THE UNIVERSITY
of EDINBURGH

Embodied Agents in Real-World Robots

Integrated Control and Machine Learning for Intelligent Behaviours

Kai Yuan

Doctor of Philosophy

The University of Edinburgh

School of Informatics

2022

Lay Summary

Undoubtedly, Robotics will benefit our society and help dealing with urging societal and health issues, such as healthcare, nuclear decommission, space exploration, disaster response, and agriculture. The ability to *intelligently* and *autonomously* locomote and manipulate in these environments plays a crucial role in the success of the deployment of robots for these types of dangerous and expensive missions.

Locomotion: the ability to move; manipulation: using robotic arms to interact with the environment

In the past 50 years, the field of robotics has seen rapid developments: in a controlled environment, robots can run, backflip, manipulate complex objects, and grasp in cluttered environments. However, the algorithms enabling these motions are designed based on a specific task, which means that if the robot encounters a new scenario that it is not programmed for, it has difficulties to handle novel situations. This greatly reduces the real world applications.

In this thesis, we combined control with learning and developed control architectures and algorithms that allow robots to move and behave like biological systems. As a result, the robots acted in a more general fashion and learned new skills from a set of existing one, greatly improving the applicability of robots in unstructured, unknown environments.

To this end, we combined control with Deep Reinforcement Learning to design control systems that synthesis the skills of multiple experts for fall-resilient locomotion and dual arm cooperation. Furthermore, we investigated autonomous robotic control from a Neuroscience perspective and proposed a control framework that mimics the deep temopral architecture of human motor control that can be used as general control framework for autonomous robotics tasks.

As outcome of this research, the proposed algorithms in this thesis that achieve intelligent, dynamic, and robust motions are realised on real world robots, such as NASA's humanoid Valkyrie, ANYbotics' quadruped ANYmal, DeepRobotics' quadruped Jueying, and dual arm manipulators (Franka Panda).

Please visit the link https://drive.google.com/drive/folders/1vJUIGlqpbponzBUr1YhaM-Fj6LoMVNKv?usp=sharing for the supplementary videos of this thesis.

Abstract

The central contribution of this thesis is providing a reliable framework and algorithms to make robots move as versatile and reliable as biological systems. To this end, this work proposes a hierarchical control framework that allows the combination of *classical control on the lower levels* for control of the robot's actuators achieving stability and balance control and *machine learning on the higher-levels* for decision making and planning. Using Machine Learning for the decision making and planning enables the robot to behave more intelligently, dynamically, and animal-like, while the control algorithms for the lower levels maintain the robustness and stability properties of classical control.

Combining control for stability and machine learning for intelligence.

In particular, this thesis presents *contributions in six areas of robotic motion control*: (1) biologically motivated implicit hierarchical generative model for autonomous robot operations, (2) multi-expert learning and skill synthesis, (3) improved formulation for Model-Predictive Control (MPC) of legged robots, (4) rapid robot policy development and deployment through fast sample collection for Deep Reinforcement Learning (DRL), (5) reverse-engineering AI policies into an equivalent, safe, transparent, and certifiable controllers, and (6) automatic parameter tuning.

First, we draw inspiration from human motor control and insights in Neuroscience and propose an implicit hierarchical generative model for robot control that mimics the deep temporal architecture of human motor control and show that this can be used to fully autonomously learn to complete a complex task of object retrieval, delivery, and navigation.

Implicit hierachical generative model based on neuroscientific insights from human motor control.

Using the DRL policies trained in the previous works', we propose a Multi-Expert Learning Architecture (MELA) that allows using multiple experts to train and synthesise new skills. We show that MELA can be used to learn animal-like, dynamic, and adaptive behaviours on the quadruped robot Jueying.

Multi-Expert Learning Architecture generates adaptive skills from a group of representative expert skills.

We extend MELA, into a more general Multi-Expert Synthesis (MES) framework, where we propose general guidelines for MES. To this end, we propose an automatic state selection procedure, and identify and solve common issues in multi-expert systems. Through MES, we achieve fall-resilient locomotion on the quadruped ANYmal and dual-arm cooperation for grasping ungraspable object on bi-manual robot setup using Franka Panda.

Multi-Expert Synthesis enables learning fall-resilient locomotion and dual-arm cooperative manipulation.

We propose a numerically stable, Linear-Inverted Pendulum Model (LIPM) based formulation for MPC, and show its robustness properties on the task of legged locomotion for the humanoid Valkyrie. Furthermore, we show that the proposed formulation is more robust to external disturbances than other LIPM formulations for MPC.

Numerically stable, robust formulation for locomotion on NASA's Valkyrie.

In the domain of DRL, we propose a fast sample collection procedure on consumer-grade computers through parallelisation that enables the trainining and deployment of policies like trotting on quadruped ANYmal and balancing on humanoid Valkyrie within an hour.

Fast sample collection for Deep Reinforcement Learning enables training trotting on ANYmal and balancing on Valkyrie within an hour.

Reverse-engineering an opaque AI policy into a transparent, certifiable controller.

Next, we use the Artificial Intelligence (AI) policy trained through DRL as basis for a reverse-engineering framework. For the task of humanoid push recovery on Valkyrie, we show how an opaque AI policy can be used to obtain a transparent, certifiable controller with same properties as the AI policy. We show that ankle, hip, toe, and stepping strategies emerge from the reverse-engineered controller in a unified manner without the need of explicit switching.

Alternating Bayesian Optimisation: automatic parameter tuning of high-dimensional parameters for whole-body control of Valkyrie.

Lastly, we propose an algorithm called Alternating Bayesian Optimisation (ABO) to automatically tune the high-dimensional parameters for whole-body control of Valkyrie. Contrary to classical Bayesian Optimisation, which scales only up to 10 dimensions, we show that ABO can tune up to 36 parameters for whole-body control and up to 60 dimensions on the global optimisation benchmark COCO.

Acknowledgements

First and foremost, I am extremely grateful and fortunate to have been supervised by Dr. Zhibin (Alex) Li. I would like to thank him for all his support, advise, and continuous mentorship throughout my PhD. As an outstanding motivator and leader, he created a nourishing and creative environment giving me the freedom to pursue my research interests, and to grow both professionally and personally.

I would like to express my sincerest gratitude to my family and friends. This accomplishment would have been impossible without your love, support, and encouragement.

I would like to thank my extraordinary friends and colleagues from the Institute of Perception, Action and Behaviour (IPAB), and especially the members of the Advanced Intelligent Robotics (AIR) Lab for the past four joyous years, their mentorship and camaraderie, engaging lunch and coffee breaks, all the fruitful discussions, and late-night lab sessions.

I would also like to thank all my collaborators, former mentors and colleagues for their help, advise, and valuable feedback. Special thanks to Professor Mingguo Zhao for giving me the opportunity and guidance to start my journey in the wonderful world of Robotics.

Last but not least, my gratitude goes to the Edinburgh Centre for Robotics (ECR) and the Engineering and Physical Sciences Research Council (EPSRC) for funding my research.

Dedication

This thesis is dedicated to my family and friends. Thank you for all your unconditional love, tremendous understanding, continuous support, and encouragement. This journey would have not been possible without you.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Kai Yuan

Preface

This thesis is written as cumulative dissertation, where every chapter in the main body comprises one publication.

After an introduction in Chapter 1, where we introduce and motivate this work, the main contributions of this thesis are outlined in Chapter 2. In Chapter 2, we summarise the contributions of all publications, put the publications in context with each other and this thesis, and present their applications. Lastly, a list of publications is provided.

The main part of this thesis present the contributions in the domain of intelligent, multi-expert control for autonomous robots (Chapters 3 – 5), which contain edited and reformatted versions of the publications [1, 2, 3].

We draw conclusions and provide an outlook in Chapter 6.

In the Appendix A – E, our contributions in the domain of control theory and learning for robotics, and methods enabling multi-expert learning are presented. These Appendix chapters constitute an edited version of the publications [4, 5, 6, 7].

Contents

1 Introduction

Undoubtedly, Robotics will benefit our society and help dealing with urging societal and health issues, such as healthcare [8], nuclear decommission [9], space exploration [10], disaster response [11], and agriculture [12]. The ability to *intelligently* locomote and manipulate in these environments plays a crucial role in the deployment of robots for dangerous and expensive missions.

Applications of manipulators and legged robots.

However, (artificial) intelligence remains one of the grand challenges in robotics [13]. In the past 50 years, the field of robotics has seen rapid developments: in a controlled environment, robots can run [14], backflip [15], manipulate complex objects [16], and grasp in cluttered environments [17]. The algorithms enabling these motions are designed based on a specific task, which means that if the robot encounters a new scenario that it is not programmed for, it has difficulties to handle novel situations. This greatly reduces the real world applications.

Examples of state of the art robot capabilities and their current limitations.

1.1 Motivation and Objective

Figure 1.1: Robots on which the algorithms of this thesis were applied on: humanoid Valkyrie, quadrupeds Jueying and ANYmal, and dual arm manipulators Panda.

To improve the robustness and generality of robot algorithms, and thus the capabilities of robots, this thesis explores ways to combine classical optimal control methods with gradient-free methods from Machine Learning, such as Deep Reinforcement Learning [18] and Bayesian Optimisation [19]. This thesis aims to contribute in the domain of robotic mobility and manipulation by achieving motion intelligence that is comparable to biological systems, such as humans and animals, in terms of versatility, robustness, and aesthetics.

Goal of this thesis: contributing towards achieving motion intelligence comparable to biological systems.

As outcome of this research, we show how the algorithms proposed in this thesis achieve intelligent, dynamic, and robust motions, and demonstrate their effectiveness on real world robots (Fig. 1.1), such as NASA's humanoid Valkyrie [10], ANYbotics' quadruped ANYmal [20], DeepRobotics' quadruped Jueying [21], and dual arm manipulators (Franka Panda) [22].

1.2 Structure of this Thesis

In the main part of this thesis, we propose three ways on how to integrate learning and control to achieve motion intelligence in robots – the main aim of this thesis. The components required – control strategies and learning policies – are outlined in the Appendix A - E.

In the following subsections, a summary of the integrated learning and control algorithms for intelligent robot motion control is given and a description is given how and in what context the individual, proposed methods can be applied.

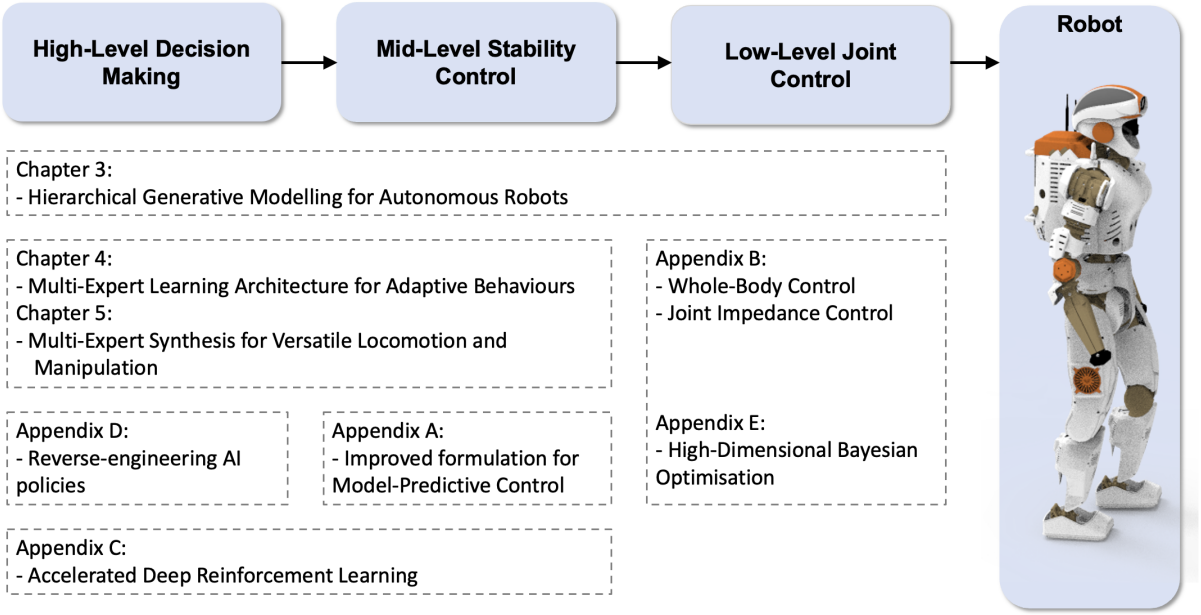


Figure 1.2: Hierarchical Control Architecture inspired by the deep temporal architecture in human motor control. The components and contributions required for the successful deployment of the hierarchical control architecture on robots are described in the Chapters 3 - 5 and Appendix A - E. . Each Chapter corresponds to a publication (see Chapter 2).

A visual summary of the contributions of this work are depicted in Figure 1.2 and discussed in-depth in Chapter 2. For intelligent behaviour of robots, the control architecture is structured in a hierarchical manner. This control architecture – as discussed in Chapter 3 – mimics the deep temporal architecture of human motor control and consists of three levels described in the following:

(1) the highest level performs decision making and is responsible for setting goals or target commands. In Chapter D, an AI policy – obtained through Deep Reinforcement Learning – is reverse-engineered into a transparent controller, which is used as high-level policy to command the robot where to place its Centre of Mass and Feet. In Chapter 4 and 5, a Multi-Expert Learning Architecture is presented that synthesis multiple experts for adaptive behaviours, versatile locomotion and manipulation. The policy is designed to act as both high-level decision making and as mid-level stability controller. Lastly, a hierarchical generative model is presented in Chapter 3. The proposed generative model

enables the robot to fully autonomously complete complex loco-manipulation tasks. Additionally, Deep Reinforcement Learning as presented in Appendix C merges the functionality of both high-level decision making and mid-level stability control.

(2) the mid-level policy aims to track the high-level commands while stabilising the system. Two methods are used for stability control: Model-Predictive Control and Deep Reinforcement Learning (DRL). In Appendix A an improved formulation for Model-Predictive Control is presented that achieves both Gait Planning (tracking high-level foot step commands) and Feedback Control (maintaining balance under uncertainties). Supplementing the control method, Appendix C introduces the concept of DRL and presents a method for accelerated Deep Reinforcement Learning.

(3) the low-level controller performs joint control and directly deploy the mid-levels commands on the robot's actuators. In Appendix B the concepts of Whole-Body Control and Joint Impedance Control are introduced. Depending on the use case calculating appropriate torques for the actuators might be done in a coupled, holistic manner through Whole-Body Control or on an individual joint basis through Joint Impedance Control. Lastly, Appendix E presents a method to automatically tune the control parameters used for low-level joint control.

Integrating Learning and Control into Hierachical Generative Model

Figure 1.3: Fall-resilient locomotion on Jueying using proposed Multi-Expert Learning Architecture (MELA).

Stable and intelligent behaviour by combining learning and control.

In the Chapters 3 - 5, we present three approaches that combine the control and learning methods with the aim to achieve motion intelligence. Combining control with learning allows more intelligent behaviours from learning while maintaining the safety and transparency aspects of control.

In particular, we present a neuroscientifically inspired implicit hierarchical generative model for autonomous robot operations in Chapter 3. Further, we propose a multi-expert synthesis framework in Chapters 4 and 5. Finally, we present a reverse-engineering framework in Appendix D that yields a transparent, safe controller from a trained DRL policy [6]

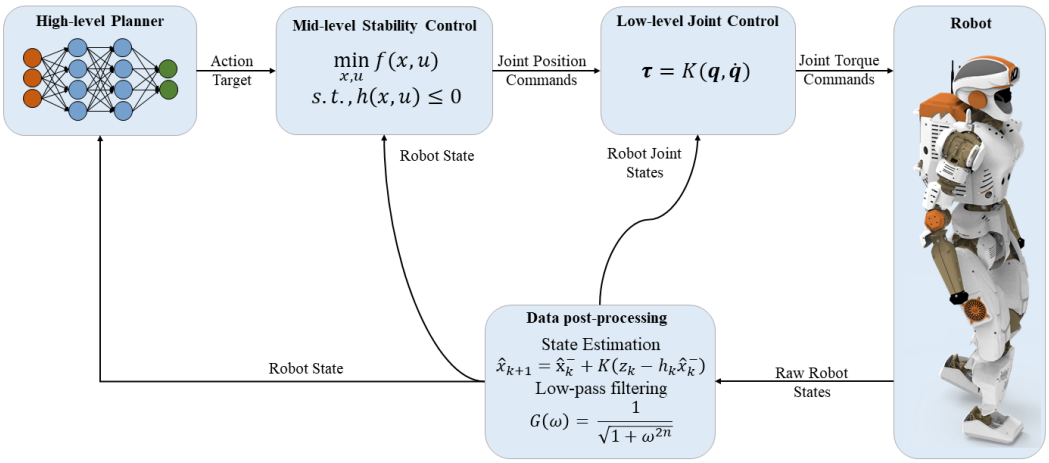


Figure 1.4: Hierarchical control structure for robots.

Control for Robotics

To control complex dynamical systems, such as robots (Fig. 1.1), multiple layers of control are required. The hierarchical control structure depicted in Fig. 1.4 is used to deploy versatile, robust, and stable motions on the real robot.

The core idea lies in a high-level planner computing a desired reference trajectory that is tracked by a mid-level stability controller while guaranteeing stability. The mid-level stability controller provides the low-level joint controller with target values, which are translated into joint torques on the robot.

In this work, two concepts of controls are presented to realise such a hierarchical control structure: proactive Model-Predictive Control (Appendix A) for mid-level stability control, and reactive low-level joint control (Appendix B). While Model-Predictive Control enables both planning and feedback control for legged locomotion, reactive low-level joint control is used to realise joint torques on the actual robot while maintaining stability of the system and respecting physical constraints of the robots.

Learning for Motion Intelligence

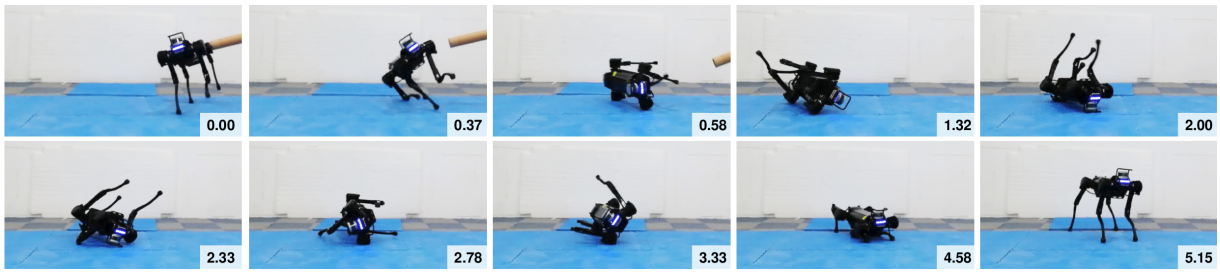


Figure 1.5: Jueying performing learned Fall Recovery motion after being pushed over by external disturbance.

Deep Reinforcement Learning and Bayesian Optimisation for learning versatile and intelligent motions.

To provide the robot with additional intelligence, two common approaches for learning are presented: Deep Reinforcement Learning

(Appendix C) and Bayesian Optimisation (Appendix E), which is used to automatically tune parameters of the whole-body controller.

Deep Reinforcement Learning (DRL) extends the notion of Reinforcement Learning by using Deep Neural Networks to represent the policy and value function. In this thesis, we will use DRL to learn intelligent robotic behaviours. The learned policies are directly deployed on the robot (Appendix C), reverse-engineered into a transparent controller with the same characteristics in Appendix D, and multi-expert learning approaches (Chapters 4, 5, and 3).

Bayesian Optimisation is a sample efficient black box optimisation procedure, and is used to tune parameters of control frameworks.

1.3 Related Work

This section aims to provide the reader with an overview of this thesis and to put this thesis in the context of control and machine learning. A more detailed related work description in the fields of control for robotics, machine learning for robotics, and multi-expert learning can be found in the respective chapters.

Control for Robotics

Control for robotics can be categorised in predictive and reactive control. Following, we present Model-Predictive Control (MPC) that predicts future states and plans a trajectory according to these predictions based on the current state and under consideration of constraints. Furthermore, we introduce a reactive control method: Whole-Body Quadratic Programming.

Use of both predictive (Model-Predictive Control) and reactive control (Whole-Body Quadratic Programming) for robots.

Model-Predictive Control

MPC performs two roles: open-loop planning and closed-loop feedback control. MPC achieves closed-loop control by continuously solving the feed-forward optimization under constraints using the feedback of current states and constraints. The main challenge of MPC lies in solving the feed-forward optimization in real-time as the close-loop often runs multiple times per second. To this end, simplified models, such as the Linear Inverted Pendulum (LIP) [23] has been proposed. More recently, non-linear models have been used to better capture the whole-body dynamics of robots [24, 25].

For further details, see Appendix A.

MPC has been extensively used in the domain of robotic locomotion of floating based systems. For example, improved disturbance rejection was achieved by constraining the Zero-Moment Point [26]. Furthermore MPC was used for automatic foot placement [27], Push Recovery [28], and Capture Point (CP) tracking [29, 30].

In the past decades, the importance of MPC for dynamic locomotion has increased. First an unconstrained MPC scheme was proposed in [31] for walking pattern generation of bipedal humanoids. Improved disturbance rejection was then achieved by constraining the ZMP [26].

Furthermore MPC was used for automatic foot placement [27], Push Recovery [28], and Capture Point (CP) tracking [29, 30].

