vukobratović and B. Borovac, "Zero-moment point—thirty five years of its life," *International journal of humanoid robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[22] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics*. Springer, 2014, vol. 101.

[23] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*. IEEE, 2006, pp. 200–207.

[24] Z. Li, C. Zhou, Q. Zhu, and R. Xiong, "Humanoid balancing behavior featured by under-actuated foot motion," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 298–312, 2017.

[25] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2017, pp. 1–13.

[26] F. Peng, L. Ding, Z. Li, C. Yang, and C.-Y. Su, "Optimal balancing control of bipedal robots using reinforcement learning," in *2016 12th World Congress on Intelligent Control and Automation (WCICA)*. IEEE, 2016, pp. 2186–2191.

[27] C. Zhou and Q. Meng, "Dynamic balance of a biped robot using fuzzy reinforcement learning agents," *Fuzzy sets and Systems*, vol. 134, no. 1, pp. 169–187, 2003.

[28] Y.-z. Chen, W.-Q. Hou, J. Wang, J.-W. Wang, and H.-x. Ma, "A strategy for push recovery in quadruped robot based on reinforcement learning," in *2015 34th Chinese Control Conference (CCC)*. IEEE, 2015, pp. 3145–3151.

[29] C. Yang, T. Komura, and Z. Li, "Emergence of human-comparable balancing behaviours by deep reinforcement learning," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 372–377.

[30] C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li, "Learning whole-body motor skills for humanoids," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 270–276.

[31] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[32] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[33] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, pp. 663–670.

[34] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[35] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

[36] J. Merel, A. Ahuja, V. Pham, S. Tunyasuvunakool, S. Liu, D. Tirumala, N. Heess, and G. Wayne, "Hierarchical visuomotor control of humanoids," *arXiv preprint arXiv:1811.09656*, 2018.

[37] C. Yang, K. Yuan, S. Heng, T. Komura, and Z. Li, "Learning natural locomotion behaviors for humanoid robots using human bias," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2610–2617, 2020.

[38] J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess, "Learning human behaviors from motion capture by adversarial imitation," *arXiv preprint arXiv:1707.02201*, 2017.

[39] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[40] A. Sharma and K. M. Kitani, "Phase-parametric policies for reinforcement learning in cyclic environments," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[41] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1241–1246.

[42] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[43] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," *arXiv preprint arXiv:1910.02812*, 2019.

[44] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. van de Panne, "On learning symmetric locomotion," in *Motion, Interaction and Games*, 2019, pp. 1–10.

[45] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–12, 2018.

[46] Z. Li, C. Zhou, J. Castano, X. Wang, F. Negrello, N. G. Tsagarakis, and D. G. Caldwell, "Fall prediction of legged robots based on energy state and its implication of balance augmentation: A study on the humanoid," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5094–5100.

[47] S. Wang and K. Hauser, "Real-time stabilization of a falling humanoid robot using hand contact: An optimal control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 454–460.

[48] S. Wang and K. Hauser, "Realization of a real-time optimal control strategy to stabilize a falling humanoid robot with hand contact," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3092–3098.

[49] S. Wang and K. Hauser, "Unified multi-contact fall mitigation planning for humanoids via contact transition tree optimization," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[50] J. Stückler, J. Schwenk, and S. Behnke, "Getting back on two feet: Reliable standing-up routines for a humanoid robot." in *IAS*, 2006, pp. 676–685.

[51] F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, S. Kajita, K. Yokoi, H. Hirukawa, K. Akachi, and T. Isozumi, "The first humanoid robot that has the same size as a human and that can lie down and get up," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1633–1639.

[52] C. Semini, J. Goldsmith, B. U. Rehman, M. Frigerio, V. Barasuol, M. Focchi, and D. G. Caldwell, "Design overview of the hydraulic quadruped robots," in *The Fourteenth Scandinavian International Conference on Fluid Power*, 2015, pp. 20–22.

[53] H. Jeong and D. D. Lee, "Efficient learning of stand-up motion for humanoid robots with bilateral symmetry," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1544–1549.

[54] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[55] P. G. Adamczyk, S. H. Collins, and A. D. Kuo, "The advantages of a rolling foot in human walking," *Journal of experimental biology*, vol. 209, no. 20, pp. 3953–3963, 2006.

[56] Z. Li, C. Zhou, Q. Zhu, R. Xiong, N. Tsagarakis, and D. Caldwell, "Active control of under-actuated foot tilting for humanoid push recovery," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 977–982.

[57] Z. Li, B. Vanderborght, N. G. Tsagarakis, and D. G. Caldwell, "Human-like walking with straightened knees, toe-off and heel-strike for the humanoid robot icub," 2010.