In contrast to prior MPC methods that use an LIP model, we propose an MPC framework in this thesis (Appendix A) that particularly distinguishes the external perturbation in our formulation making it real-world applicable, and also has dual-uses for Motion Planning and Feedback Control. Our formulation is able to generate a stable COM motion given Centre of Pressure (COP) or CP references while satisfying the ZMP constraints. For further details regarding the advantages and comparison of the various LIP model-based MPC formulations, see Appendix A.

Whole-Body Quadratic Programming

For further details, see Chapter B.

Whole-Body Quadratic Programming (WBQP) is an Inverse Dynamics method that calculates appropriate torques to account for gravity and Corriolis forces while tracking a given reference trajectory.

WBQP have been widely use for robotics tasks, such as locomotion and whole-body control [32, 33]. In combination with a Trajectory Planner [34, 35], WBQP is used to robustly realise planned motions [36] under constraints. This paradigm of using optimisation to first plan and then robustly track the planned trajectory has found wide application in the control of floating-based robots [11, 37, 38, 24].

However, it shall be noted that while whole-body control guarantees stability, their reliance on the model and constraints make whole-body control only suitable in well modelled environments. Stability and performance issues arise at situations like contact switches, deviating model parameters, or deviation from assumptions made for linearisation [39].

Machine Learning for Robotics

Learning policies from self-generated data contrary to using labelled data.

The learning aspect for Robotics in this work focuses on the Reinforcement Learning paradigm [40], i.e., learning a policy from data generated by rolling out a policy instead of using labelled data as in supervised learning [41]. In the following, we highlight related work in Bayesian Optimisation, Deep Reinforcement Learning, and Multi-Expert Learning.

Bayesian Optimisation

For further details, please see Chapter E.

Bayesian Optimization (BO) [19], a derivative-free, sample-efficient optimization algorithm, is suitable to find global optima for black-box optimization functions [42], and is widely used for hyper-parameter tuning [43]. In robotics, BO has been used to find gait parameters on real, physical hardware in [44] (5 dim./9 dim.), [45] (7 dim.), [46] (8 dim.), [47] (15 dim.). Additionally, domain knowledge has been applied to find suitable kernels for improving sample efficiency on hardware experiments [44, 48]. However, the dimensionality in these works was

relatively low, and hence a direct implementation of BO performed well.

Despite its benefits, the capability of BO in finding global optima is limited by the high dimensionality, as the search space grows exponentially. Dimensionality reduction approaches, such as Principal-Component Analysis [49] or Partial Least Squares [50], would fail when the evaluation of the objective function is expensive, or the data is insufficient to find correlations in parameters. An approach to combine BO with LQR was suggested in [51], in which, rather than tuning the full state-feedback controller, only the lower dimensional LQR weights were tuned via BO. For humanoid Whole-Body Control, a Trial-and-Error learning algorithm in [52] dealt with model inaccuracies by learning repulsors to alter reference motions that prevents an unstable configuration of state space.

Recently, high-dimensional BO methods have been proposed: in [53] only a subset is optimized over by randomly dropping out parts of the parameters space; in [54] a local search is performed near a priorly given location. However, the work in [53] does not consider the correlation between the parameters and randomly drops out parameters, while the solution in [54] relies on prior knowledge of the optimum which is mostly unavailable without pre-tuning.

To overcome these limitations, we propose a novel approach [7] that applies domain knowledge for partitioning the parameter space, and is able to find an optimum without assuming the location of the optimum (see Appendix E). We adopt the idea of Alternating Optimization [55] and iteratively optimize one partitioned parameter sub-space at a time while keeping the rest of the parameter space fixed.

Our proposed algorithm has similarities with Coordinate Descent approaches [56]. However, instead of a sequential search along the coordinates until convergence, our proposed method searches in sub-hyperboxes of the parameter space and is independent on the convergence of previous iterations. In combination with BO, a global optimization algorithm, the alternating nature reduces the risks of local minima and allows parallelization, which are the two characteristics that are not provided by Coordinate Descent methods.

Deep Reinforcement Learning

Recent algorithmic advances in Reinforcement Learning [18] incorporated Deep Neural Networks (DNN) as function approximators [57]. Consequently, continuous control tasks as prevalent in robotics can be solved by directly optimising the actor through policy gradient methods [58].

For further details, please see Chapter C.

Deep Reinforcement Learning (DRL) algorithms can be categorised into two families: on-policy and off-policy algorithms. While on-policy algorithms evaluate and improve the same policy from which the sampled data originates from, off-policy algorithms do not require the samples to originate from the same policy as the trained, target policy. Current state of the art on-policy algorithms for continuous control include Trust Region Policy Optimization [59], Proximal Policy Optimization

[60], and Asynchronous Advantage Actor-Critic [61]. For off-policy algorithms, Soft-Actor Critic [62], Deep Deterministic Policy Gradient (DDPG) [63], and Twin Delayed DDPG [64] have seen successes in controlling robotic systems.

These algorithms have been applied in robotics to imitate animal locomotion [65], achieve fall recovery [66], yield human-like locomotion [67], perform grasping in cluttered environments [68], perform dexterous manipulation [69], and achieve remarkable results in robotic benchmarks [70, 71].

While DRL finds powerful solutions that maximise the user-defined cumulative reward, its requirement on large amounts of data and necessity to retrain policies from scratch even for similar tasks make scaling the solution impractical. This motivates the application of multi-expert learning, where multiple experts are used to both acquire new skills and used in a unified manner.

Multi-Expert Learning

For further details, please see Chapter 5.

The essential principle of multi-expert learning lies in designing a hierarchical control architecture where non-relevant information is coordinated and hidden across the layers. Thus, experts focus solely on performing their particular skills, while the high-level policy is responsible for drawing from the experts' skills and completing the task.

This idea originates from Hierarchical Reinforcement Learning (HRL) [72]. In HRL, for discrete and tabular cases, the concepts of temporal [73] and state abstraction [74] are used to determine the information that the components receive. Despite the advantages of encoding non-relevant information across layers, composing and synthesising expert knowledge is not possible in the standard HRL framework since only one expert can be selected at a time. This problem is addressed by learning continuous latent variables that blend the experts in a latent space [75, 76].

The traditional approach for the latent space blending is the Mixture of Experts (MoE), where the outputs of individual experts (actions) specialized on sub-problems are combined by a gating function [77]. The core idea of MoE, combining the outputs of experts via a gating network, has been adapted in the areas of machine translation [78], computer vision [79], robotics [80], Reinforcement Learning (RL) [81], and Computer Graphics [82].

With the advent of Deep Learning and Deep Neural Network, DRL has been used to extend the discrete tabular HRL concepts into the continuous control domain. In the context of multi-expert learning and multi policy composition, this led to frameworks where low-level experts encode motion primitives, control fragments, or skills, while a high-level policy selects the expert [83, 84, 85]. Alternative to learning a high-level policy to select appropriate motion primitives, COCoMoPL [86] proposes a framework, where near-optimal motion primitives are learned and synthesised into a motion as weighted combination of these motion primitives.

In [87], a method is proposed that does not synthesise expert skills into a unified policy, but rather expands the existing skill by decomposing the task into simpler subtask and training a local policy for the subtasks.

A modular framework is proposed in [88], which learns transition policies that connect primitive skills to complete sequential tasks. An extended work in [89] uses the modular framework to learn to coordinate the learned primitive skills for task completion. In [90] a method is proposed that learns task-agnostic skills to use their composite to solve new tasks in an HRL fashion. Task completion is accomplished by the task-agnostic skills performing commands provided by the high-level command.

Contributions 2

Despite remarkable advancements in the fields of Control and Machine Learning for Robotics as mentioned in Chapter 1.3, robust and intelligent behaviour of robots remain a challenge. This work aims to contribute towards more intelligent robotic behaviours.

This chapter outlines the contributions of this thesis. In the following, we first summarise the contributions. Second, a detailed outline and context of the contributions of every publication is provided. Further, the applications of each contribution are detailed to serve as a guideline indicating which contribution and newly proposed method is suited for which application. The chapter concludes with a full list of publications.

2.1 Summary of Contributions

The central contribution of this work is providing a reliable framework and algorithms to make robots move as versatile and reliable as biological systems. This work proposes a hierarchical control framework (see Chapter 3) that allows the combination of classical control on the lower levels to control the actuators of the robot achieving stability and balance control, and Machine Learning on the higher-levels of control for decision making and planning. Using Machine Learning for the decision making and planning enables the robot to behave more intelligent, dynamically and animal-like, while the control algorithms for the lower levels maintain the robustness and stability properties of classical control.

Machine learning for intelligent behaviour; control for stable and robust actuation of the robot.

In the main body of this thesis, we propose methods that integrate Machine Learning approaches into a Control framework (Chapters 3 – 5). In the Appendix, methods for Control for Robotics (Appendix A – B) and Learning for Motion Intelligence (Appendix C – E) are presented. Every publication is summarised as chapter in this thesis. Following, we summarise the contributions of every publication.

Combining Learning & Control

First, we present an implicit hierarchical generative model that mimics the deep-temporal archictecture of human motor control and can learn autonomous task completions (Chapter 3). We show how the proposed hierarchical generative model enables learning to autonomously completey a complex task requiring locomotion, manipulation, and grasping. In particular, the humanoid robot can retrieve and deliver a box, open and walk through a door to reach the final destination. Second, we develop a Multi-Expert Learning Architecture (MELA) that allows learning new skills from a set of existing ones (Chapter 4). We further analyse Multi-Expert Synthesis (MES) in Chapter 5, where we highlight guidelines for MES and propose an automatic state selection process.

Neuroscientifically inspired control architecture for fully autonomous task completion.

Multi-Expert Learning Architecture and Multi-Expert Synthesis.

Paper P1: Hierarchical generative modelling for autonomous robots.

The contributions of the hierarchical generative model in **Paper P1** are:

1. Considering autonomous robot control from a neuroscientific perspective: deriving core principles of human motor control to robotics.
2. Proposing a hierarchical generative model that mimics deep temporal architecture of human motor control for autonomous robot control.
3. Framework on how to realise the hierarchical generative model in robotics.
4. Validation and demonstration of robustness of hierarchical generative model on a loco-manipulation tasks for humanoid Valkyrie.

Paper P2: Multi-expert learning of adaptive legged locomotion.

The contributions of MELA in **Paper P2** are:

1 Propose a multi-expert learning architecture (MELA) that contains multiple expert neural networks, each with a unique motor skill, and a gating neural network (GNN) that fuses expert networks dynamically into a more versatile and adaptive neural network.
2. Show that MELA (i) generates multiple distinctive skills effectively, (ii) diversifies expert skills using co-training, and (iii) synthesizes multiskill policies with adaptive behaviors in unseen situations.
3. Demonstrate a breadth of adaptive behaviors for contact-rich locomotion on a real robot and various extreme test scenarios in a physics simulation.

Paper P3: Multi-Expert Synthesis for Versatile Locomotion and Manipulation Skills.

The summarised contributions of MES in **Paper P3** are:

1. A *systematic design for Multi-Expert Synthesis (MES)* to enable efficient learning of skill coordination and effective synthesis of multiple experts during tasks.
2. Formulation of a *state selection algorithm* that uses the learned state value function to quantitatively identify essential states and rule out task-irrelevant variables.
3. A proposed *explicit enforcement of diversity among multiple experts* by maximising their discriminability.
4. Applicability of the proposed methods for MES *achieve motor skill synthesis in both robot locomotion and manipulation.*

Control for Robotics

Robust robot control through predictive and reactive control.

In Appendix A – B we introduce control methods to guarantee the stability and balance of legged robots.

Please note, that for some cases, a joint impedance controller suffices for stable control of the robot, which will also be discussed in Appendix B.

For any robotic system, stability - the ability to perform motions without diverging states - is the most crucial aspect. For stability and robustness of the robotic system, we will present a combination of Model-Predictive Control (MPC) for gait planning and feedback control (Appendix A), and Inverse Dynamics based Whole-Body Control (Appendix B) realising the generated trajectories.

Paper P4: An Improved Formulation for Model Predictive Control of Legged Robots for Gait Planning and Feedback Control.

Summarised the contributions of **Paper P4** in the fields of MPC are:

1. Analysis of numerical and physical issues in existing MPC frameworks;
2. A proposed formulation that solves the numerical and physical problems analyzed above;
3. Integration of the proposed MPC with whole-body control for real-world application;
4. Benchmarking of the proposed MPC framework against existing ones.

Learning for Motion Intelligence

Motion intelligence is achieved through Deep Reinforcement Learning and Bayesian Optimisation.

Reverse-engineering an AI policy into a transparent controller.

Appendix C – E, we outline two learning methods for Robotics: Deep Reinforcement Learning (Chapter C) and Bayesian Optimisation (BO) (Chapter E). Further, we propose a method to reverse-engineer the learned, opaque Deep Reinforcement Learning policy into a transparent controller (Chapter D).

Through Deep Reinforcement Learning (Chapter C), a policy is autonomously learned that exhibits intelligent motions and deployed on real robot. Automatic parameter tuning is conducted through Bayesian Optimisation, where optimal parameters for any control or learning framework are found.

Paper P5: Fast Sample Collection Through Massive Parallelisation For Accelerated Deep Reinforcement Learning Development.

The summarised contributions of **Paper P5** in the domain of Deep Reinforcement Learning are as follows:

1. Proposing and discussing technical steps for parallel sample collection of robotic data for Machine Learning approaches achieving a real time speedup by a factor of several hundred on a consumer-grade computer.
2. Presenting a learning framework that combines parallel sample collection with Deep Reinforcement Learning for both on- and off-policy methods.
3. Training a trotting and balancing policy within 1 hour for ANYmal and Valkyrie respectively and deployment of the policies on the real robot platforms.
4. Solving MuJoCo benchmark tests within minutes using our proposed approach.

Paper P6: Bayesian Optimisation for Whole-Body Control of High Degree of Freedom Robots through Reduction of Dimensionality.

The contributions of **Paper P6** in the domain of automatic parameter tuning through BO are:

1. A novel Alternating Bayesian Optimization (ABO) algorithm that is able to optimize black-box objective functions with high-dimensional parameters.
2. An automated parameter tuning framework for Whole-Body Control that can find the high-dimensional optimal parametric set from scratch within a few iterations.
3. Evaluation of the versatility of the proposed ABO on the COCO benchmarking platform that shows consistent convergence of finding near global optima for challenging high-dimensional objective functions.

Paper P7: Decoding Motor Skills of Artificial Intelligence and Human Policies: A Study on Humanoid and Human Balance Control.

The contributions of the reverse-engineered controllers in **Paper P7** are as follows:

1. New paradigm for using machine learning to facilitate quicker control development, as an alternative way of leveraging the power of machine learning in addition to other options that intend to use learning directly in real-world applications.
2. Infer underlying principles of an AI policy by studying its perception-action relation, i.e., reverse-engineer an equivalent controller in terms of functionality based on a black-box policy.
3. Propose a Minimum-Jerk Model-Predictive Control (MJMPC) framework that quantitatively reflect both the AI and human push recovery policies.
4. Show that the engineered controller has high similarity (Coefficient of Determination more than 90%) with the collected data, and also exhibits the same human-like push recovery strategies, which emerge from the proposed MJMPC without the need of manual switching between the strategies.

2.2 Contributions in Combining Learning & Control

Paper P1: Hierarchical generative modelling for autonomous robots

Authors: Kai Yuan*, Noor Sajid*, Karl Friston, and Zhibin Li

*Equal contribution and co-first authors

Abstract: Humans can produce complex movements when interacting with their surroundings. This relies on the planning of various movements and subsequent execution. In this paper, we investigated this fundamental aspect of motor control in the setting of autonomous robotic operations. We consider hierarchical generative modelling—for autonomous task completion—that mimics the deep temporal architecture of human motor control. Here, temporal depth refers to the nested time scales at which successive levels of a forward or generative model unfold: for example, the apprehension and delivery of an object requires both a global plan that contextualises the fast coordination of multiple limb movements. This separation of temporal scales can also be motivated from a robotics and control perspective. Specifically, to ensure versatile sensorimotor control, it is necessary to hierarchically structure high-level planning and low-level motor control of individual limbs. We use numerical experiments to establish the efficacy of this formulation and demonstrate how a humanoid robot can autonomously solve a complex task requiring locomotion, manipulation, and grasping, using a hierarchical generative model. In particular, the humanoid robot can retrieve and deliver a box, open and walk through a door to reach the final destination. Our approach, and experiments, illustrate the effectiveness of using human-inspired motor control algorithms, which provide a scalable hierarchical architecture for autonomous performance of complex goal-directed tasks.

Published in: Nature Machine Intelligence, 2022 (under review)

Contribution: In this work, we applied the core principles of human motor control and apply it to achieve autonomous robotic operations on Valkyrie. We propose an implicit hierarchical generative model that has close similarities with human motor control and show how the framework can learn to autonomously and robustly complete a complex loco-manipulation task of box retrieval and delivery, and manipulation to open a door.

Applying the core principles of human motor control on Valkyrie to achieve autonomy for robotic tasks.

Context: We extended the work from P2 and P3 by lifting the architecture-limitation of MELA, where the experts needed the same action space, by showing that a hierarchical structure can operate a robot, where every limb is an expert.

Extended work of P2 and P3.

Application: This work presents a modular control framework that is used throughout this thesis and allows autonomous task completion for robotics applications. By design, the proposed hierachical generative model can be used for any autonomous task that requires both planning for task completion and control for closed-loop, stable realisation on the robot. As the framework poses no assumption on the individual modules, the individual components can be replaced by any required skill – either a learned policy or an engineered controller. Example applications and implementations of that combines learned policies with engineered controllers can be found in Chapter 3.

Paper P2: Multi-expert learning of adaptive legged locomotion

Authors: Chuanyu Yang*, Kai Yuan*, Qiuguo Zhu, Wanming Yu, and Zhibin Li

*Equal contribution and co-first authors

Abstract: Achieving versatile robot locomotion requires motor skills that can adapt to previously unseen situations. We propose a multi-expert learning architecture (MELA) that learns to generate adaptive skills from a group of representative expert skills. During training, MELA is first initialized by a distinct set of pretrained experts, each in a separate deep neural network (DNN). Then, by learning the combination of these DNNs using a gating neural network (GNN), MELA can acquire more specialized experts and transitional skills across various locomotion modes. During runtime, MELA constantly blends multiple DNNs and dynamically synthesizes a new DNN to produce adaptive behaviors in response to changing situations. This approach leverages the advantages of trained expert skills and the fast online synthesis of adaptive policies to generate responsive motor skills during the changing tasks. Using one unified MELA framework, we demonstrated successful multiskill locomotion on a real quadruped robot that performed coherent trotting, steering, and fall recovery autonomously and showed the merit of multi-expert learning generating behaviors that can adapt to unseen scenarios.

Published in: Science Robotics, 5(49), 2020

Contribution: In this paper, we proposed a multi-expert learning architecture (MELA) that contains multiple expert neural networks,

Multi-Expert Learning Architecture that synthesises expert skills into adaptive behaviours for animal-like motions on quadruped.

each with a unique motor skill, and a gating neural network (GNN) that fuses expert networks dynamically into a more versatile and adaptive neural network. We showed that MELA generates multiple distinctive skills effectively and synthesizes multiskill policies with adaptive behaviors in unseen situations.

We also demonstrated a breadth of adaptive behaviors for contact-rich locomotion on a real robot and various extreme test scenarios in a physics simulation.

Context: This work built upon the DRL policies trained through P5 where we trained various policies for legged robots, e.g., trotting, and fall recovery. Using these policies as experts, the question arose whether we can learn a unified policy that combines all expert behaviours. Furthermore, we drew inspiration from biological systems, which learn new skills from a set of base skills. Applied to the context of legged locomotion, we showed that trotting and fall recovery can be used as base skills to learn adaptive behaviours to deal with unseen scenarios and learn skills, such as steering, goal following, and fall-resilient locomotion.

Uses DRL policies from P5 and basis for P3.

The principle of using multiple experts for sensorimotor coordination was further extended in P3, where we generalised MELA into Multi-Expert Synthesis (MES).

Application: The proposed MELA framework is suitable for any multi-expert situation the goal is to blend the skills of the experts and learning new skills from the set of existing expert skills. The experts are resprented as Neural Network and usually learned through back-propagation of reward signals in a Reinforcement Learning paradigm. An example application can be found in Chapter 4.

Paper P3: Multi-Expert Synthesis for Versatile Locomotion and Manipulation Skills

Authors: Kai Yuan, and Zhibin Li

Abstract: This work focuses on generating multiple coordinated motor skills for intelligent systems and studies a Multi-Expert Synthesis (MES) approach to achieve versatile robotic skills for locomotion and manipulation. MES embeds and uses expert skills to solve new composite tasks, and is able to synthesise and coordinate different and multiple skills smoothly. We proposed essential and effective design guidelines for training successful MES policies in simulation, which were deployed on both floating- and fixed-base robots. We formulated new algorithms to systematically determine task-relevant state variables for each individual experts which improved robustness and learning efficiency, and an explicit enforcement objective to diversify skills among different experts. The capabilities of MES policies were validated in both simulation and real experiments for locomotion and bi-manual manipulation. We demonstrated that the MES policies achieved robust locomotion on the quadruped ANYmal by fusing the gait recovery and trotting skills. For object manipulation, the MES policies learned to first reconfigure an object in an ungraspable pose and then grasp it through cooperative dual-arm manipulation.

Published in: Autonomous Robots, 2022 (under review)

Contribution: We proposed a systematic design approach for Multi-Expert Synthesis (MES) to enable efficient learning of skill coordination and effective synthesis of multiple experts during tasks. To this end, we formulated a state selection algorithm that used the learned state value function to quantitatively identify essential states and rule out task-irrelevant variables. We also tackled a problem in multi-expert systems called mode-collapse. We proposed an explicit enforcement of diversity among multiple experts by maximising their discriminability. We demonstrated that MES can achieve motor skill synthesis in both robot locomotion and manipulation.

Automatic state space selection for Multi-Expert Synthesis of quadrupeds and dual-arm coordination.

Context: This work is an extension of P2, where propose a more general formulation called Multi-Expert Synthesis (MES). We extended the application from locomotion to manipulation as well. Furthermore, we improved MELA by automatically selecting state space, which was done manually in P2, and explicitly enforcing diversity across experts.

Extension of P2 with more general assumptions and applied on quadrupeds and dual arm manipulators.

Application: Similar to MELA, MES can be used for any situation, where the skills of multiple experts should be unified into one policy and new skills should be learned from the existing experts. Further, the two main contributions of this work – automatic state space selection and expert diversification – are applicable for the design of state spaces and whenever multiple experts are trained concurrently and diversification among them preventing mode collapse is desired. An example application can be found in Chapter 5.

2.3 Contributions in Control for Robotics

Paper P4: An Improved Formulation for Model Predictive Control of Legged Robots for Gait Planning and Feedback Control

Authors: Kai Yuan, and Zhibin Li

Abstract: Predictive control methods for walking commonly use low dimensional models, such as a Linear Inverted Pendulum Model (LIPM), for simplifying the complex dynamics of legged robots. This paper identifies the physical limitations of the modelling methods that do not account for external disturbances, and then analyses the issues of numerical stability of Model Predictive Control (MPC) using different models with variable receding horizons. We propose a new modelling formulation that can be used for both gait planning and feedback control in an MPC scheme. The advantages are the improved numerical stability for long prediction horizons and the robustness against various disturbances. Benchmarks were rigorously studied to compare the proposed MPC scheme with the existing ones in terms of numerical stability and disturbance rejection. The effectiveness of the controller is demonstrated in both MATLAB and Gazebo simulations.

Published in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1-9). IEEE, 2018

Numerically stable and robust control formulation for MPC.

Contribution: This paper proposed a new formulation for Model Predictive Control (MPC) of Legged Robots that solves numerical and analytical issues that previous Linear Inverted Pendulum Models (LIPM) based formulations exhibited. LIPM is a commonly used in MPC, because it captures the dynamics of floating based systems well, while being linear and simplified model and thus suitable for real-time optimisation over long prediction horizons.

More robust to external disturbances than existing LIPM formulations.

In this work, we showed that our proposed formulation is physically correct, more robust to external disturbances than existing LIPM formulations, and numerically stable for long prediction horizons, which is not the case for some existing LIPM formulations. We further showed that combined with a whole-body controller, the humanoid Valkyrie can locomote over unmodelled terrains of 5° pitch and 10° roll inclination respectively.

The controller is used to actuate robots in our subsequent works.

Context: This paper is the foundation of all subsequent control architectures, where we developed a control framework in which we can actuate a legged robot in a robust and stable manner. Our proposed improved formulation for MPC is used as a mid-level stability controller to provide stable trajectories that are then tracked with a low-level joint controller. This paper is the basis for our work in Appendix E, where we automatically optimise the parameters of the controller developed in this work. Further, the control architecture developed in this work is used our later publication [39] to actuate the robot and make it follow high-level reference trajectories.

Application: The proposed formulation for MPC of legged robots can be used for any legged robot, e.g., bipeds or quadrupeds. Using the proposed formulation both gait planning and feedback control can be achieved to enable the locomotion of legged robots. An example application demonstrating robust biped locomotion can be found in Appendix A.

2.4 Contributions in Learning for Motion Intelligence

Paper P5: Fast Sample Collection Through Massive Parallelisation For Accelerated Deep Reinforcement Learning Development

Authors: Kai Yuan, Quentin Rouxel, Yu Ning, and Zhibin Li

Abstract: For learning methods in Robotics, such as Deep Reinforcement Learning, the most time-consuming component is the collection of data and samples. In this work, we propose a fast sample generation procedure through parallelisation of environments leveraging the multi-threaded architecture of modern computer hardware and utilising the parallel nature of GPU's. We show how these capabilities can be used in conjunction with state of the art on- and off-policy Deep Reinforcement Learning algorithms to solve a variety of Robotics problems. In particular, we conduct benchmarks on MuJoCo environments achieving up to 1200 faster than real time solving the benchmarks within

minutes. Furthermore, we train a trotting and balancing policy for the quadruped ANYmal and biped Valkyrie respectively, and deploy the policies on the real robots. Through our proposed parallelisation framework, we achieve simulation speed 200 times faster than real time, and can successfully train and deploy the trotting and balancing policies on the real robots within 1 hour.

Published in: IEEE/RSJ International Conference on Intelligent Robots and Systems 2022 (under review)

Contribution: This paper proposed technical steps to allow fast sample collection of robotic data through massive parallelisation for Machine Learning approaches achieving a real time speedup by a factor of several hundred on a consumer-grade computer. This allows the rapid development of learned robotic programmes on real robots. We showed, that training a trotting and balancing policy can be achieved within 1 hour for ANYmal and Valkyrie respectively. These policies were directly deployed on the real robot platforms. Furthermore, we open-sourced the developed software.

Deep Reinforcement Learning (DRL) training and deploying policies on ANYmal and Valkyrie within an hour through massive parallelisation.

Context: This work is the foundation for all subsequent publications that use DRL, i.e., P1, P2, P3, and P7. The parallelised architecture and the DRL algorithms outlined in this paper are used to train DRL policies, which are then directly deployed on real robots, such as Valkyrie, ANYmal, Jueying, and Panda. The fast sample collection procedure described in this paper allowed us rapid development, where one policy was trained and deployed on the real robot within hours.

The DRL training procedure from this work is used to rapidly develop algorithms in all of our subsequent works using DRL.

Beside direct deployment, the trained policies are further used in P7, where we reverse-engineer the trained policy, or use the policies as experts for Multi-Expert Synthesis (P1, P2, P3s).

Application: The proposed parallelisation technique allows fast collection of data in simulation. It is thus applicable for any algorithm that requires large amounts of simulation data. Examples include Reinforcement Learning algorithms that use the simulation data to train a policy that maximises a cumulative, user-defined reward. Example applications are in every work that uses Deep Reinforcement Learning to train the policy, i.e., Chapters 3 – 5, Appendix C, D.

Paper P6: Bayesian Optimisation for Whole-Body Control of High Degree of Freedom Robots through Reduction of Dimensionality

Authors: Kai Yuan, Iordanis Chatzinikolaidis, and Zhibin Li

Abstract: This paper aims to achieve automatic tuning of optimal parameters for whole-body control algorithms to achieve the best performance of high-DoF robots. Typically the control parameters at a scale up-to hundreds are often hand-tuned yielding sub-optimal performance. Bayesian Optimization (BO) can be an option to automatically find optimal parameters. However, for high dimensional problems, BO is often infeasible in realistic settings as we studied in this paper. Moreover, the data is too little to perform dimensionality reduction techniques such as Principal Component Analysis or Partial Least Square. We

hereby propose an Alternating Bayesian Optimization (ABO) algorithm that iteratively learns the parameters of sub-spaces from the whole high-dimensional parametric space through interactive trials, resulting in sample efficiency and fast convergence. Furthermore, for the balancing and locomotion control of humanoids, we developed techniques of dimensionality reduction combined with the proposed ABO approach that demonstrated optimal parameters for robust whole-body control.

Published in: IEEE Robotics and Automation Letters, 4(3), pp.2268-2275, 2019

Contribution: We proposed a novel algorithm called Alternating Bayesian Optimization (ABO) that optimises black-box objective functions with high-dimensional parameters. In common Bayesian Optimisation (BO), only a parameter space up to 10 dimensions can be optimised over. We show that our ABO algorithm can automatically tune 36 parameters for Whole-Body Control from scratch within a few iterations. We further evaluated the versatility of ABO on the COCO benchmarking platform that shows consistent convergence of finding near global optima for challenging high-dimensional objective functions (up to 60 dimensions).

Automatic parameter tuning (> 36 parameters) of Whole-Body Controller for Valkyrie.

Context: This work builds upon P4, where we developed a mid-level stability controller and a low-level joint controller for robot locomotion. The main obstacle for well-performing controllers was the time-consuming and often sophisticated tuning of parameters. Our proposed automatic parameter tuning provided parameters in an automatic fashion with higher performance than manually tuned parameters. These parameters were used in the publications P4 and used as starting parameter set in [39].

ABO is used to automatically tune parameters of our whole-body controllers.

Application: The proposed automatic parameter tuning method is applicable whenever parameters should be tuned in an automatic fashion, where a meta objective is known for how the robot should act. An example application is shown in Appendix E.

Paper P7: Decoding Motor Skills of Artificial Intelligence and Human Policies: A Study on Humanoid and Human Balance Control

Authors: Kai Yuan, Christopher McGreavy, Chuanyu Yang, Wouter Wolfslag, and Zhibin Li

Abstract: From the advancement in computers, computer-aided design has emerged for mechanical and electronic engineering, architecture, and many other engineering fields. Foreseeing a similar development curve and technology wave, we forecast a new emerging discipline in the near future that uses learning-aided approaches to catalyze control development, alongside other similar applications such as medicine discovery. In this article, we propose a new paradigm for using machine learning to facilitate quicker, more efficient, and more effective control development, as an alternative way of leveraging the power of machine learning in addition to other options that intend to use learning directly in real-world applications.

Published in: IEEE Robotics & Automation Magazine, 27(2), pp.87-101., 2020

Contribution: This paper proposed a new paradigm for using machine learning in Robotics. Alternative to directly deploying a learned policy on the robot, this work proposed to use the learned policy as guideline for control design. By reverse-engineering a controller that exhibits the same characteristics as the policy, we get the best out of both worlds: the controller is certifiable, transparent, can be analysed, and further fine-tuned in an interpretable manner, while the controller exhibits the complex behaviours of the learned policy.

New paradigm for Machine Learning: reverse-engineering an AI policy into controller with same behaviour while being transparent and certifiable.

We showed this paradigm on the humanoid Valkyrie for the task of push recovery. We proposed a Minimum-Jerk Model-Predictive Control (MJMPC) framework that quantitatively reflect both the AI and human push recovery policies. The engineered controller has high similarity (Coefficient of Determination more than 90%) with the collected data, and also exhibited the same human-like push recovery strategies, which emerge from the proposed MJMPC without the need of manual switching between the strategies.

Context: This work used a DRL policy trained in the DRL framework outlined in P5 to reverse-engineer a controller.

Uses the DRL policy from P5.

Application: The proposed paradigm to reverse-engineer a policy can be used whenever data from a black-box policy is available and domain expertise exists regarding what type of controller family could be replicate the black-box policy. An example application can be found in Appendix D, where push recovery domain expertise is used to reverse-engineer both an AI policy and a human policy.

2.5 Practical Aspects of Algorithm Design

Beside technical contributions that consitute of novel algorithms, this thesis also provides practical aspects as contribution. In particular, we provide insights and lessons learned regarding the state space design, reward shaping process, and control framework.

Please find details regarding state space design in the Sections 3.4, 4.5, 5.3, C.4. Reward shaping is detailed in Sections 3.4, 3.4, 4.5, 5.3, 3. Lastly, a discussion regarding the influence of control parameters can be found in Sections 4.5, 5.5, E.3.

2.6 List of Publications

For an overview of the up-to-date publication list, please refer to https://scholar.google.com/citations?user=8eLlbhMAAAAJ&hl=en.

First author publications.

- **Yuan, K.***, Sajid, N.*, Friston, K. and Li, Z., 2022. Hierarchical generative modelling for autonomous robots. submitted to *Nature Machine Intelligence*. (*these authors contributed equally to this work)

- **Yuan, K.** and Li, Z., 2022, Multi-Expert Synthesis for Versatile Locomotion and Manipulation Skills. submitted to *Autonomous Robots*.
- **Yuan, K.**, Rouxel, Q., Ning, Y. and Li, Z., 2022. Fast Sample Collection Through Massive Parallelisation For Accelerated Deep Reinforcement Learning Development. submitted to *Springer Nature Applied Sciences*.
- Yang, C.*, **Yuan, K.***, Zhu, Q., Yu, W. and Li, Z., 2020. Multi-expert learning of adaptive legged locomotion. *Science Robotics*, 5(49). (*these authors contributed equally to this work)
- **Yuan, K.**, McGreavy, C., Yang, C., Wolfslag, W. and Li, Z., 2020. Decoding Motor Skills of AI and Human Policies: A Study on Humanoid and Human Balance Control. *IEEE Robotics & Automation Magazine*, 27(2), pp.87-101.
- **Yuan, K.**, Chatzinikolaidis, I. and Li, Z., 2019. Bayesian optimization for whole-body control of high-degree-of-freedom robots through reduction of dimensionality. *IEEE Robotics and Automation Letters*, 4(3), pp.2268-2275.
- **Yuan, K.** and Li, Z., 2018. An improved formulation for model predictive control of legged robots for gait planning and feedback control. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1-9).

Co-authored publications.

- Acero, F., **Yuan, K.** and Li, Z., 2022. Learning Perceptual Locomotion on Uneven Terrains using Minimal Visual Information. submitted to *Robotics and Automation Letters (RAL)*
- Rouxel, Q., **Yuan, K.**, Ruoshi Wen, and Li, Z., 2022. Multi-Contact Motion Retargeting using Whole-body Optimization of Full Kinematics and Sequential Force Equilibrium. submitted to *IEEE/ASME Transactions on Mechatronics*.
- Hu, W., **Yuan, K.**, Yang, C., and Li, Z., 2021. Learning Motor Skills of Reactive Reaching and Grasping of Objects. *IEEE ROBIO 2021*.
- Sun, Z., **Yuan, K.**, Hu, W., Yang, C. and Li, Z., 2020. Learning Pregrasp Manipulation of Objects from Ungraspable Poses. *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9917-9923).
- McGreavy, C., **Yuan, K.**, Gordon, D., Tan, K., Wolfslag, W.J., Vijayakumar, S. and Li, Z., 2020. Unified push recovery fundamentals: Inspiration from human study. *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 10876-10882).
- Wen, R., **Yuan, K.**, Wang, Q., Heng, S. and Li, Z., 2020. Force-guided high-precision grasping control of fragile and deformable objects using semg-based force prediction. *IEEE Robotics and Automation Letters*, 5(2), pp.2762-2769.
- Yang, C., **Yuan, K.**, Heng, S., Komura, T. and Li, Z., 2020. Learning natural locomotion behaviors for humanoid robots using human bias. *IEEE Robotics and Automation Letters*, 5(2), pp.2610-2617.
- Hu, W., Chatzinikolaidis, I., **Yuan, K.** and Li, Z., 2018. Comparison study of nonlinear optimization of step durations and foot placement for dynamic walking. *2018 IEEE International*

Conference on Robotics and Automation (ICRA) (pp. 433-439).

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

Hierarchical generative modelling for autonomous robots

3

3.1 Introduction

Humans can control their bodies to produce intricate motor behaviours that are aligned with their objectives e.g., opening the door or moving a box. These tasks rely on the coordination of two distinct processes: motor planning and generation [91, 92, 93, 94]. To realise this, human motor control prescribes nested time scales at which different levels of the neuronal hierarchy evolve [95, 96], e.g., a high-level plan to arrive at a place can entail multiple, individual, reflexive low-level limb movements for walking. Similar distinctions have been introduced in the robotics and control literature [97], with promising results by combining robot control with learnt motor movements. However, control and planning are often designed manually and separately, with little or no feedback between the two [10]. This fundamental structure limits the performance of the robot, and demand more human involvement at the deployment stage even in controlled conditions.

Knowing these fundamental principles, we postulate that designing appropriate robotics systems need to consider motor control as a consequence or outcome of a (learnt) hierarchical generative model – which is able to optimise and adapt future actions in uncertain environments. These hierarchical generative models are the descriptions of how sensory observations are generated: encodings of sensorimotor relationships relevant for motor control [98, 99]. This implies that autonomous systems must be conceptualised as hierarchical motor control, capable of dealing with complex and diverse tasks [100].

Practically, hierarchical generative models speak to a spatiotemporal separation of planning and motor control of the individual limbs, despite functional integration [101, 98, 102]. This can be delineated as a series of distinct layers that provide appropriate motor control [103](Fig. 3.1). The lowest layers predict the proprioceptive signals – generated using a forward model encoding the physics, and the kinetics that undergird motor execution. These can be characterised as predicting shifts in the equilibrium position without explicit modelling of the task dynamics (i.e., the equilibrium point hypothesis) [104, 105]. Then, the layer above generates necessary movements to make the projections of the trajectory that would transfer the agent from one fixed point to the next. This speaks to the mid-level stability control that a human has over limbs, to perambulate in an upright manner and maintain the centre of gravity. The highest layer then pertains to planning [106, 102]. Here, different states represent endpoints of an agent's plan, e.g., move a box from a table to another.

To verify our proposition, we introduce an implicit hierarchal generative model for autonomous robot operations using a learning-based scheme. Notably, our formulation has three distinct layers pertaining to planning and motor generation. Aligned with the equilibrium point

hypothesis [105, 104]*, this is a forward model that predicts necessary proprioceptive inputs to execute a desired movement, and the reflexes fill in the gaps and realise the descending predictions or set points. Practically, different layers were optimised together for an appropriate functional integration between the highest and the middle layers. However, it shall be noted that only the middle layer planner had the access to the state feedback, which allows a particular type of factorisation (or functional specialisation). We will further elaborate technical details at later sections.

Using this generative model, we focus on solving the tasks that comprises of sequential, conditional decision-making in two scenarios. In the first scenario, the robot needs to reach the goal location by opening a closed door. The door only opens when a button is pressed. Here, the planner needs to coordinate both legs and arms: walking towards the door happens first with leg movements and then arms start to move and press the button, and finally the robot can reach the goal location. The second scenario requires the robot to carry a box from one table to another. Here the planner needs to follow the order of approaching and picking up the box first, walking towards the second table, and placing the box upon arrival. For both scenarios, our algorithm can perform coherent locomotion, manipulation, and grasping movements similar to humans, and therefore solve all these complex tasks successfully.

The paper is organised as follows. In Section 2, we describe the realisations of two hierarchical generative models for motor control: human motor control and a robotic system capable of autonomous operations. The robotic system instantiates an implicit hierarchical generative model for motor control, which embeds a bidirectional information propagation between different layers of the generative model. Next, we show that our proposed architecture can perform tasks remarkably similar to humans (Section 3). We highlight the common ground between our approach and optimal motor control theory, conventional messaging schemes, linear quadratic regulators, and potential future directions (Section 4). Moreover, we discuss that learning is a natural part of such generative models and how learning can be integrated with existing robotics approaches in a compatible manner. Lastly in Section 5, we provide details of our implementation of algorithms.

* Equilibrium point hypothesis: all movements are generated by the nervous system through a gradual transition of equilibrium points along a desired trajectory[105, 104].