[58] Z. Li, C. Zhou, N. Tsagarakis, and D. Caldwell, "Compliance control for stabilizing the humanoid on the changing slope based on terrain inclination estimation," *Autonomous Robots*, vol. 40, no. 6, pp. 955–971, 2016.

[59] S.-H. Hyon, R. Osu, and Y. Otaka, "Integration of multi-level postural balancing on humanoid robots," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 1549–1556.

[60] B. J. Stephens and C. G. Atkeson, "Dynamic balance force control for compliant humanoid robots," in *2010 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2010, pp. 1248–1255.

[61] Z. Li, B. Vanderborght, N. G. Tsagarakis, and D. G. Caldwell, "Fast bipedal walk using large strides by modulating hip posture and toe-heel motion," in *2010 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2010, pp. 13–18.

[62] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[63] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[64] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[65] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *arXiv preprint arXiv:1511.04143*, 2015.

[66] N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver, "Learning and transfer of modulated locomotor controllers," *arXiv preprint arXiv:1610.05182*, 2016.

[67] K. Yuan and Z. Li, "An improved formulation for model predictive control of legged robots for gait planning and feedback control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[68] C. Liu, A. G. Lonsberry, M. J. Nandor, M. L. Audu, A. J. Lonsberry, and R. D. Quinn, "Implementation of deep deterministic policy gradients for controlling dynamic bipedal walking," *Biomimetics*, vol. 4, no. 1, p. 28, 2019.

[69] B. Stephens, "Humanoid push recovery," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2007, pp. 589–595.

[70] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2006, pp. 137–142.

[71] T. Komura, A. Nagano, H. Leung, and Y. Shinagawa, "Simulating pathological gait using the enhanced linear inverted pendulum model," *IEEE Transactions on biomedical engineering*, vol. 52, no. 9, pp. 1502–1513, 2005.

[72] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online decision of foot placement using singular lq preview regulation," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2011, pp. 13–18.

[73] W. Hu, I. Chatzinikolaidis, K. Yuan, and Z. Li, "Comparison study of nonlinear optimization of step durations and foot placement for dynamic walking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 433–439.

[74] W. Han and R. Tedrake, "Feedback design for multi-contact push recovery via lmi approximation of the piecewise-affine quadratic regulator," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 842–849.

[75] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 31–38.

[76] Z. Li, N. G. Tsagarakis, and D. G. Caldwell, "Stabilizing humanoids on slopes using terrain inclination estimation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4124–4129.

[77] T. Koolen, M. Posa, and R. Tedrake, "Balance control using center of mass height variation: limitations imposed by unilateral contact," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 8–15.

[78] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater, *et al.*, "Valkyrie: Nasa's first bipedal humanoid robot," *Journal of Field Robotics*, vol. 32, no. 3, pp. 397–419, 2015.

[79] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th Symposium on Operating Systems Design and Implementation)*, 2016, pp. 265–283.

[80] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[81] P.-B. Wieber and C. Chevallereau, "Online adaptation of reference trajectories for the control of walking systems," *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 559–566, 2006.

[82] B. J. Stephens and C. G. Atkeson, "Push recovery by stepping for humanoid robots with force controlled joints," in *2010 10th IEEE-RAS International conference on humanoid robots*. IEEE, 2010, pp. 52–59.

[83] Z. Li, N. G. Tsagarakis, and D. G. Caldwell, "Walking trajectory generation for humanoid robots with compliant joints: Experimentation with coman humanoid," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 836–841.

[84] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine, "Variational inverse control with events: A general framework for data-driven reward definition," in *Advances in Neural Information Processing Systems*, 2018, pp. 8538–8547.

[85] M. Srouji, J. Zhang, and R. Salakhutdinov, "Structured control nets for deep reinforcement learning," *arXiv preprint arXiv:1802.08311*, 2018.

[86] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J.-B. Mouret, "A survey on policy search algorithms for learning robot controllers in a handful of trials," *IEEE Transactions on Robotics*, 2019.

[87] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[88] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez, "Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3066–3073.

[89] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–13, 2017.

[90] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.

[91] L. Righetti and A. J. Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1585–1590.

[92] A. Tavakoli, V. Levdik, R. Islam, and P. Kormushev, "Prioritizing starting states for reinforcement learning," *arXiv preprint arXiv:1811.11298*, 2018.

[93] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine, "Leave no trace: Learning to reset for safe and autonomous reinforcement learning," *arXiv preprint arXiv:1711.06782*, 2017.

[94] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," *arXiv preprint arXiv:1712.00378*, 2017.

[95] N. Ogihara and N. Yamazaki, "Generation of human bipedal locomotion by a bio-mimetic neuro-musculo-skeletal model," *Biological cybernetics*, vol. 84, no. 1, pp. 1–11, 2001.