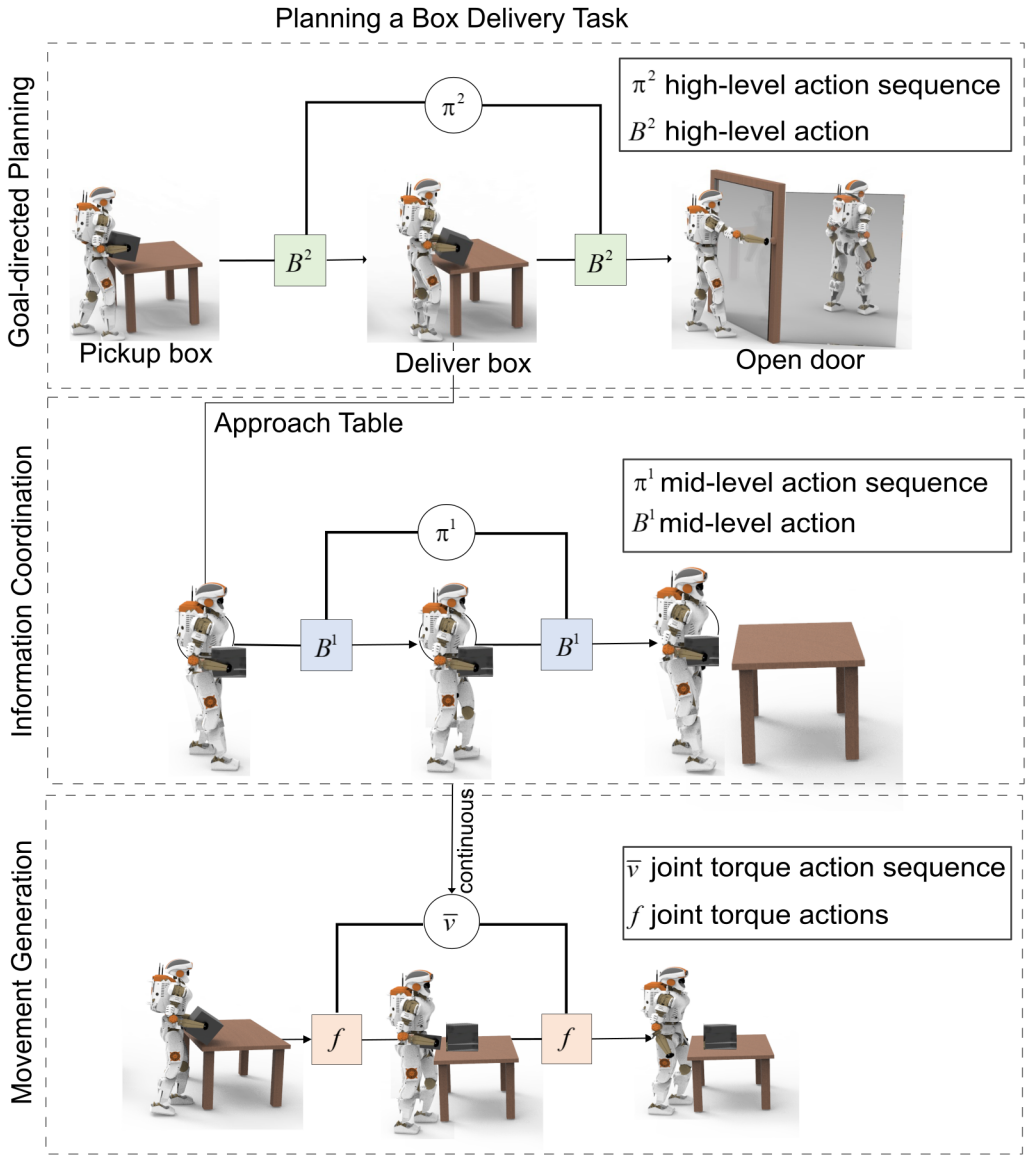


Figure 3.1: Pictorial representation of a hierarchical generative model for moving boxes – a form of motor control. Typically, this represents the conditional dependencies between states and how they cause outcomes. For simplicity, we compressed this as filled squares that denote actions, and circles that represent action sequences. The key aspect of this is its hierarchical structure that represents sequences of action over time. Here, actions at higher levels generate the initial actions for lower levels–that then unfold to generate a sequence of actions: c.f., associative chaining. Crucially, lower levels cycle over a sequence for each transition of the level above. It is this scheduling that endows the model with deep temporal structure. Particularly, planning (first row; highest level) to "deliver the box" generates the actions for the information coordination level (second row; middle level) i.e., " movement towards the table". This in turn determines the initial actions for movement generation (third row; lowest level) of arms to "place the box" on the table.

3.2 Results

We hypothesise that motor control is a natural outcome of hierarchical generative modelling (Fig. 3.1). Hierarchical generative models are descriptions of how observations of sensory information are generated – expressed as a joint distribution over the parameters of interest. These can be factorised out to represent particular conditional dependencies of interest that formalise sensorimotor relationships related to motor control [99, 98]. Here, causes of proprioceptive input can be predicted

by fitting the model to observed sensory data [107]. This provides a more transparent way to interpret the undergoing decision-making process within the robot. Interestingly, having a generative model gives for free the five core principles of hierarchical motor control introduced in [100] (see Table 3.1).

The above has established the relevance of hierarchical generative models for motor control. In the following, we describe the realisations in two domains: human and robotics (see Fig. 3.2).

Generative Model for Human Motor Control

The generative models observed in human motor control feed in continuous proprioceptive signals and propagate information to the highest levels that are responsible for planning, and then back down at each level to generate particular movements (Fig. 3.2). These proprioceptive signals are usually multisensory, here we limit to motor control using muscular input for brevity.

The generative model's lowest (and fast evolving) layer includes the spinal cord and the brainstem. The local (factorised) circuits in these regions are responsible for organising movements, given descending predictions from different higher levels. Example projections, from upper motor neurons in the cortex, control movement indirectly via pathways that project to the brainstem motor control centres, which, in turn, project to the local organising circuits in the brainstem and the spinal cord. Accordingly, these areas are responsible for evaluating the discrepancy between the proprioceptive inputs and descending predictions of these data, to drive muscle contraction via the musculoskeletal mechanics [108, 102]. It is believed that these regions coordinate multiple joint movements semi-autonomously over time [109, 110][†]. At this level, different neuronal ensembles have distinct, partially autonomous influences, e.g., the red nucleus controls movements of the arms.

Moving to the middle level of this generative model, we consider the role of the cerebellum and the basal ganglia. The cerebellum receives inputs from the spinal cord and other areas, and integrates these to fine-tune motor activity. In other words, it does not initiate movement, but contributes to its coordination, precision, and speed, through a feedforward mode of operation. Therefore, it can be thought of being responsible for amortised (habitual) control for deliberative goal-directed behaviours, which are characterised by the subcortical and cortical interactions. Additionally, the cerebellum can receive information from the motor cortex, process this information, and send motor impulses to the skeletal muscle (via the spinal cord). Conversely, the basal ganglia is responsible for motor program selection, informed by the thalamus and the motor cortex [111] – appropriately modulating the lower level of the generative model (spinal cord via the brainstem). Basal ganglia is revealed to support the learning of action selection [112].

[†] These trajectories are a succession of fixed points such that each fixed point is realised in a specified (e.g., theta) cycle over a particular time period. Then motor sequence can be regarded as filling in the gaps between a series of fixed points, where each fixed points is specified by the level above in very scheduled saltatory discrete fashion.

Table 3.1: Summary of the key principles of hierarchical motor control (Merel et al., 2019), with exemplar realisations in human motor control and our robotic system. We omit the principle of modular objectives here (sub-systems trained to optimise specific objectives different from the global task objective) to avoid confusion of localised objectives under a hierarchical generative model – because a factorised generative model architecture, with the objective of maximising model evidence naturally leads to distinct factor specific objectives at each point in the hierarchy.

Principle	Description	Hierarchical Generative Models	Human Motor Control	Autonomous Robot Operations
Information factorisation	Different information is processed by distinct sub-systems.	Result of factorised distribution of appropriate latent states within the generative model.	Different sensory signals are routed to different parts in the hierarchy e.g., what and where streams. These neuronal populations can be characterised as factorised states responsible for sub-systems.	Only task relevant sensory signals are used by the individual layers with irrelevant states hidden across layers. This speaks to an explicit factorisation of sensory signal and which parts of the system has access to it.
Partial autonomy	Lower hierarchical levels can semi-autonomously produce outputs with minimum input from layers above.	Result of factorising out the state space into multiple distinct layers, and lower layers can independently accomplish sub-goals at a (relatively) fast temporal scale.	Semi-autonomous coordination of joint movement at lower layers (i.e., brainstem and spinal cord). These operate at faster temporal scale, and do not require continuous input for higher layers.	Full autonomy and stability guarantee in individual layers. Explicitly, we introduce stable mid-level and low-level motions for random higher-level inputs. This ensures that lower layers can independently perform fast movements.
Amortised control	Re-execute appropriate behaviours rapidly using learnt movements.	Learnt probability distributions that parameterise this generative model can be used for amortised control. That allows for habitual control based on previously learnt distributions.	Cerebellum is responsible for amortised control of deliberative and goal-directed behaviours, evoking fast habitual control for repeated actions.	The system learnt policies (i.e., action-state mappings) that provide habitual control for rapidly re-executing appropriate actions.
Multi-joint coordination	Degenerate coupling of different components operating as a whole for motor control.	Result of state factorisations that introduce flexible mapping across and within each layer.	Different neuronal ensembles have distinct influences e.g., the red nucleus controls movements of the arms. Much like factorised states, these neuronal ensembles come together to produce intricate movements.	The system is equipped with multiple sub-structures (or policy mappings) that are responsible for specific actuator movement. Together these come together, across and within layers, to produce particular motor movements.
Temporal abstraction	Abstraction of time across hierarchical layers.	Result of hierarchical generative models where higher layers evolve slower than and constrain, the layer below.	Different levels evolve at different temporal and spatial scales, with primary motor cortex responsible for planning (slow timescale) and spinal cord responsible for generation (fast timescale)	The three layers of the system evolve at different temporal scales – much like any hierarchical generative model. The high-level planning is at a slow timescale, mid-level stability control at medium timescale, and low-level joint control at a fast timescale.

The higher level involves the cerebral cortex and the thalamus. The thalamus nuclear receive projections from the cerebellum (related to movement and sensory stimulation) and basal ganglia (related to wilful movements), and thalamus nucleus relay projects directly to the primary motor and premotor association cortices [113]. The cortex has access to factorised sensory streams of proprioceptive signal (e.g., visual, auditory, etc) and hence can coordinate or override habitual control defined by the lower levels. Specifically, the primary motor cortex is responsible for deliberative planning, control, and execution of voluntary movements, for example, when learning a new motor skill.

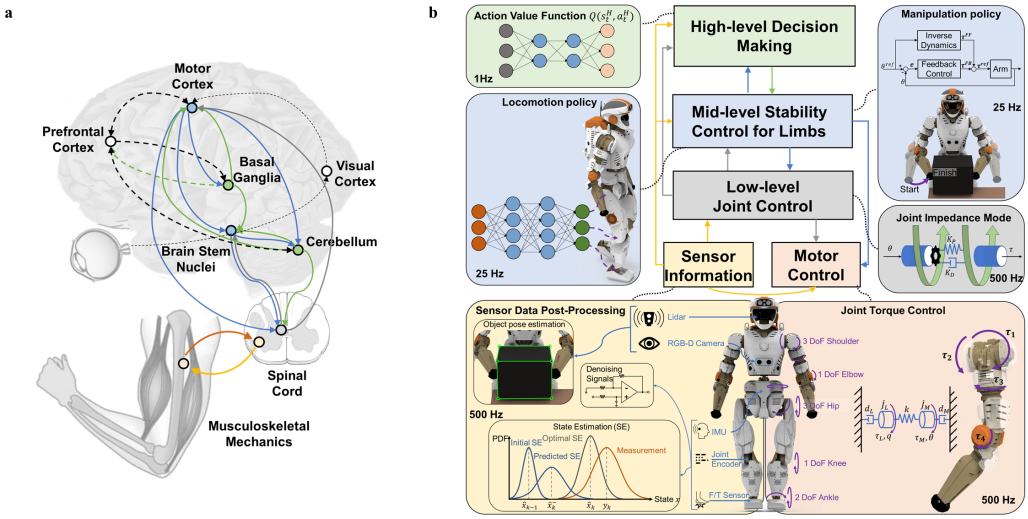


Figure 3.2: Algorithmic realisations of the conceptual framework. Panel A corresponds to the generative model of human motor control, and Panel B corresponds to the implicit generative model for a robotics system. The green nodes in Panel A and green boxes in Panel B refer to the highest layers of human motor control and our implicit generative model respectively. In the implicit generative model, high-level decision making is realised as neural network learned through Deep Reinforcement Learning. The blue nodes in Panel A correspond to the middle level of human motor control and the blue boxes in Panel B are mid-level realisations in our proposed architecture realised as Deep Neural Network policy learned through Deep Reinforcement Learning for Locomotion and Inverse Kinematics and Dynamics policy for Manipulation. On the lowest level, depicted in grey nodes and boxes, a joint impedance controller calculates the torques required for actuation of the robot. Yellow and light-red are sensor information and motor control respectively.

Implicit Hiearchical Generative Model for a Robotics System

Following the key principles of hierarchical motor control (Table 3.1) and the generative model in Figure 3.1, we design an implicit generative model for the humanoid robot consisting of three layers: high-level decision making, mid-level stability control, and low-level joint control (Fig. 3.2). We formulate factorised information flow by restricting state access to specific policies across the three layers, i.e., the locomotion policy only has access to the lower joint states and goals. Here, temporal abstraction is an outcome of specifying a hierarchical generative model, where policies are evaluated at different timescales.

The highest planning layer, evolving at the slowest rate, selects an appropriate sequence of actions of the limbs, which are needed to complete a particular sub-task. It decides where the hands should be and what direction to go. Practically, deep reinforcement learning

(RL) is used here to learn the correct generation of Cartesian position commands for the mid-level stability control.

These commanded targets are realised at the layer below that regulates the balance and stability of the robot while performing manipulation and locomotion. Manipulation is instantiated as a minimum-jerk model-predictive controller that moves the arms to the target positions provided by the high-level policy, and locomotion as a learnt locomotion policy via deep RL that coordinates twelve degrees of freedom legs to reach the destination provided by the higher level. Both these policies are designed to ensure that infeasible set points from the high-level are corrected for the mid-level stability control, so that only stable joint target commands are provided to the low-level joint controller.

Despite receiving inputs from other layers, each layer has partial autonomy over its final predictions and goal. Furthermore, multi-joint coordination is realised by learning a policy that coordinates all joints of legs appropriately for the current state, while the arms coordinate their joints through Inverse Kinematics (IK).

The low-level joint controller is instantiated as joint impedance control and tracks the joint position commands given by the mid-level stability controller. Based on tuned stiffness and damping, the joint impedance control calculates desired torque to follow the target positions closely and smoothly. Lastly, the torque commands are tracked by the actuators using embedded current control of onboard motor drivers.

Hierarchical Generative Model for Loco-Manipulation

In the following, we show how our proposed paradigm of formulating the control architecture as a hierarchical generative model enables a robot to learn autonomous completion of a loco-manipulation task. Additionally, the learned policy exhibits generality and robustness to uncertainty (Fig. 3.3 a-e) while exhibiting the core principles of hierarchical motor control. First, we validate the proposition that hierarchical generative modelling is necessary for autonomous robotic operations. We demonstrate this in two sequential decision-making tasks: moving a box from one table to another and opening a door by pressing a button.

Our implicit hierarchical generative model for a robotic system can successfully and autonomously achieve locomotion, manipulation, and grasping movements similar to humans, and solve all these complex tasks in a coherent manner. Briefly, the highest policy layer of the robot system determinises the action sequence necessary for task completion and sends commands to the lower layers. This allows the robot to carry out the following actions: i) walk to the first table, ii) move arms to pick up the box, iii) walk to the second table, iv) release hand and place the box on the table. Upon successful completion of this task, the same generative model can be used to perform the second task as well. This is achieved by having the transition of a high-level policy to open a closed door, instead of moving a box. The accompanying commands are sent to the lower layers responsible for limb control stability and

joint control. They instantiate mid-level locomotion and manipulation policies to move to the door and position the body close enough to press the button. Having the door opened, the robot passed through the door towards the final goal.

To assess the robustness and generality of the hierarchical generative model, we introduced several perturbations that were not observed during training (Fig. 3.3). First, we introduced external perturbation by placing obstacles (i.e., 5kg box, Fig. 3.3a) in front of the robot and pushing its pelvis (Fig. 3.3b). The results are encouraging: the mid-level locomotion policy withstood both disturbances, moved the obstacle out of the way, and took a step to recover balance after the push. To test the performance further, we modified the environment with unseen conditions by adding a 5° inclined surface (Fig. 3.3c) and a low-friction glass plate (friction coefficient of 0.3, Fig. 3.3d) in front of the door. The robot successfully completed the task post each perturbation. More interestingly, we lesioned the robot by amputating its right foot completely (Fig. 3.3e). Despite this handicap that was never encountered, with only a stump touching the ground in place of its right foot, our hierarchical control was sufficiently robust to deal with this situation and the robot was able to keep balance and complete the task.

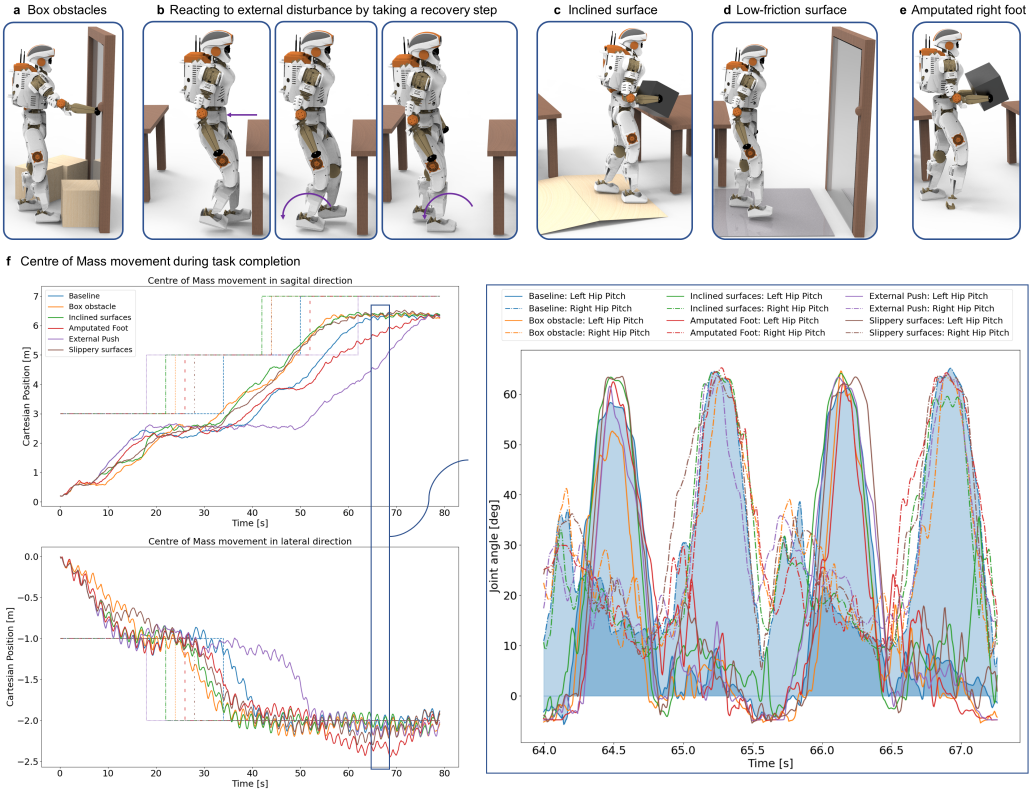


Figure 3.3: Robustness of the system in light of perturbations. Panels A-E show how the robot successfully completes the task in perturbation test scenarios that it has not encountered during training and demonstrate the robustness of our proposed method. From left to right, we place 5kg box-obstacles in front of the robot, push the robot from the front, alter the floor with an inclined and slipper surface, and lesion the robot, but removing the right foot. In Panel F, the amortised control is shown. The hip pitch joint motion is exemplarily used to show how the policy adapts to the perturbation and rapidly re-executes a motion to counteract the perturbation.

Next, we evaluate whether the aforementioned robotic system satisfies the key principles of hierarchical motor control (Table 3.1) that explains

the robust task performance.

Information Factorisation

In this system, factorisation exists across model layers and policy controls – with each responsible for particular information processing. This factorisation ensures that particular external perturbations have minimum impact on task performance. For example, when disturbances occur on the robot legs, it does not significantly impede the task completion. This is because these perturbations are only visible to the locomotion policy, and other layers and control policies can be independent of the disturbances and hence retain their nominal operations.

Practically, information factorisation necessitates clearly defined roles for each sub-system. Thus, any failures in performance can be isolated and fine-tuned for future tasks. For example, if the robot falls over while walking to a goal, the locomotion policy can be identified as the root cause, and hence improving the locomotion policy will resolve the issue without resolving to the high-level planner or the manipulation policy. Further examples include oscillation of the robot limbs, which can be attributed to the low-level joint control; or walking towards the wrong direction, which was due to the command from the high-level policy.

Partial Autonomy

The system is designed with partial autonomy, i.e., minimum interference or support from other layers. Specifically, we implement a clear separation between the highest and middle layer, though they are trained together. This was particularly relevant because the high-level planning layer can send random action sequences to the mid-level stability controller. Without partial autonomy, the robot might become unstable and not able to learn to move appropriately, given such random or potentially unstable high-level commands.

Figure 3.4 visualises this case when the robot is provided with random commands to both the arms and legs. This causes the robot to walk in random directions (Fig. 3.4a) and the arms move around randomly (Fig. 3.4b). Despite imperfect motion tracking, the robot does not fall over and is able to complete the tasks despite of movement deviations.

Amortised Control

After training, the robot engages in amortised control with the ability to re-execute appropriate behaviours rapidly using previously learnt movements. We observed this in baseline and perturbed task setting (inset trajectories in Fig. 3.3f), where the amortised locomotion policy is used to successfully complete task without additional learning.