[96] T. Mori, Y. Nakamura, M.-A. Sato, and S. Ishii, "Reinforcement learning for cpg-driven biped robot," in *AAAI*, vol. 4, 2004, pp. 623–630.

[97] V. Klemm, A. Morra, C. Salzmann, F. Tschopp, K. Bodie, L. Gulich, N. Küng, D. Mannhart, C. Pfister, M. Vierneisel, *et al.*, "Ascento: A two-wheeled jumping robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7515–7521.

[98] K. Araki, T. Miwa, H. Shigemune, S. Hashimoto, and H. Sawada, "Standing-up control of a fallen humanoid robot based on the ground-contacting state of the body," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 3292–3297.

[99] A. Radulescu, I. Havoutis, D. G. Caldwell, and C. Semini, "Whole-body trajectory optimization for non-periodic dynamic motions on quadrupedal systems," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5302–5307.

[100] J. A. Castano, C. Zhou, and N. Tsagarakis, "Design a fall recovery strategy for a wheel-legged quadruped robot using stability feature space," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019.

[101] J. Borràs, C. Mandery, and T. Asfour, "A whole-body support pose taxonomy for multi-contact humanoid robot motions," *Science Robotics*, vol. 2, no. 13, 2017.

[102] J. Borras and T. Asfour, "A whole-body pose taxonomy for loco-manipulation tasks," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1578–1585.

[103] R. Fabre, H. Gimbert, L. Gondry, L. Hofer, O. Ly, S. N'Guyen, G. Passault, and Q. Rouxel, "Rhoban football club–team description paper," *Humanoid KidSize League, Robocup 2015 Hefei*, 2015.

[104] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes, "Drecon: data-driven responsive control of physics-based characters," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–11, 2019.

[105] A. R. Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, "Setting up a reinforcement learning task with a real-world robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4635–4640.

[106] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," *arXiv preprint arXiv:1903.09537*, 2019.

[107] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *science*, vol. 315, no. 5817, pp. 1416–1420, 2007.

[108] T. Drew, J. Kalaska, and N. Krouchev, "Muscle synergies during locomotion in the cat: a model for motor cortex control," *The Journal of physiology*, vol. 586, no. 5, pp. 1239–1245, 2008.

[109] M. Mischiati, H.-T. Lin, P. Herold, E. Imler, R. Olberg, and A. Leonardo, "Internal models direct dragonfly interception steering," *Nature*, vol. 517, no. 7534, pp. 333–338, 2015.

[110] S. K. Karadimas, K. Satkunendrarajah, A. M. Laliberte, D. Ringuette, I. Weisspapir, L. Li, S. Gosgnach, and M. G. Fehlings, "Sensory cortical control of movement," *Nature neuroscience*, vol. 23, no. 1, pp. 75–84, 2020.

[111] H. Markram, "The blue brain project," *Nature Reviews Neuroscience*, vol. 7, no. 2, pp. 153–160, 2006.

[112] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ieee, 2013, pp. 194–201.

[113] "DARPA robotics challenge (DRC)," https://www.darpa.mil/program/darpa-robotics-challenge, accessed: 2020-7-16.

[114] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, *et al.*, "No falls, no resets: Reliable humanoid behavior in the darpa robotics challenge," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 623–630.

[115] N. Bernstein's, "The co-ordination and regulation of movements," 1967.

[116] M. L. Latash, "Stages in learning motor synergies: A view based on the equilibrium-point hypothesis," *Human movement science*, vol. 29, no. 5, pp. 642–654, 2010.

[117] J. Ramos and S. Kim, "Dynamic locomotion synchronization of bipedal robot and human operator via bilateral feedback teleoperation," *Science Robotics*, vol. 4, no. 35, 2019.

[118] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "A sparse model predictive control formulation for walking motion generation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2292–2299.

[119] H.-W. Park, P. M. Wensing, S. Kim, *et al.*, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," 2015.

[120] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[121] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, 2015.

[122] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[123] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[124] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[125] I. Chatzinikolaidis, Y. You, and Z. Li, "Contact-implicit trajectory optimization using an analytically solvable contact model for locomotion on variable ground," *IEEE Robotics and Automation Letters*, 2020.

[126] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[127] E. O. Neftci and B. B. Averbeck, "Reinforcement learning in artificial and biological systems," *Nature Machine Intelligence*, vol. 1, no. 3, pp. 133–143, 2019.

[128] K. Bouyarmane, S. Caron, A. Escande, A. Kheddar, A. Goswami, and P. Vadakkepat, "Multi-contact planning and control," in *Humanoid Robotics: A Reference*. Springer, 2019, pp. 1763–1804.

[129] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[130] M. M. A. Posa, "Optimization for control and planning of multi-contact dynamic motion," Ph.D. dissertation, Massachusetts Institute of Technology, 2017.