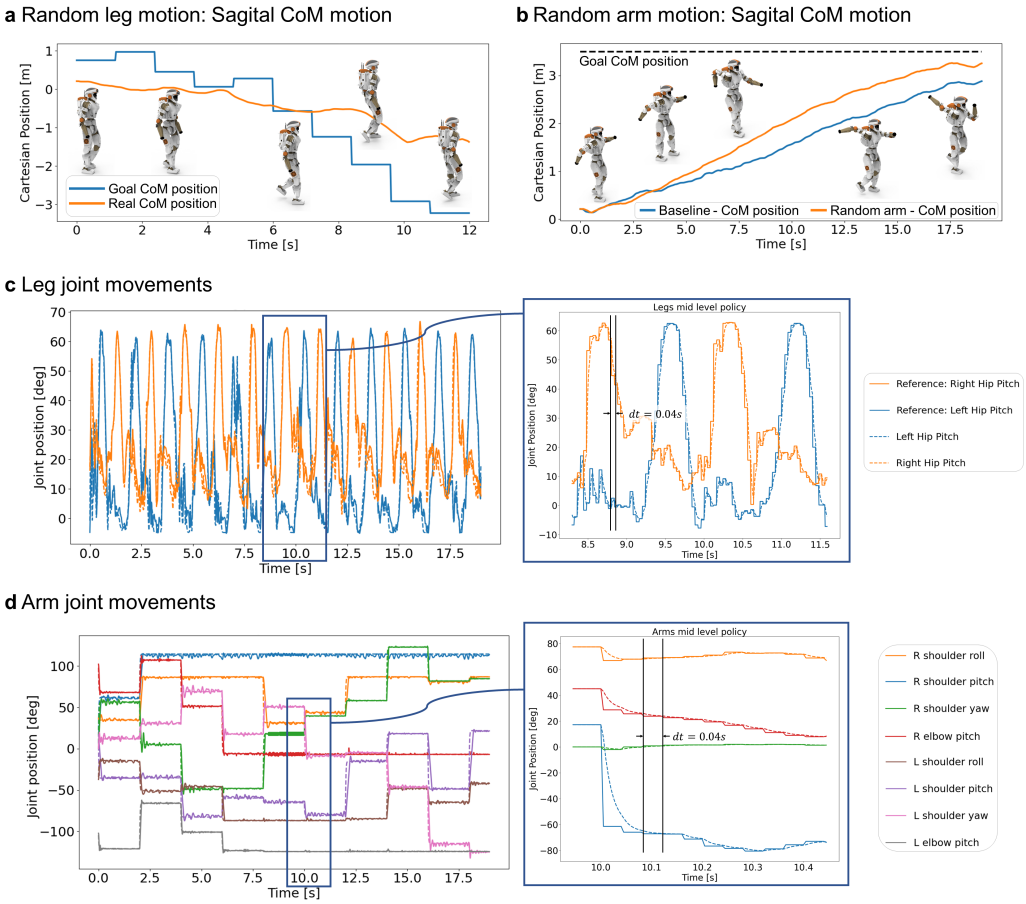


Figure 3.4: State and temporal dynamics of the robot during task performance with random-high level commands. Panels a and b show the sagittal motion of the Centre of Mass (CoM) while following random leg and arm commands respectively. From the robot snapshots corresponding to the time they're shown, the partial autonomy of the mid-level stability controllers can be seen, i.e., good performance of the individual layers despite random and fast changing command inputs. Panels c and d show the leg and arm movements respectively. Here, the temporal abstraction can be seen, where the high-level commands are provided at 0.5 Hz, and the mid-level commands are realised at 25Hz. The joint commands are realised at 500Hz on the joint actuators. In the inset plots of Panels c and d, it can be seen that the joint position trajectories evolve similarly as postulated in the equilibrium point hypothesis.

Multi-Joint Coordination

The robot has multiple sub-structures that are responsible for specific control, which can work together in different ways to generate particular motor movements. Figure Fig. 3.5a demonstrates this multi-joint coordination when pressing the button to open the door in the presence of an obstacle (Task 2). To achieve so, the right arm motions had to coordinate appropriately according to the initial hand position. Also, the shoulder roll (Fig. 3.5b orange line) and elbow (Fig. 3.5b red line) had to adjust and adapt differently from the baseline. Explicitly, these do not yield a fixed motion, instead, the manipulation policy coordinates these joints based on where the Centre of Mass (CoM) is. Therefore, during the baseline reaching motion, arms move differently than the motion in box obstructed case, where the CoM is in a different position because boxes are obstructing the door.

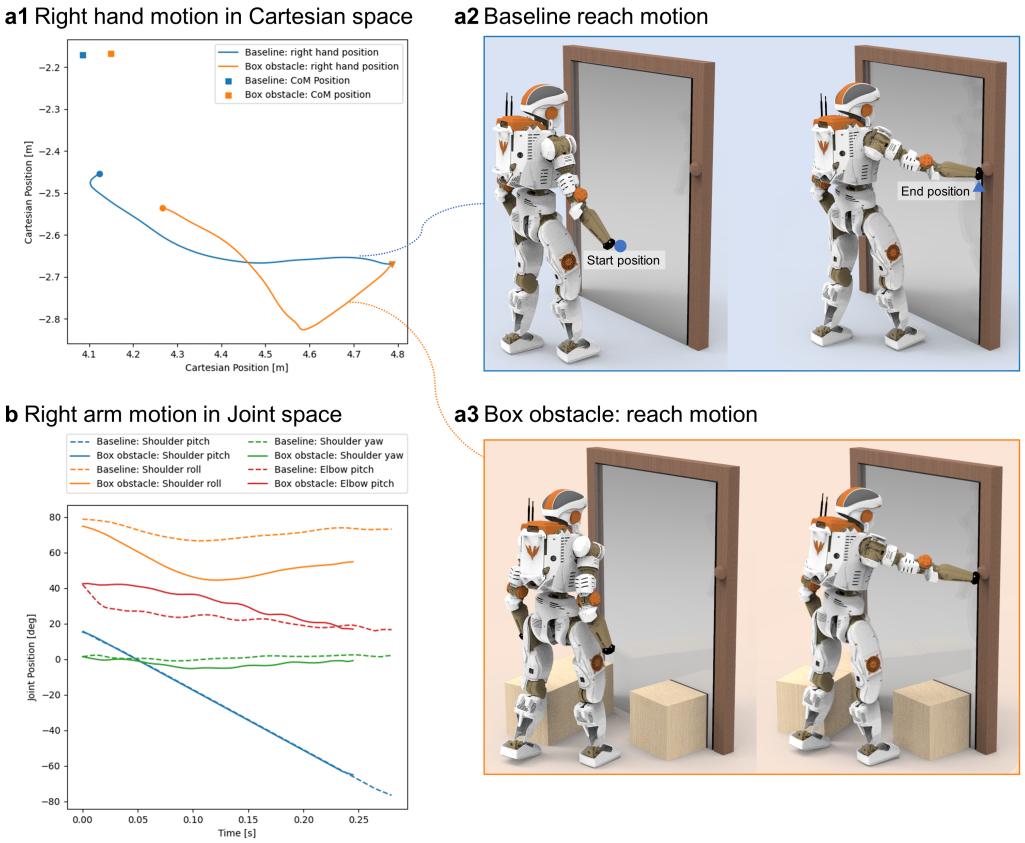


Figure 3.5: Adaptive arm manipulations during opening the door task. Panel a shows the right arm's motion in cartesian space for the baseline (no obstacle) and the box obstacle scenario. When there is an obstacle in front of the door, the arm motion must adapt as the robot's upper body position deviates from its baseline position. The Multi-joint coordination aspect can be seen in Panel b. The adapts to the change of scenario, coordinating the multiple arm joints, such that the end position of the arm is pressing the button to open the door while being obstructed by the box obstacles.

Temporal Abstraction

By design, the three system layers evolve at different temporal resolutions. Figure 3.4 visualises these distinct temporal scales as the robot perambulates. The highest policy layer has a slow timescale of 0.5Hz (Fig. 3.4a). This allows the lower layers to carry out the command in a partially autonomous way, i.e., uninterrupted. Conversely, the mid-level stability control for limbs has a faster timescale at 25Hz (inset trajectories of Fig. 3.4c and 4d). This is needed to generate rapid predictions for the locomotion and manipulation policies. Lastly, the low-level joint control executes these control commands at a frequency of 500Hz on the actuator level.

3.3 Conclusion

In this work, we showed that hierarchical generative models can adequately equip agents with the ability to perform autonomous control of robots. Hierarchical generative modelling provides sensorimotor control systems with a more transparent and flexible approach to interpret the robot's decision making. Importantly, this can be leveraged to improve task performance and identify system failures. It shall be noted that our

proposed hierarchical framework adheres to the key principles of hierarchical motor control, and therefore we are offered the possibilities to: (i) create a system, (ii) rollout the policy and evaluate its performance, (iii) identify the root cause of the limitations in performance, and (iv) resolve issues in the root cause and improve the performance. Our proposed hierarchical generative model showcases how the principles of hierarchical motor control can be applied to robotics to provide these with full autonomy and more robustness. We validated this claim on a robot loco-manipulation task that required the robot to retrieve and deliver a box, opening a door, and walking towards a final goal. Further, we showed the robustness of our approach by showing the successful task completion under perturbances, which the robot did not encounter during training, such as altered environments, external pushes, and even amputating the right foot. By providing robots with full autonomy, humans are relieved from the task of sending low-level commands every few seconds - e.g., foot and hand placement - by interpreting a steady stream of perception and sensor data as in a shared-autonomy and semi-autonomous paradigm. Consequently, cognitive overload that may cause human errors can be prevented. A notable example of human error in high stress situations causing the crash of a robot was the wrong foot placement commanded by an experienced operator during the DARPA Robotics Challenge, the heretofore most advanced competition to test the capabilities of anthropomorphic disaster response robot. Our future work will in improve the planning objectives. In this work, we have shown how a simple objective can achieve good results, and we aim to replace the planner with more sophisticated and neuroscientifically more grounded objectives, i.e., through active inference [114]. Furthermore, we will investigate the robustness capabilities of our method through robotic neuropsychology: causing lesions or stroke-like conditions on the robot to investigate both methods and causes of the resulting policies.

3.4 Methods

We present the hardware implementation here for realising the generative model outlined in Section 2. In particular, we first describe the robotic platform on which the algorithms are implemented. Second, we delineate the task which is autonomously solved by the generative model. Next, we explain the details on the generative model, which consists of high-level decision making, mid-level stability control, and low-level joint control.

Robot Platform

The motions for autonomous task completion are implemented on NASA's humanoid Valkyrie [10]. Valkyrie was designed to operate in extraterrestrial planetary space missions such as unmanned predeployments on Mars – it is 1.87m tall, consists of 44 Degrees of Freedom, and weighs 129kg with ranges of motion similar to humans. The 25 series-elastic actuators in arms, torso, and legs enable human-like locomotion and manipulation; all the joint limits are detailed in Table 3.2. Valkyrie

can sense the environment through proprioceptive and exteroceptive sensors, including a multitude of gyroscopes, accelerometers, load cells, pressure sensors, sonar, LIDAR, depth cameras, and stereo sensors.

Table 3.2: Mechanical specifications of Valkyrie.

	Lower joint position $[rad]$	Upper joint position $[rad]$	Joint velocity $[rad/s]$	Joint torque $[Nm]$
Shoulder roll	-1.3	1.5	5.9	190
Shoulder pitch	-2.9	2.0	5.9	190
Shoulder yaw	-3.1	2.2	11.6	65
Elbow pitch	-2.2	0.1	11.5	65
Torso roll	-0.2	0.3	9.0	150
Torso pitch	-0.1	0.7	9.0	150
Torso yaw	-1.3	1.2	5.9	190
Hip Roll	-0.6	0.5	7.0	350
Hip pitch	-2.4	1.6	6.1	350
Hip yaw	-0.4	1.1	5.9	190
Knee pitch	-0.1	2.1	6.1	350
Ankle Pitch	-0.9	0.7	11	205
Ankle Roll	-0.4	0.4	11	205

Tasks of Interest

To demonstrate how the hierarchical generative model can autonomously solve complex tasks, which do require a particular sequence and coordination of locomotion and manipulation skills, we specifically designed a task that demand coordination of limbs and reasoning about the sequence of actions.

The designed task consists of four subtasks (Fig. 3.6): picking up a box from the first table, delivering the box to the second table, opening the door, and walking to the destination or goal position. To accomplish so, the subtasks have to be carried out in an exact sequence.

The proposed framework allows the robot to learn successful task completion through autonomous interactions with the environment. This is achieved by designing a reward function for the high-level policy, such that cumulative maximisation of reward leads to task completion (see Section 3.4).

The trained policy learns to first pick up the box and hold the box using its hands while approaching the second table. Once the box can be safely placed at the second table, the policy commands the arms to release the box and instruct the robot to move towards the door, while keeping arms in a nominal position. By pushing a button placed on the right side of the door frame, the robot can then open the door. The policy learns to approach the final destination at the right time only when the door open widely and finally walks towards the goal position.

Implicit Generative Model

The generative model is realised by implementing three levels of control in a hierarchical manner (Fig. 2 in the manuscript): high-level decision

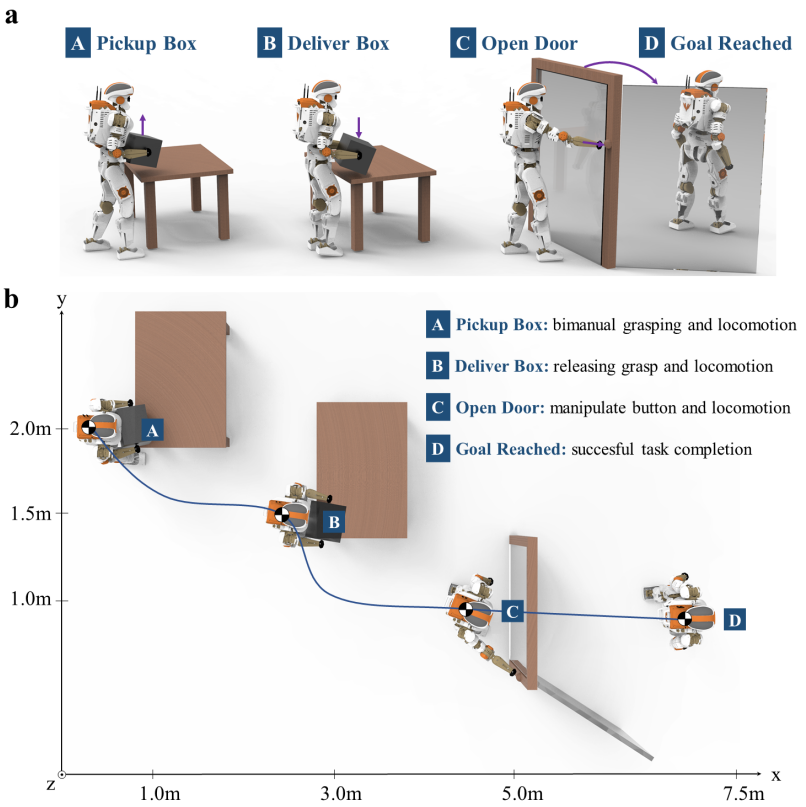


Figure 3.6: Visual presentation of the robot task: pick up a box, deliver it to the second table, approach the door, press the button to open the door, and enter the destination once the door opens fully. (a) shows the side view of the task; (b) shows the top-down view with x-y coordinates in Cartesian space.

making, mid-level stability control, and low-level joint control. All components are designed and trained separately, starting from the lowest level.

First, accurate and robust motor control needs to be guaranteed, such that the low-level joint position control can be realised. The stiffness and damping parameters are tuned to accurately and compliantly track the references, which provides the mid-level stability control. The mid-level stability control consists of a manipulation and a locomotion policy, which are individually designed. The locomotion policy is trained to walk towards a commanded goal position, while the manipulation policy is designed to place the hands on a target position. Finally, the high-level decision making policy is trained via Deep Reinforcement Learning, which learns to provide appropriate commands to these mid- and low-level policies.

High-level Decision Making

We achieved high-level decision making, the correct sequence and choices of robot actions, through training a Deep Neural Network that approximates the action value function $Q(s, a)$ over the environment and chooses the action a which yields the highest value in state s.

Double Q-learning

We used Double Q-learning [115] to train a Q-network $Q(s, a; \phi)$ parametrised by weights ϕ to approximate the true action value function $Q(s, a)$. At

run-time, the action a is obtained as the argument of the maximum Q-value $\text{argmax}_a\, Q(s, a; \phi)$ in state s.

In Double Q-Learning, two separate Q-networks Q_1, Q_2 are used for action selection and value estimation. Compared to Deep Q-Learning [116], where one Q-network is used to conduct both action selection and action value estimation, using two separate Q-networks here can improve training stability [115].

The network parameter ϕ_i is obtained by $\min_{\phi_i} L(\phi_i)$:

$$\min_{\phi_i} E[(r + \gamma Q_j(s', a^*; \phi_j) - Q_i(s, a; \phi_i))^2], \tag{3.1}$$

with reward r, discount factor γ, network parameters ϕ_i, ϕ_j, Q-networks Q_i, Q_j, current state s, next state s', best action $a^* = \text{argmax}\, Q_i(s, a; \phi_i)$. During training, either network parameters ϕ_1 or ϕ_2 is randomly selected, trained, and used for action selection, while the other network parameter is used to estimate the action value. The tuple $(s, a, r, s') \sim U(D)$ is obtained from the Experience Replay by uniformly sampling from buffer D, which is filled by rolling out the actions on the robot.

Training Procedures

The high-level policy sends and updates the actions $a \in \mathcal{A} \subseteq \mathcal{R}^9$ at $1Hz$ frequency, which are the positions in Cartesian space for the pelvis $a_{\text{pelvis}} \in \mathcal{R}^3$, left and right hands $a_{\text{lh}}, a_{\text{rh}} \in \mathcal{R}^3$. The actions a are executed by the mid-level stability controller. The locomotion policy walks towards the pelvis targets p_{Pelvis}, while the manipulation policy places the hands to the hand targets $p_{\text{lhand}}, p_{\text{rhand}}$.

The states $s \in \mathcal{S} \subseteq \mathcal{R}^9$ are the vector $\vec{s}_{\text{pelvis}} = a_{\text{pelvis}} - p_{\text{Pelvis}} \in \mathcal{R}^3$ from current pelvis position p_{Pelvis} to the pelvis target a_{pelvis}, the vector $\vec{s}_{\text{hands}} = a_{\text{lh, rh}} - p_{\text{lh, rh}} \in \mathcal{R}^6$ from current hand positions $p_{\text{lh, rh}}$ to left and right hand targets $a_{\text{lh, rh}}$. Lastly, three boolean variables are provided as the observation state when the door is open, the box is on the table, and the box is in the hand.

The reward terms r_i are determined based on the task completion, such as whether the robot has passed the delivery table, the arm joints are in nominal position, the box is between the robot hands, the box is at the delivery table, the door is open, and whether the robot is at the goal. The weights $w_i, i = 1, ...6$ can be found in Table 3.3.

At every time step, the reward r is the sum of sparse, boolean states:

$$r = w_1 r_{\text{pt}} + w_2 r_{\text{jn}} + w_3 r_{\text{bih}} + w_4 r_{\text{bot}} + w_5 r_{\text{do}} + w_6 r_{\text{ag}},$$

with passed table reward r_{pt}, joints nominal reward r_{pt}, box in hand reward r_{pt}, box on table reward r_{pt}, door open reward r_{pt}, and at goal reward r_{pt}.

	w_1	w_2	w_3	w_4	w_5	w_6
Value	1	0.1	1	2	2	5

Table 3.3: Weights for high-level reward

We perform early termination of the episode if the robot falls, or collided with itself, tables, or the door. Early termination discourages the policy from entering states and actions that would lead to early termination

as the cumulative reward becomes low, due to reaching the end of an episode.

We initialise the robot in different positions of the environment to allow the robot entering states that are hard to discover merely by exploration. The robot was spawned at the configurations that are close to the final goal, in front of the open or closed door, and in front of tables.

Mid-level Stability Control

The mid-level stability control layer consists of two components: the manipulation policy for the arms is realised as Model-Predictive Control (MPC) scheme, and a locomotion policy for the legs is learned through Deep Reinforcement Learning.

Manipulation Policy

The manipulation policy receives Cartesian target positions for the hands as reference from the High-level Decision Making policy, and outputs joint positions to the Low-level Joint Controller. This is achieved by combining two parts (Fig. 3.7): Model-Predictive Control (MPC) that generates a stable, optimal trajectory in Cartesian space and Inverse Kinematics (IK) [117] that transforms desired actions from the Cartesian space to the joint space.

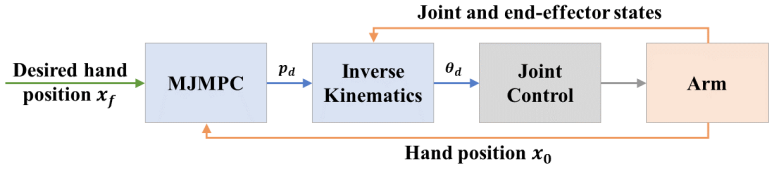


Figure 3.7: Control diagram of the manipulation policy. The Minimum Jerk Model-Predictive Control scheme provides a Cartesian space trajectory which is transformed into the joint space by Inverse Kinematics.

To provide the smoothest possible motions for the hands, we formulated the optimal control problem as the minimum-jerk optimisation, while satisfying dynamics constraints on the hands. The optimal trajectory is then implemented in an MPC fashion. The MPC control applies the first control input of the optimal input trajectory and then re-optimise based on the new state at the next control loop [118]. In this way, MPC successively solves an optimal control problem over a prediction horizon N and achieves feedback control, while ensuring the optimality.

For the hand position x, an objective function J is designed to minimise jerk $\dddot{x}$ (the input u of the system) with final time t_f:

$$J = \frac{1}{2}\int_0^{t_f} \left(\frac{\mathrm{d}^3 x(t)}{\mathrm{d}t^3}\right)^2 \mathrm{d}t = \frac{1}{2}\int_0^{t_f} u(t)^2 \mathrm{d}t. \tag{3.2}$$

The Minimum Jerk MPC (MJMPC) solves the following constrained

optimisation problem at very time step at a frequency of $25Hz$:

$$\begin{aligned}
\min_{u(t)} \quad & \frac{1}{2}\int_0^{t_f} u(t)^2 \mathrm{d}t \\
\text{subject to} \quad & \frac{\mathrm{d}^3 x(t)}{\mathrm{d}t^3} = u \\
& [x(0), \dot{x}(0), \ddot{x}(0)] = [x_0, \dot{x}_0, \ddot{x}_0] \\
& [x(t_f), \dot{x}(t_f), \ddot{x}(t_f)] = [x_f, \dot{x}_f, \ddot{x}_f] \\
& [x_{\min}, \dot{x}_{\min}, \ddot{x}_{\min}] \leq [x, \dot{x}, \ddot{x}] \leq [x_{\max}, \dot{x}_{\max}, \ddot{x}_{\max}], \qquad (3.3)
\end{aligned}$$

with initial condition $[x_0, \dot{x}_0, \ddot{x}_0]$, and terminal condition $[x_f, \dot{x}_f, \ddot{x}_f]$.