[131] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[132] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[133] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, "Using deep reinforcement learning to learn high-level policies on the atrias biped," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 263–269.

[134] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning*, 2020, pp. 317–329.

[135] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete event dynamic systems*, vol. 13, no. 1-2, pp. 41–77, 2003.

[136] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," *arXiv preprint arXiv:1710.09767*, 2017.

[137] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine, "Latent space policies for hierarchical reinforcement learning," *arXiv preprint arXiv:1804.02808*, 2018.

[138] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[139] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[140] X. Chang, T. M. Hospedales, and T. Xiang, "Multi-level factorisation net for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2109–2118.

[141] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine, "Mcp: Learning composable hierarchical control with multiplicative compositional policies," in *Advances in Neural Information Processing Systems*, 2019, pp. 3686–3697.

[142] A. G. Feldman, V. Goussev, A. Sangole, and M. F. Levin, "Threshold position control and the principle of minimal interaction in motor actions," *Progress in brain research*, vol. 165, pp. 267–281, 2007.

[143] E. Bizzi, N. Hogan, F. Mussa-valdi, and S. Giszter, "Does the nervous system use equilibrium-point control to guide single and multiple joint," *Behavioral and brain sciences*, vol. 15, pp. 603–613, 1992.

[144] F. L. Moro, N. G. Tsagarakis, and D. G. Caldwell, "A human-like walking for the compliant humanoid coman based on com trajectory reconstruction from kinematic motion primitives," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2011, pp. 364–370.

[145] A. T. Sprowitz, A. Tuleu, A. J. Ijspeert, *et al.*, "Kinematic primitives for walking and trotting gaits of a quadruped robot with compliant legs," *Frontiers in computational neuroscience*, vol. 8, p. 27, 2014.

[146] S. Starke, H. Zhang, T. Komura, and J. Saito, "Neural state machine for character-scene interactions," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–14, 2019.

[147] "Jueying® | DeepRobotics," http://www.deeprobotics.cn/default/details, accessed: 2019-12-13.

[148] N. Hogan and D. Sternad, "Dynamic primitives of motor behavior," *Biological cybernetics*, vol. 106, no. 11-12, pp. 727–739, 2012.

[149] E. Spyrakos-Papastavridis, N. Kashiri, J. Lee, N. G. Tsagarakis, and D. G. Caldwell, "Online impedance parameter tuning for compliant biped balancing," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 210–216.

[150] J. W. Hurst and A. A. Rizzi, "Series compliance for an efficient running gait," *IEEE Robotics & Automation Magazine*, vol. 15, no. 3, pp. 42–51, 2008.

[151] Y. Hashiguchi, K. Takaoka, and M. Kanemaru, "The development of a practical dexterous assembly robot system without the use of force sensor," in *Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century.(Cat. No. 01TH8560)*. IEEE, 2001, pp. 470–475.

[152] Y. Zhao, N. Paine, S. J. Jorgensen, and L. Sentis, "Impedance control and performance measure of series elastic actuators," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 3, pp. 2817–2827, 2017.

[153] H. V. Hasselt, "Double q-learning," in *Advances in neural information processing systems*, 2010, pp. 2613–2621.

[154] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[155] C. Gilbert, "Visual control of cursorial prey pursuit by tiger beetles (cicindelidae)," *Journal of Comparative Physiology A*, vol. 181, no. 3, pp. 217–230, 1997.

[156] J. Camhi and E. Johnson, "High-frequency steering maneuvers mediated by tactile cues: antennal wall-following in the cockroach," *Journal of Experimental Biology*, vol. 202, no. 5, pp. 631–643, 1999.

[157] B. Wu, I. Akinola, J. Varley, and P. K. Allen, "Mat: Multi-fingered adaptive tactile grasping via deep reinforcement learning," in *Conference on Robot Learning*, 2020, pp. 142–161.

[158] S. H. Huang, M. Zambelli, J. Kay, M. F. Martins, Y. Tassa, P. M. Pilarski, and R. Hadsell, "Learning gentle object manipulation with curiosity-driven deep reinforcement learning," *arXiv preprint arXiv:1903.08542*, 2019.

[159] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," *arXiv preprint arXiv:1907.03613*, 2019.

[160] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2950–2957, 2020.

[161] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," *arXiv preprint arXiv:2002.08550*, 2020.

[162] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[163] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.

[164] Y. Hu and G. Montana, "Skill transfer in deep reinforcement learning under morphological heterogeneity," *arXiv preprint arXiv:1908.05265*, 2019.

[165] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[166] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[167] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "Sfv: Reinforcement learning of physical skills from videos," *ACM Trans. Graph.*, vol. 37, no. 6, Nov. 2018.