The resulting Cartesian trajectory p^d from MJMPC is transformed into joint position commands θ^d through Inverse Kinematics (IK). More formally, IK describes a transformation $T : \mathcal{C} \to \mathcal{Q}$ from Cartesian space $\mathcal{C}$ to joint space $\mathcal{Q}$ [117]. The joint position commands θ^d are then tracked by the low-level joint position controller as described in Section 11.

Locomotion Policy

The locomotion policy $\pi(s;\theta)$ coordinates the 12 Degree of Freedom (DoF) leg joints and is represented as Deep Neural Network (network parameters θ) that receives robot states s as inputs and outputs 12 target joint positions for legs. It is trained through Soft-Actor Critic (SAC) [62] – an off-policy Deep Reinforcement Learning (DRL) algorithm.

SAC optimises a maximum entropy objective $J_{\mathrm{SAC}(\pi)}$:

$$J_{\mathrm{SAC}}(\pi) = \sum_{t=0}^{T} \mathbb{E}[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))], \qquad (3.4)$$

with reward r, state s_t and action a_t at time t, temperature parameter α, and policy entropy $\mathcal{H}(\pi)$.The parameters θ for policy π_θ are obtained by minimising $J_\pi(\theta)$:

$$J_\pi(\theta) = \mathbb{E}[\log \pi_\theta(a_t|s_t) - Q_\phi(s_t, a_t)]. \qquad (3.5)$$

The action value function $Q_\phi(s_t, a_t)$ is obtained by minimising the Bellman residual $J_Q(\phi)$:

$$J_Q(\phi) = \mathbb{E}[\tfrac{1}{2}(Q_\phi(s_t, a_t) - \hat{Q}(s_t, a_t))^2], \qquad (3.6)$$

with Bellman equation $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}[V_\psi(s_{t+1})]$ and discount factor γ. An estimation of value function V_ψ is obtained through minimising $J_V(\psi)$:

$$J_V(\psi) = \mathbb{E}[\tfrac{1}{2}(V_\psi(s_t) - \mathbb{E}[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t|s_t)])^2]. \qquad (3.7)$$

The training process of policy π_θ is detailed in Algorithm 1.

The training procedures, including the design of reward, action space, and state space, are as in [67]. The action space $\mathcal{A} \in \mathcal{R}^{12}$ are the joint positions of the 12 Degree of Freedoms (DoF) of the legs (for each leg 3 DoF hip, 1 DoF knee, 2 DoF ankle). The target joint positions are tracked by the low-level joint controller (see Section 11).

Algorithm 1: Pseudocode for SAC

$\mathcal{D} \leftarrow \emptyset$, initialise replay buffer
for iter=1,2,... **do**
  while collected samples < batch size **do**
    Sample and perform action $a_t \sim \pi_\theta$
    Collect $d = \{s_t, a_t, r(s_t, a_t), s_{t+1}\}$
    Store sample d in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup d$
  for policy update=1,...,K **do**
    Sample batch from replay buffer $\mathcal{D}$ for update
    Update policy π_θ via (3.5)
    Update action value function Q_ϕ via (3.6)
    Update value function V_ψ via (3.7)

The state space $\mathcal{S} \in \mathcal{R}^{27}$ consists of the desired pelvis position/reference (the walking destination), and proprioceptive feedback states of the robot including pelvis orientation, linear and angular velocity of the pelvis, force of both feet, joint positions of the legs, and the gait phase.

The reward comprises of an imitation term and a task term similar to [67]:

$$r_i = w_i r_{\text{imitation}} + w_t r_{\text{task}}, \tag{3.8}$$

with weights w_i, w_t and reward terms $r_{\text{imitation}}, r_{\text{task}}$ for imitation and task respectively.

The aim of $r_{\text{imitation}}$ is to imitate the joint position, feet pose, and contact pattern of a reference motion capture trajectory as close a possible. The reward term r_{task} rewards upright posture, short distance to the goal position, and regularises the joint velocity and torque.

Low-level Joint Control

The low-level joint control tracks target joint positions θ^d provided by the mid-level stability controller. It is realised as *Joint Impedance Controller* that regulates around the set point. Additionally, a feed-forward controller (*Inverse Dynamics* in Fig. 3.8) provides joint torques τ^{FF} that compensates for the dynamical influences from gravity and Coriolis forces and hence achieves more accurate tracking of the desired joint motions θ^d.

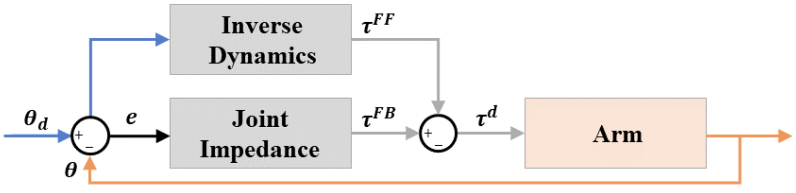


Figure 3.8: Dual-loop control structure for low-level joint control: Inverse Dynamics (feedforward) loop and Joint Impedance Control (feedback).

Feedback Control The desired control effort is calculated using stiffness K_{P_i} and damping gains K_{D_i}. The joint impedance control calculates desired joint torque τ^d using position θ and its derivative

$\dot{\theta}$:

$$\tau^d = K_{P_1}(\theta^d - \theta) - K_{D_1}\dot{\theta}. \tag{3.9}$$

At the actuator level, the motor driver implements an internal current control to track the desired joint torque τ^d using a proportional-derivative law, where the desired motor current I is computed as:

$$I = K_{P_2}(\tau^d - \tau) - K_{D_2}\dot{\tau}. \tag{3.10}$$

Feedforward Control

The feedforward torques τ^{FF} are computed through Inverse Dynamics [117], which transforms $T : \Theta \to \mathrm{T}$ from desired joint motions $\theta^d \in \Theta$ to joint torques $\tau^{FF} \in \mathrm{T}$, based on the equations of motions satisfying the Newton–Euler dynamics of a multi-link rigid bodies given the joint configuration $\theta^d, \ddot{\theta}^d$. The resulting joint torques τ^{FF} compensate for gravity and Coriolis forces given the desired joint configuration θ^d.

Multi-Expert Learning of Adaptive Locomotion Behaviours

4

In this chapter, we present a Multi-Expert Learning Architecture (MELA) that enables the synthesis of multiple experts to learn adaptive skills from a set of existing ones [2]. In particular, we show how MELA can be used to learn agile and robust maneuvers on a quadruped robot. Please find the n-th supplementary figure and table denoted by fig. Sn and table Sn in the Appendix J. The media attachment can be found in https://robotics.sciencemag.org/content/suppl/2020/12/07/5.49.eabb2174.DC1.

4.1 Introduction

Adaptive motor skills enable living organisms to accomplish sophisticated 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 behaviors is a long-standing scientific theme for biologists and roboticists. From a neurological perspective, it is worth

Figure 4.1: Challenging locomotion scenarios and agile maneuvers 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 part of the body. (B) Different adaptive behaviors from our proposed learning framework that generated dynamic motions and complex coordination of legs for immediate recovery from failures. Time in snapshots is measured in seconds. (C) Resilient locomotion using adaptive skills in the presence of unexpected disturbances.

understanding how the sensorimotor control system in animals processes various sensory information and produces adaptive reactions in unseen situations [119, 120, 121, 122]. From a robotics perspective, it is interesting to take a bioinspired approach and transfer biological principles, such as primitive neural circuits, to produce robot behaviors similar to that of animals [119]. Because the underlying mechanisms of the motor cortex cannot yet be fully replicated [122, 123], we take the latter approach by drawing inspiration from biological motor control to develop learning algorithms that can achieve skill adaptation for robot locomotion [124].

In this study, we investigate how an artificial agent can learn to generate multiple motor skills from a set of existing skills, particularly for critical tasks that require immediate responses. As an ex- ample of learning a complex physical task, playing soccer consists of several subskills, such as dribbling, passing, and shooting. During training, players first practice the most important subskills separately. Once mastered, all different subskills are used in a flexible combination to improve all these techniques. Our research in multiskill learning has studied skill-adaptive capabilities such as these, and we draw inspiration from animal motor control when designing the learning and control architecture. Using a quadruped robot as a test bed, we aim to produce adaptive robot behaviors to succeed in unexpected situations in a responsive manner.

Background

The DARPA Robotics Challenge (DRC) from 2012 to 2015 fostered the development of semiautonomous robots for dangerous missions such as disaster response in unstructured environments [125]. Most DRC robots had different forms of legged design for the dexterity to traverse irregular surfaces. Despite the tremendous engineering ef- forts, no robot could recover autonomously from falls [126]. To date, most legged robots still lack such an autonomous ability to generate adaptive actions to deal with unexpected situations.

Because of the uncertainties in unforeseen situations, locomotion failures are likely to happen. We illustrate these challenges in robot locomotion using field tests (Fig. 4.1A) and the adaptive behaviors that are robust to uncertainties (Fig. 4.1 B and C). Typically, falling occurs within a second of the robot losing balance, and the time window for fall prevention is around 0.2 to 0.5 s. Therefore, it is critical to immediately coordinate different locomotion modes to mitigate perturbations and prevent or recover from failures. In comparison with robot systems, biological systems (e.g., cats, dogs, and humans) exhibit higher versatility [127], and the key to the performance gap is the difference in motion intelligence, which allows biological systems to handle changing and complex situations [127, 128, 129].

Our paper studies a machine learning approach that learns reactive locomotion skills and generates adaptive behaviors 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.

Related work

When a robot interacts with its environment, it can be difficult to determine which part of the robot is in contact; this is challenging for classical control solutions because careful modeling of contacts is often needed. The main approach in the legged locomotion community uses model-based mathematical optimization to solve these multicontact problems, such as model predictive control (MPC), whole-body quadratic programming (QP), and trajectory optimization (TO). To achieve fast online computation, MPC uses simplified models and short predictive horizons to plan task-space motions for walking [130], running [131], and pacing [132]. QP methods are used for whole-body control to map task-space motions (e.g., those from the MPC) to joint-space actions while considering the whole robot model and physical constraints [133, 20]. A unified but computationally expensive technique is to optimize all models and constraints together (whole robot model, contact model, environment constraints), i.e., through nonlinear MPC (NMPC) [134] and whole-body optimization [135, 136, 137].

In the optimization 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 [138, 139]. Furthermore, the explicit properties of the robot and the environment need to be modeled [140] but are expensive to compute [33] and thus difficult to run in real time in complex settings even with exhaustive computing [141]. This fundamental principle suffers from the curse of dimensionality and therefore limits the scalability to real-time solutions in more complex and challenging problems [142].

To this end, deep reinforcement learning (DRL) becomes attractive for acquiring task-level skills: Through rewarding intended outcomes and penalizing undesired ones, an artificial agent can learn desirable behaviors [138, 139]. Using DRL offers several advantages: The training process can be realized 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 [143, 144] and, in recent studies, on hardware [145, 146, 147], e.g., demonstration of learning-based control on a real robot using separate policies for fall recovery and walking [66]. However, similar to other DRL approaches, the learning policies in [66] were specialized 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 multiple experts' existing knowledge [148]: Experts are trained to encode low-level motor primitives, whereas a high-level policy selects the appropriate expert [83, 85]. However, generating new skills cannot be achieved in the standard HRL framework because only one expert is selected at a time. This problem can be addressed by learning to continuously blend

high-dimensional variables of all experts [75]. One related approach is the mixture of experts (MoE) that synthesizes the outputs of individual experts specialized on subproblems using a gating function [77], which has been used in robotics [80], computer vision [79], and computer graphics [82]. However, compared with blending the experts' high-dimensional variables, MoE has known limitations in scaling to high degree-of-freedom (DoF) systems [82] and exhibits expert imbalance problems, i.e., favoring certain experts while degrading others [149].

Our approach

We draw inspiration from biological motor control to design our control and learning framework. Biological studies suggest that motor behavior is controlled by the central nervous system (CNS) that resets the reference position of body segments, and the difference between the reference and actual position excites the muscular activities for generating appropriate forces [150]. This precludes the need to com- pute the inverse dynamics, simplifies control, and minimizes the computation [151]. Because the spring-damper property provided by the impedance control resembles the elasticity of biological muscles, we applied the equilibrium-point (EP) control hypothesis 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 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 force interactions with the environment (see Materials and Methods). By doing so, we can focus on developing the learning algorithms at the top layer to achieve motor intelligence.

Contributions

This work aims to demonstrate how hierarchical motor control using DRL can achieve a breadth of adaptive behaviors for contact-rich locomotion. We propose a multi-expert learning architecture (MELA) that contains multiple expert neural networks, each with a unique motor skill, and a gating neural network (GNN) that fuses expert networks dynamically into a more versatile and adaptive neural network.

Compared with the approach of using kinematic primitives [152, 153], the proposed MELA policy indirectly modulates the joint torques by changing the reference 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 [83, 85], MELA continuously combines network parameters of all experts seamlessly and is therefore more responsive than other methods because there is no wait time due to disjointed switching of experts. In addition, because MELA synthesizes the experts in a high-dimensional feature space, i.e., weighted average of the network parameters (weights and biases of the neural networks), it

does not suffer from the same expert imbalance problems as MoE [149]. Similar multi-expert structures that blend experts in high-dimensional feature space were studied in computer graphics for kinematic animation [149, 154, 155] 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) generates multiple distinctive skills effectively, (ii) diversifies expert skills using cotraining, and (iii) synthesizes multiskill policies with adaptive behaviors in unseen situations. The adaptive behaviors are verified by proof-of-concept experiments on a real robot and various extreme test scenarios in a physics simulation. By synthesizing multiple expert skills, the collective expertise of MELA is more versatile than each single expert, because of the dynamic structure of MELA, which integrates all experts based on the online state feedback. Such multi-expert learning allows each expert to specialize in unique locomotive skills, i.e., some prioritize postural control and failure recovery, while others acquire strategies for maximizing 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 in achieving reactive and adaptive motor behaviors to changing situations consisting of unforeseen scenarios, unexpected disturbances, and different locomotion modes, which have not been addressed well by other unified frameworks in the literature. We will show the advantages of our proposed work, which can produce a variety of strategies to ensure task success. For example, the robot can produce quick responses to recover balance in different unexpected postures and stabilize dynamic transitions. This is an indicative level of machine intelligence—the ability to autonomously produce locomotive skills that would require substantial intelligence to design if they needed to be programmed by humans.

4.2 Results

We summarize the results and the learning method in Movie 1. In the following sections, we report the results of adaptive locomotion behaviors, followed by technical details of the learning framework.

Multi-expert learning framework

We first define key terminology to help explain the concepts in this article: motor skill, expert, and locomotion mode. Motor skill (or "skill" for short): a feedback policy that generates coordinated actions to complete a specific type of task; this serves as a building block for constructing more complex manoeuvres. Expert: a deep neural network with specialised motor skills. Locomotion mode: a combination of limb movement and patterns, such as standing, turning on the spot, trotting forwards/backwards, steering left/right and fall recovery.

To complete tasks in new scenarios, an artificial agent needs the ability to adapt relevant skills during runtime, which is why we propose the use of MELA: a hierarchical reinforcement learning structure which

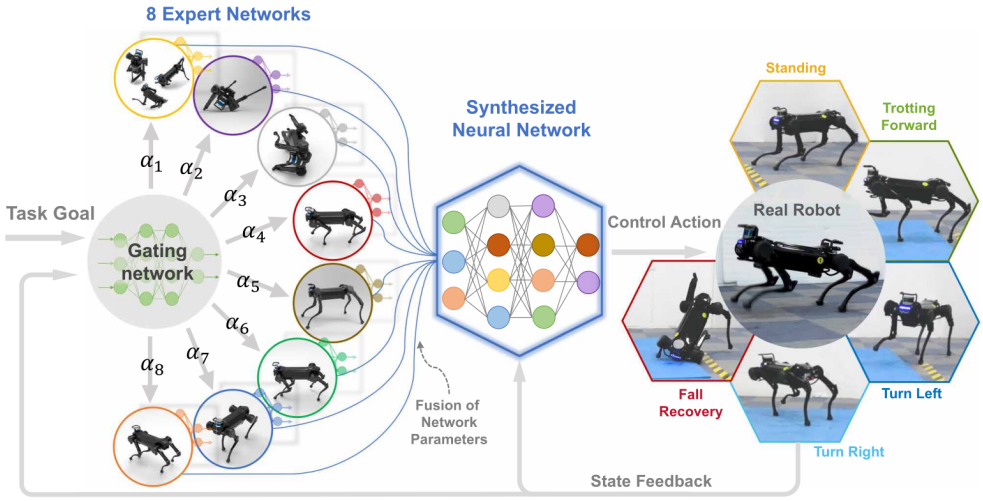


Figure 4.2: Multi-Expert Learning Architecture (MELA): a hierarchical deep reinforcement learning framework that generates new behaviour by combining multiple deep neural networks (DNNs) together to produce versatile locomotive skills. The Gating Neural Network (GNN) generates variable weights (α) to 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 consortium of experts.

consists of a collection of Deep Neural Networks (DNNs) and a Gating Neural Network (GNN). As shown in Fig. 4.2, the GNN continuously fuses expert DNNs into a single 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 (see Materials and Methods). Through repurposing and combining 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 a set of policies, where each one accomplishes a distinct task. 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 GNN 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 (see Materials and Methods). 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. 4.2): 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 minimum of 7 DNNs can be used 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 using 8 experts in total. Further comparison found that using more than 8 experts had no further performance but more training time (fig. S1). It shall be noted that the 7 situations are only used as a guide to determine the number of experts at the initial design process, and does not need to match the order of numbering of MELA's learned motor skills in the latter section.

Learning individual motor skills - Fall recovery and trotting

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 [75, 82]. Like training for sports, it is essential to practise individual skills that are distinctly different from each other. Similarly, MELA has a two-step training procedure for 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.

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

We evaluated the robustness of recovery policies and categorised the learned reactive behaviours into four strategies (Fig. 4.3A): (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 where the robot exploits its 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. 4.3A1. Active righting is the strategy where the robot pushes itself using the leg and elbow, creating momentum to actively flip itself to a prone position, as shown in Fig. 4.3A2. Stepping emerges when necessary during standing to regain balance, involving coordination and switching of support legs. An example of stepping is shown in Fig. 4.3A3, and such multi-contact switching was all generated naturally using the learned policy based on the online state feedback.

From all fall recovery experiments (Fig. 4.3A and movie S1-2), we can see the responsive and versatile reactions, including the emerged stepping behaviour. Compared to baseline fall recovery which is manually engineered with a fixed pattern (fig. S2), our learning policy is able to recover from various fall scenarios because it can respond to dynamic changes using online feedback, while the handcraft controller only addresses a narrow range of situations.

The Jueying robot can robustly trot under three ground conditions with different stiffness, friction, and obstacles (Fig. 4.3B and movie S3). In Fig. 4.3B1, the concrete ground was covered by thin carpets with high friction, while in Fig. 4.3B2, 2cm thick foam mats were laid, creating a softer and more slippery surface. In Fig. 4.3B3, 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 using 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., $\tau = K_p(q^d - q^m)$ [156]. The expert has indirectly learned active compliance control by regulating the references based on feedback of the current joint positions to minimise joint torques.

Multi-expert skills for multimodal locomotion

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 into 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 to respond 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 tested on a real robot, which demonstrated the capabilities of achieving a diversity of locomotion tasks, adapting to external environmental changes responsively, whilst also following 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 cover the wide range of sensory inputs. Without hardware risks, we operated the robot in extremely dynamic motions to activate different experts, allowing analysis of all distinct motor skills. This provides data of variable weights that reflects the activation level of all experts over each motor skill. Figure 4A 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. 4.4A, 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.

The 10 analysis 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. 4.4 (A) Specialised activation of eight experts across different motor skills, where the distinct activation patterns indicate a unique specialisation (table S1). Fig. 4.4 (B-C) The 2D projection of the gating network's activation pattern by t-SNE, where the neighbourhoods and clusters of the samples are visualised. Samples representing similar activation appear in close proximity, whereas the different ones are distant from each other. Fig. 4.4 (B) The output samples of the gating network, which are classified by the index of the dominant expert that has the highest activation (Fig. 4A). Fig. 4.4 (C) The output samples of the gating network, which are classified by the physical states during a distinct locomotion mode, e.g., trotting, balancing, turning left/right. Fig. 4.4 Fig. 4.4 (D-E) The 2D projection of the actions from the pre-trained, co-trained, and synthesised expert policies using t-SNE analysis: Fig. 4.4 (D) an Fig. 4.4d (E) are the target actions classified during fall recovery and trotting tasks, respectively.

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 motor skill specialisation of experts 1 to 8 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 be found in fig. S3 where experts are sorted by activation patterns across different motor skills.

Analysis of skill adaptation and transfer

Apart from the skill specialisation (Fig. 4.4A), 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. 4.4B-E), and vice versa.

In Fig. 4.4B-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. 4.4B and Fig. 4.4C, which are grouped by dashed lines corresponding to locomotion (green) and fall recovery (bronze), suggesting that the gating network perceives these two as different modes. The t-SNE analysis is labelled according to the experts (Fig. 4.4B) and motor skills (Fig. 4.4C), 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. 4.4A.

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. 4.4D-E, the actions generated from eight 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 skills and further acquired more profound and newly emerged skills during MELA's co-training stage. The data in Fig. 4.4D-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 partially preserves the original skills, which are reconstructed by fusing eight distinctive experts.

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. 4.5A); (ii) double-mode left-right steering on the spot (Fig. 4.5B); (iii) triple-mode of simultaneous left-right steering and trotting (Fig. 4.5C); and (iv) target-following locomotion involving all modes, i.e., standing, left-right steering, trotting and fall recovery (movie S4-5).

In our study, adaptive behaviours refer to the online synthesised skills that adapt reactively 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 (Fig. 4.5 and fig. S4A) and variable-speed trotting (fig. S5); (ii) Transitional skills that coordinate dynamical transitions smoothly between different locomotion modes, e.g., transition from

various failure poses to trotting (Fig. 4.5 and fig. S4B-E). Five representative cases of the adaptive behaviours from the MELA policy can also be found in fig. S4.

Figure 5A shows successful fall recovery performed by the MELA policy, and the similarity with those in Fig. 4.3A indicates that the MELA policy has reused some pre-trained skills. Figure 5B 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 (Fig. 4.5B), 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 (fig. S6). 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 5C and 5D show target-following tasks requiring simultaneous trotting and steering on the real robot. The task was to chase a virtual target given by a user command, i.e., a variable position vector with respect to the robot. In Fig. 4.5C, a small target position ahead of the robot was provided, e.g., 0.28 m in the heading direction (fig. S7A), and the robot performed left/right turning while trotting forward. In the experiment shown in Fig. 4.5D, a farther target position of 0.48 m was commanded (fig. S7B), and the robot was chasing a distant target and trotting at larger steps. Torque saturation of motors occurred more often on the lab floor (fig. S8), and the robot had three successive tripping and recovery incidents. Key snapshots around falling and reactive responses are in Fig. 4.5D. Once the states of gait failures were sensed, MELA was able to produce immediate reactions to restore balance within one second (fig. S9), and the robot recovered from tripping and continued trotting without human intervention. Figure 5E shows an outdoor experiment where the robot first recovered from a falling posture and was later knocked down during a long walk on the grass. MELA was able to recover the robot from fall and resume trotting successfully (fig. S10). From all experiments, the synthesised MELA expert demonstrated flexible motor skills, coordinated movements and smooth transitions, showing how crucial it is to have such reactive responses and feedback control for robust and autonomous locomotion.

As shown in Fig. 4.5, MELA enabled the robot to complete all validation successfully, and demonstrated dynamic fall-resilient locomotion. The gating network in MELA has learned how to generate variable weights for all experts in response to the state feedback and to provide smooth transitions across all modes, and see data analysis of the cases (Fig. 4.5D-E) in fig. S11 and fig. S12. 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. Additional analysis of the gating pattern and the relationship between experts is presented in the Supplementary Materials.

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, a moving slope, rough

terrain (fig. S13) as well as robustness tests with variations of masses and motor failures (fig. S14). During successful locomotion in these unseen scenarios, the MELA policy performed versatile adaptations to new situations (movie S6 and S7). We note that the MELA framework has learned how to deal with transitions at various gait phases (fig. S15-19), and the synthesised policy is different from the eight basic motor skills which indicates a nonlinear interpolated behaviour among expert skills (fig. S20). All these experiments and simulations validate MELA's capability of producing flexible behaviours in a variety of novel situations.

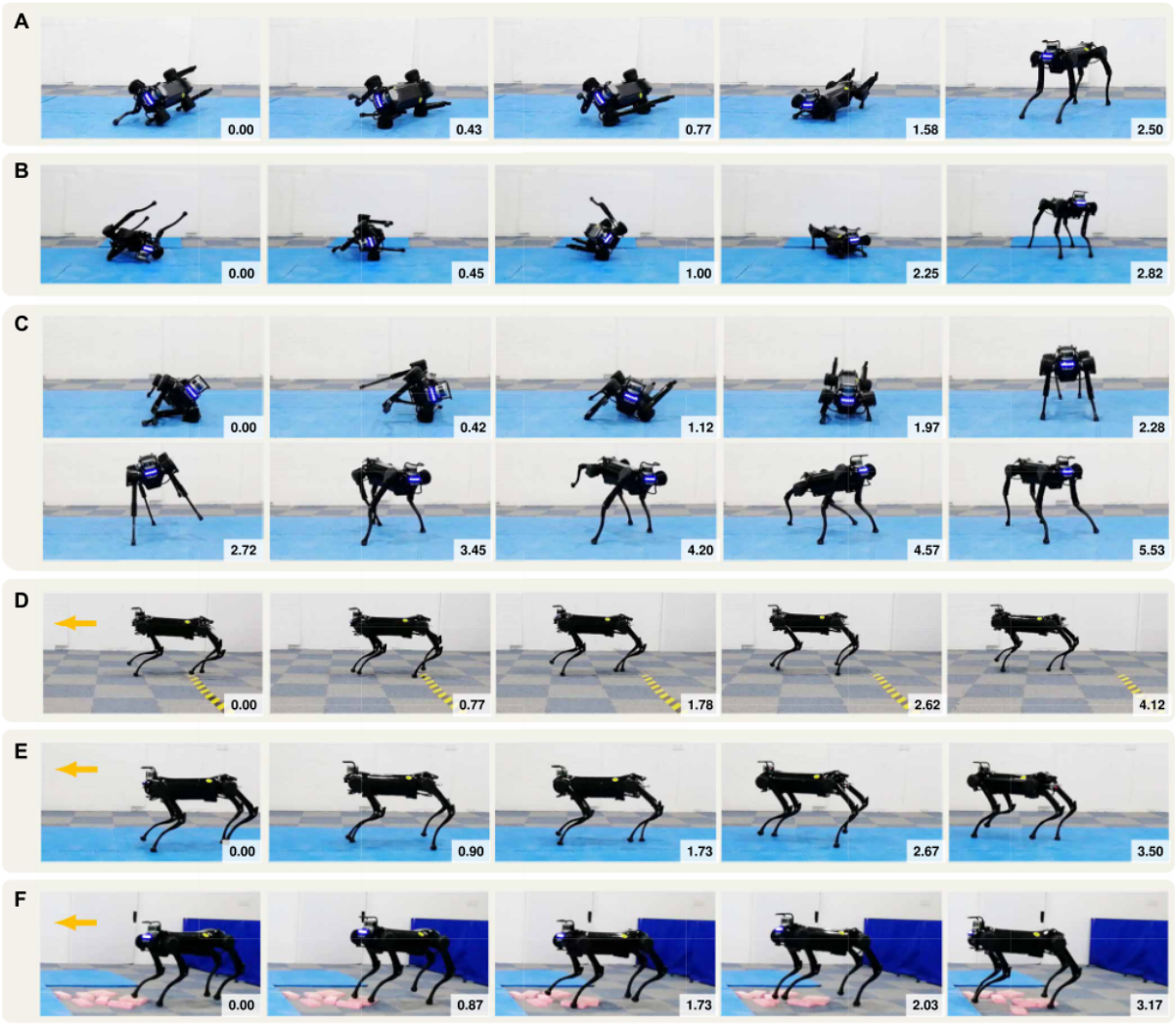


Figure 4.3: Individual motor skills for the fall recovery and trotting. (A) A configuration between prone and lateral decubitus positions where legs were stuck underneath the body: The robot first pushed the ground to lift up the body for ground clearance and then retrieved legs to a prone posture for standing up. (B) The robot actively used elbow-push to generate a large momentum to self-right to a prone position. (C) A stepping behavior was learned and performed naturally to keep balance. (D) Stable trotting on a hard floor. (E) Stable trotting on soft slippery foam mats. (F) Stable trotting over scattered obstacles, showing the compliant interaction and robustness learned by the trotting expert. Time in snapshots is measured in seconds

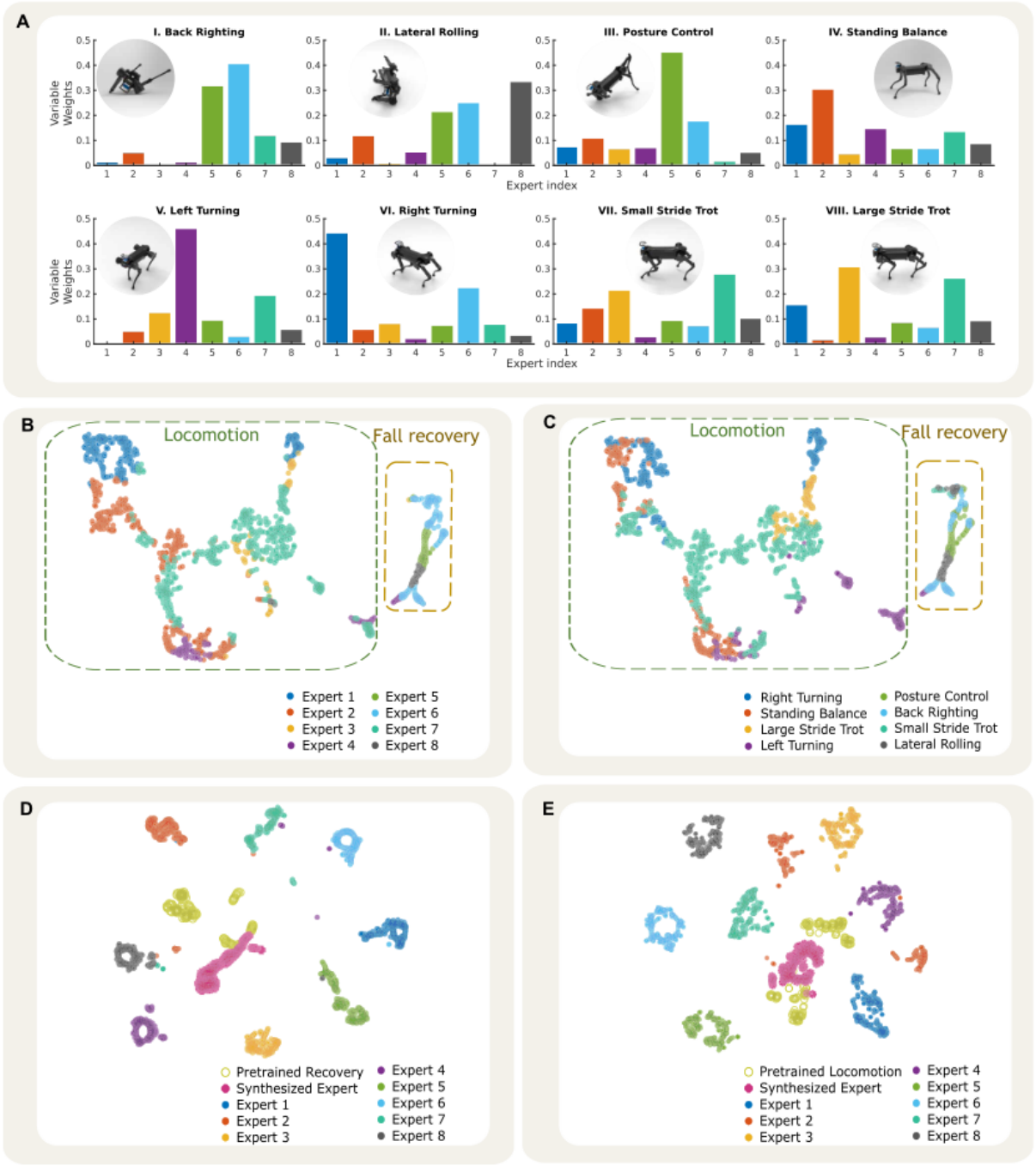


Figure 4.4: Analysis of the specialization of experts across different motor skills, and the patterns of the gating network and the expert networks using the t-SNE. The analysis was based on the simulation tests using the policy from a single training run, which was representative because the training process can consistently reproduce policies with very similar characteristics of skill specialization. (A) Specialized activation of eight experts across different motor skills, where the distinct activation patterns indicate a unique specialization (table S1). (B and C) 2D projection of the gating network's activation pattern by t-SNE, where the neighborhoods and clusters of the samples are visualized. Samples representing similar activation appear in close proximity, whereas the different ones are distant from each other. (B) Output samples of the gating network, which are classified by the index of the dominant expert that has the highest activation (A). (C) Output samples of the gating network, which are classified by the physical states during a distinct locomotion mode, e.g., trotting, balancing, and turning left/right. (D and E) 2D projection of the actions from the pre trained, cotrained, and synthesized expert policies using t-SNE analysis: (D) and (E) are the target actions classified during fall recovery and trotting tasks, respectively.

Figure 4.5: Dynamically synthesised MELA policy running on a real quadruped robot and demonstrating adaptive and agile locomotion behaviours. (A) Successful fall recovery performed by the MELA expert, inheriting original skills from the pre-trained experts. (B) Newly emerged skills, dynamic steering on the spot, naturally learned through the MELA framework. (C) Target-following experiment with simultaneous trotting and steering using online synthesised skills. (D) 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). (E) Adaptive trotting and fall recovery experiment on a sloped grass field, showing the capability of robust traversal in unstructured environments. (Time in snapshots is in second).

4.3 Conclusion

This study aims to achieve versatile robot motor skills in contact-rich multimodal locomotion. In contrast to most solutions which are dedicated to separate narrow-skilled tasks, we approached this challenge with 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 mission success in real-world applications.

In comparison with MoE, MELA's approach of fusing network parameters prevents the expert imbalance problem and provides diversity among 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 consortium 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 the weights of each specialisation according to different conditions; and the composed synthesised MELA expert partially preserves some of the original skills. We implemented the MoE approach using the same pre-trained experts and training procedure, and the MoE policy exhibited degenerated experts in steering skills which led to asymmetric gait and turning behaviour (note S3 and fig. S21).

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.

4.4 Limitations and future work

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 [157], dynamic manoeuvres [158], and utilisation of affordance to coordinate whole-body support poses [159]. 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 locomotion modes, 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.

4.5 Materials and Methods

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

Robot platform

We implemented our learning algorithms on the Jueying quadruped robot [21] 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 are actuated by brushless electric motors with low gear ratio (i.e., 7:1) and high-fidelity joint torque control (table S2).

Deep reinforcement learning framework

The goal of the DRL is to train an artificial agent to infer optimal actions from the current state by learning from past experience. The experience samples are stored as tuples containing the current state s_t, the action generated from policy $a_(t) \sim \pi_t(s_t)$, the reward r_t, and the next transition state s_{t+1}. These samples are first collected and stored in a Replay Buffer, from which they are drawn later to train the policy. We used the Soft Actor Critic (SAC) algorithm [62], and below are the details of reward, state, action, training procedures and loss function.

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), trotting requires reward (i) to (xii), and multimodal target-following locomotion 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 S3 and table S4.

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 [67]. 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 obtained from the state estimation by fusing the leg kinematics and the accelerations 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 (fig. S22). 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 S5.

Action space

The benchmark of DRL-based locomotion [66, 160] shows 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 [66, 160], 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) the use of low-pass filters to emulate the characteristics of the frequency response of actuators, which enforces physically realisable reference motions, and (ii) the design of a particular loss function to generate smooth and non-jerky joint references and torques. We named these two techniques action filtering and smoothing loss, respectively.

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 [161], 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 for 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 (fig. S23). 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, all obtained policies exhibited smooth motions within the bandwidth (fig. S23) that can be executed on the real robot directly and safely.

Smoothing loss

The action filtering alone may not always guarantee 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 minimise the applied torques [150].

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 $J_{\text{smoothing}}$ as:

$$J_{\text{smoothing}}(\mu(s_t)) = \|\mu(s_t) - q\|_2, \tag{4.1}$$

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 $J_{\text{smoothing}}$ is the objective function that minimises the differences between the target $\mu(s_t)$ and the 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 $J_{\text{smoothing}}$ is incorporated into the SAC training loss $J_{\text{SAC}}(\pi)$ and is used for backpropagation of the neural networks, instead of being part of the reward function. Adding the smoothing loss term to $J_{\text{SAC}}(\pi)$ 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.

MELA training procedure

Figure 6 depicts both the network architecture and training procedure of our MELA framework. The MELA network consists of one gating network and eight expert networks (Fig. 4.6B). 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. 4.6A.

Here, we elaborate on MELA's two-stage training procedure. In stage 1, successful trotting and fall recovery policies were pre-trained using the scheme shown in Fig. 4.6A. In stage 2, MELA first initialised two groups of expert networks (four in each group) by copying the weights and bias from the two pre-trained experts (Fig. 4.6B) 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 co-training in stage 2, 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. 4.6B. The robot feedback states were the input of the synthesised network for generating motor actions.

Following the framework in Fig. 4.6, both stages were trained using the SAC algorithm (note S2), 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, and additional details are described in note S2. The learning curves during stage1 and stage 2 of the MELA training can be found in fig. S24.

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 [162].

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

$$\psi_{\text{synth}} = W_0 \in \mathbb{R}^{x \times h}, W_1 \in \mathbb{R}^{h \times h}, W_2 \in \mathbb{R}^{h \times y}, B_0 \in \mathbb{R}^{x}, B_1 \in R^{h}, B_2 \in \mathbb{R}^{y}, \tag{4.2}$$

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

$$\psi^{n}_{\text{expert}} = W^{n}_0 \in \mathbb{R}^{x \times h}, W^{n}_1 \in \mathbb{R}^{h \times h}, W^{n}_2 \in \mathbb{R}^{h \times y}, B^{n}_0 \in R^{x}, B^{n}_1 \in \mathbb{R}^{(}h), B^{n}_2 \in \mathbb{R}^{y}. \tag{4.3}$$

During runtime, the weights W and bias B are fused by the weighted sum formulation as:

$$W_i = \sum_{n=1}^{8} \alpha_n W^{n}_i, B_i = \sum_{n=1}^{8} \alpha_n B^{n}_i, \tag{4.4}$$

where $n = 1, ..., 8$ is the index of experts, $i = 1, 2, 3$ is the index of the corresponding layer, and $\alpha_n \in \mathbb{R}^8$are the variable weights generated by the gating network.

The fused weights W and bias B are used to construct the synthesised network dynamically during runtime using the equation as follows:

$$\phi_{\text{synth}} = \text{Tanh}(W_2 \cdot \text{ReLU}(W_1 \cdot \text{ReLU}(W_0 X + B_0) + B_1) + B_2). \tag{4.5}$$

where $X \in R^{x}$is the input parameter, Tanh(*) and ReLU(*) are the nonlinear activation functions. The weighted sum of weights Wand bias B is normalised to one using a Softmax function, also known as normalised exponential function. There are several nonlinear features in the blending process: each expert DNN is a nonlinear control policy by nature, and each weight produced by the gating network is nonlinearly rescaled by 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.

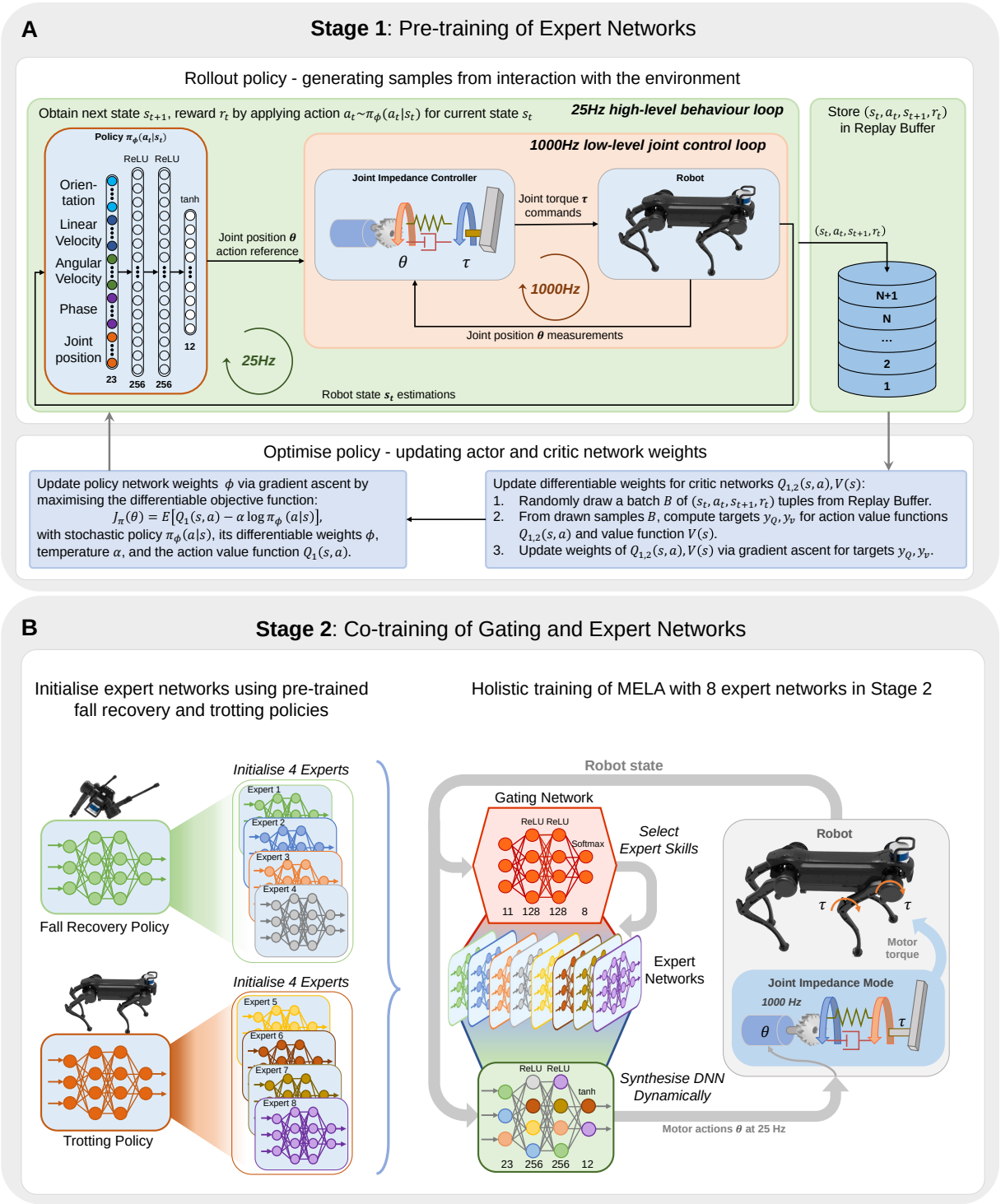


Figure 4.6: Two-stage training of MELA. (A) In stage 1, the fall recovery and trotting policies are individually trained. (B) In stage 2, the pre-trained trotting and fall recovery policies from stage 1 are used to initialise two evenly distributed groups of experts, each containing 4 experts. All these expert networks are co-trained together with the gating network.

Multi-Expert Synthesis for Versatile Locomotion and Manipulation Skills

5

In this chapter, we present a systematic design process for Multi-Expert Synthesis (MES), and propose an automatic state space selection algorithm and show how to explicitly enforce of diversity among multiple experts. We show how MES is used for motor skill synthesis of both robot locomotion and manipulation. The media can be found in https://drive.google.com/drive/folders/1vJUIGlqpbponzBUr1YhaM-Fj6LoMVNKv.

5.1 Introduction

The re-usability of control policies is an interesting and challenging research in robotics. While task-specific controllers and learned policies have been developed to achieve a wide range of tasks, such as locomotion [163, 66] on quadrupeds, as well as manipulation of objects [68]. To create useful versatile robots, the ability to perform multiple tasks is essential, however, combining and reusing these policies in a unified manner remains an open question. This is because a learned policy is trained specifically to solve a particular task which has limited applicability to transfer such a single-skilled policy to other tasks. To have a control policy applicable to other tasks commonly requires redesigning or retraining the policy.

This work is motivated to investigate a systematic approach to formalise the synthesis of multiple learned policies – the multi-expert synthesis (MES) approach that can produce versatile robotic capabilities, in which a high-level behaviour policy recombines multiple skills and fuses more flexible ones, using specialised experts based on the observation of states and task. Here MES is defined as the process of combining or blending expert skills in a latent space. This will produce a multi-expert policy, where a high-level behaviour policy selects the appropriate experts according to the real-time state observation.

The essential principle of multi-expert synthesis lies in designing a hierarchical control architecture where non-relevant information is coordinated and hidden across the layers: while the high-level behaviour

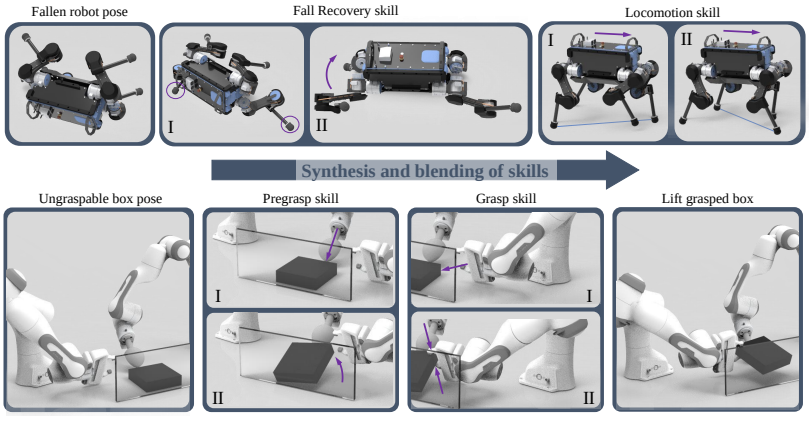


Figure 5.1: Synthesis of expert skills using Multi-Expert Synthesis. **Top**: Versatile locomotion by recovering from a prone position first and then transitioning into trotting. **Bottom**: Dual-arm cooperative manipulation, where the object is first reconfigured by the left hand into a graspable pose and then grasped by the right hand.

policy requires task-relevant information, e.g., distance to a goal or pose of the object to grasp, the low-level expert only requires the proprioceptive state of the robot. Thus, experts focus solely on performing their particular skills, while the high-level policy is responsible for drawing from the experts' skills and completing the task.

This idea originates from Hierarchical Reinforcement Learning (HRL) [72]. In HRL, for discrete and tabular cases, the concepts of temporal [73] and state abstraction [74] are used to determine the information that the components receive. Despite the advantages of encoding non-relevant information across layers, composing and synthesising expert knowledge is not possible in the standard HRL framework since only one expert can be selected at a time. This problem is addressed by learning continuous latent variables that blend the experts in a latent space [75, 76].

The traditional approach for the latent space blending is the Mixture of Experts (MoE), where the outputs of individual experts (actions) specialized on sub-problems are combined by a gating function [77]. The core idea of MoE, combining the outputs of experts via a gating network, has been adapted in the areas of machine translation [78], computer vision [79], robotics [80], Reinforcement Learning (RL) [81], and Computer Graphics [82]. However, MoE has limitations known as expert imbalance – a form of mode collapse that arises due to the lack of diversity across experts [78, 149].

As an alternative to MoE, the Multi-Expert Learning Architecture (MELA) has been proposed, where the latent space is constructed by blending the expert network weights instead of the outputs, thus leading to a higher-dimensional latent space representation of the multi-expert network than MoE [2]. It has been shown that MELA avoids the expert imbalance problem, and provides a framework to use experts for learning unique and diversified skills as well as adaptive behaviours on a quadruped robot.

In this work, we present a systematic approach to best achieve MES, and our proposed algorithms address two key problems: (1) how to select the correct state space representation for the MES feedback policies; and (2) how to avoid the lack of diversity across experts leading to expert imbalance. We will show the results of multi-expert policies for both quadruped locomotion and cooperative bi-manual manipulation (Fig. 5.1).

Specifically in MES, where every expert and the gating network has an individual state space, to correctly design the state space is crucial and non-trivial: missing key state variables will prevent the policy from learning the right action; whereas having too many and unrelated states will result in slower learning, large variances in the reward signals, and higher probability of converging to local minima [164].

Our proposed procedure addresses this problem by the automatic selection of task-relevant variables in state observations. Traditionally, the state space is designed relying on the domain knowledge, and extensive trial and error to find a state space representation that yields highest performance. However, with increasing amount of experts for different skills this becomes time consuming and potentially infeasible. Our

proposed automatic state space design process mitigates the aforementioned state space issues for MES, where multiple state spaces need to be designed for the individual experts and gating network.

Our contributions are summarised as follows:

1. A *systematic design for Multi-Expert Synthesis (MES)* to enable efficient learning of skill coordination and effective synthesis of multiple experts during tasks (see Section 5.4).
2. Formulation of a *state selection algorithm* that uses the learned state value function to quantitatively identify essential states and rule out task-irrelevant variables (see Section 5.4).
3. A proposed *explicit enforcement of diversity among multiple experts* by maximising their discriminability (see Section 23).
4. Applicability of the proposed methods for MES *achieve motor skill synthesis in both robot locomotion and manipulation* (see Section 5.5).

We validate the MES approach in two distinct domains of robotic skills: locomotion and manipulation. For the validation on a quadruped robot, we show that gait recovery and locomotion can be synthesised to achieve robust locomotion (Fig. 5.1 Top). Furthermore, our method can be used for cooperative dual-arm manipulation (Fig. 5.1 Bottom) to grasp an object from a previously ungraspable configuration. The effectiveness of the trained multi-expert policies were validated in a variety of test scenarios in both simulation and real-world experiments.

In the following, we first present related work in Section 5.2. Second, we demonstrate how expert behaviour is obtained (Section 5.3). Next, we present MES, the state selection process, and the types of multi-expert policies in Section 5.4. The results using MES for a quadruped and dual-arm, and a comparison between MoE and MELA is conducted in Section 5.5 and Section 5.6 respectively. Lastly, we conclude the work in Section 5.7.

5.2 Related Work

Model-based control methods have been widely use for robotics tasks, such as locomotion, manipulation, and whole-body control [32, 33]. In particular, recent optimisation based approaches enabled robots to plan [34, 35] and robustly realise planned motions [36] under constraints. This paradigm of using optimisation to first plan and then robustly track the planned trajectory has found wide application in the control of floating-based robots [11, 37, 38, 24].

Despite a wide range of applications and problems solved through existing control methods, the requirement of large amounts of computation power for large-scale optimisation problems, dependence and necessity of accurate models, and their limitation to generalise under uncertainties remain a challenge. To this end, learning based methods offer an alternative paradigm by learning through data while only specifying high-level objects instead of detailed mathematical specifications of the physical environment. Beside optimisation methods, such as genetic

algorithms [165], particle swarm optimisation [166], evolutionary optimisation [167], or Bayesian Optimisation [43], Reinforcement Learning (RL) [168] has been the main approach to solve problems which are difficult for classical control methods. While gradient-free optimisation methods have been mainly used to optimise for the parameters of a controller [46, 44] or a model [7], RL is learning a parametrised function representing both the control policy [60, 67] and the model [169, 170].

For complex tasks, such as continuous control and robotics tasks, Deep Neural Networks have been used as function approximators for the actor and critic, opening the field of Deep Reinforcement Learning (DRL). In robotics, DRL has been used for wide applications and problems that are hard to solve using classical control methods, such as control of life-like locomotion of humanoids [171, 67] and animals [65], fall recovery for quadrupeds [66], and hand-eye coordination for grasping [68].

Due to the data demanding nature of DRL algorithms, the policies are learned in simulation removing the risk of damaging the robot and leveraging advances in computation speed and parallel computing allowing fast collection of samples. To ensure that the simulation to reality (sim2real) transfer of the policy yields similar performances in reality as in simulation, the fidelity of the simulation can be increased [66, 145] or the policy can be trained to be robust to uncertainty [172, 65].

To complete multiple tasks using one unified framework, hierarchical control structures can be used to select appropriate sub-policies through a high-level gating mechanism. In HRL, complex tasks are solved by using existing knowledge of experts through temporal abstraction [73]. In its original form, temporal abstraction is achieved by having the top-level policy selecting among options, which are sub-policies that continuously perform one action over a period of time [73].

With the advent of Deep Learning and Deep Neural Network, DRL has been used to extend the discrete tabular HRL concepts into the continuous control domain. In the context of multi-expert learning and multi policy composition, this led to frameworks where low-level experts encode motion primitives, control fragments, or skills, while a high-level policy selects the expert [83, 84, 85]. Alternative to learning a high-level policy to select appropriate motion primitives, COCoMoPL [86] proposes a framework, where near-optimal motion primitives are learned and synthesised into a motion as weighted combination of these motion primitives.

In [87], a method is proposed that does not synthesise expert skills into a unified policy, but rather expands the existing skill by decomposing the task into simpler subtask and training a local policy for the subtasks.

A modular framework is proposed in [88], which learns transition policies that connect primitive skills to complete sequential tasks. An extended work in [89] uses the modular framework to learn to coordinate the learned primitive skills for task completion. In [90] a method is proposed that learns task-agnostic skills to use their composite to solve new tasks in an HRL fashion. Task completion is accomplished by the

task-agnostic skills performing commands provided by the high-level command.

In contrast to aforementioned methods that select one sub-policy, the outlined MES in this work continuously blends all experts. From a practical perspective, this allows the multi-expert policy to choose latent skills from multiple experts instead of just choosing a particular skill from one expert. Furthermore, MES continues to train the experts alongside the high-level behaviour network, such that the learned expert skills can be specialised and can learn new skills, while gating network blends the experts' skills.

5.3 Generating Expert Behaviours

For the multi-expert learning structure, first, individual narrowly-skilled experts need to be obtained. In this section, two procedures are presented to obtain expert behaviours that solve a specific task: (1) autonomous learning a policy through interactions with the environment and (2) learning to imitate a reference policy.

In order to solve a task, a reward, whose maximisation leads to task completion, needs to specified. If the policy is autonomously learning from scratch, methods from Deep Reinforcement Learning are deployed to maximise the reward. For imitation learning of a user-designed controller, a controller is designed first, and then a policy represented as Neural Network (NN) is trained to imitate the controller as closely as possible while maximising a task reward.

Using model-free DRL algorithms to learn a policy from scratch is helpful when the solution is not straightforward. On the other hand, using an existing manually designed controller gives more certainty of behaviours over the robustness, stability, and performance. Thus, combining control design for well-defined tasks, such as locomotion, and leveraging the exploration of learning methods for tasks in ill-defined or uncertain states and environments, such as fall recovery from any pose, provides experts for a variety of cases.

In the following, we will first introduce the control structure how the robot is controlled to achieve its task, and then present the procedure to learn experts through DRL. Lastly, we will show how an expert can be obtained from imitation learning.

Control Structure

The control structure of the robot consists of two control loops: high-level behaviour control, and low-level joint impedance control (Fig. 5.2). The high-level behaviour control loop governs the behaviour of the robot at 25Hz by mapping the robot's states into actions for the joint impedance controller, which runs at a frequency of 400Hz. From measured joint positions and velocities, the joint torques are generated in a proportional-derivative control law:

$$\tau_i = K_P(q^d - q) - K_D\dot{q}, \tag{5.1}$$

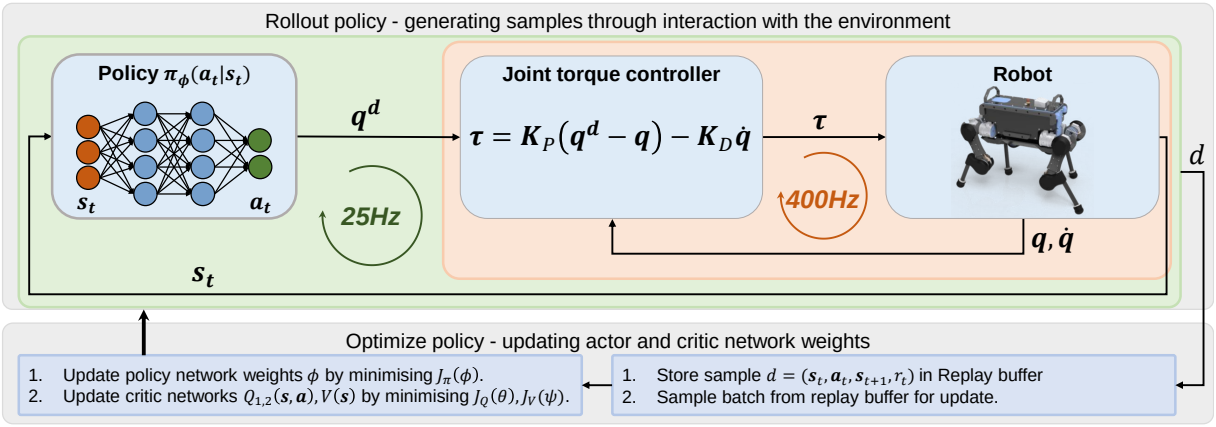


Figure 5.2: Learning framework to train specialised experts and multi-expert policies.

with joint index $i = 1, ..., N$, joint torque τ, proportional gain K_P, derivative gain K_D, target joint position q^d, measured joint position q, and joint velocity $\dot{q}$.

Here, a low-pass Butterworth filter is applied respectively on the feedback state and actions in the high-level control loop. Filtering the robot state allows reducing sensor noise while also providing smoother states for the Neural Network, thus yielding smoother target actions. Besides, to match the torque tracking bandwidth of the actuators, desired actions of the behaviours control are low-passed filtered and limited to those that are able to be executed by the actuators.

Training Experts via Deep Reinforcement Learning

For training an expert represented as Deep Neural Network, which is subsequently used in MES, Deep Reinforcement Learning (DRL) is used to solve the task (Fig. 5.2). The objective of DRL is to maximise the expected reward $\mathbb{E}[r]$ from the environment, where the reward r is designed such that its maximisation leads to completion of the task. Through explorative actions, the DRL agent improves the policy by encouraging actions that lead to high reward.

The Soft Actor-Critic (SAC) algorithm, an off-policy actor-critic DRL algorithm with stable convergence properties that intrinsically trades-off exploitation versus exploration [62], is used to train both the single-task experts and the multi-expert policy in MES (see Section 5.4). In the following, we outline the SAC algorithm and present the design decisions for training task-completing policies.

To enhance the smoothness for realisable control actions on robotic systems, an additional smoothness objective is added to the standard maximum entropy objective [173] of SAC:

$$J(\pi) = J_{\text{SAC}}(\pi) - \lambda J_{\text{smoothing}}(\mu), \tag{5.2}$$

with stochastic policy π, smoothness regularisation parameter λ, deterministic policy μ, maximum entropy objective $J_{\text{SAC}}(\pi)$, and smoothness loss $J_{\text{smoothing}}(\mu)$.

The maximum entropy objective $J_{\text{SAC}}(\pi)$ is defined as:

$$J_{\text{SAC}}(\pi) = \sum_{t=0}^{T} \mathbb{E}[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))], \tag{5.3}$$

with state s_t, action a_t, temperature parameter α, expectation $\mathbb{E}$ of the reward r and policy entropy $\mathcal{H}(\pi)$. The temperature parameter α governing the trade-off between exploration and exploitation is automatically adapted by minimising the objective $J(\alpha)$ [174]:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t}[-\alpha \log \pi_t(a_t|s_t) - \alpha \bar{\mathcal{H}}], \tag{5.4}$$

with minimum policy entropy threshold $\bar{\mathcal{H}}$.

The smoothing loss $J_{\text{smoothing}}(\mu)$ encourages the policy to generate deterministic actions $\mu(s_t)$ that are close to the current, measured state q:

$$J_{\text{smoothing}}(\mu) = \|\mu(s_t) - q\|_2. \tag{5.5}$$

Directly embedding a regularisation loss on the optimisation level yielded better smoothness, compared to the additional regularisation terms in the reward.

In the SAC algorithm, the parameters of three function approximators are learned: parameters ϕ of the policy π_ϕ, parameters θ of the soft action-value function Q_θ, and parameters ψ of the soft state-value function V_ψ.

After applying the reparametrization trick, the policy parameters ϕ can be learned by minimising the objective $J_\pi(\phi)$ [62]:

$$J_\pi(\phi) = \mathbb{E}[\log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)] + J_{\text{smoothing}}(\mu_{\phi^1}). \tag{5.6}$$

The policy π_ϕ is re-parametrised through a neural network transformation: the action $a_t = f(\mu_{\phi^1}, \sigma_{\phi^2})$ is sampled from a squashed Gaussian distribution $f(\mu_{\phi^1}, \sigma_{\phi^2}) = \tanh(\mathcal{N}(\mu_{\phi^1}, \sigma_{\phi^2}))$. The deterministic policy μ_{ϕ^1} and the standard deviation σ_{ϕ^2} are Neural Networks parametrised by the weights ϕ^1 and ϕ^2 respectively.

For stability of the training, the more conservative estimation between two Q networks Q_{θ_1} and Q_{θ_2} is used for the action value function $Q_\theta(s_t, a_t)$:

$$Q_\theta(s_t, a_t) = \min_{j=1,2} Q_{\theta_j}(s_t, a_t). \tag{5.7}$$

Minimising Bellman residual $J_Q(\theta_j)$ with bellman equation $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}[V_{\bar{\psi}}(s_{t+1})]$ and discount factor γ yields the parameters θ_j for action-value function $Q_{\theta_j}(s_t, a_t)$:

$$J_Q(\theta_j) = \mathbb{E}[\tfrac{1}{2}(Q_{\theta_j}(s_t, a_t) - \hat{Q}(s_t, a_t))^2]. \tag{5.8}$$

The parameters $\bar{\psi}$ of the target value function network $V_{\bar{\psi}}(s_{t+1})$ are obtained by polyak averaging the parameters ψ through minimising the objective $J_V(\psi)$:

$$J_V(\psi) = \mathbb{E}[\tfrac{1}{2}(V_\psi(s_t) - \mathbb{E}[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t|s_t)])^2]. \tag{5.9}$$

State Observation

Figure 5.3: Potential state variables as feedback states for locomotion and manipulation tasks.

Correctly designing state space $\mathcal{S}$ with only essential state variables provides clear reward signals and better success rate of learning good policies. On the contrary, using irrelevant states introduces large variances in the reward or feedback signals, and consequently decrease performance. Furthermore, from a computational perspective, high-dimensional state spaces require more data to generalise the whole state space, and impede convergence because the non-linear function approximator is not able to inter- or extrapolate the high-dimensional state space due to the curse of dimensionality.

In Section 5.4, we proposed a systematic approach to choose the correct subset of states among all possible variables. Fig. 5.3 depicts a non-exhaustive set of potential states for floating-base and fixed-base systems, such as quadrupeds and manipulators respectively.

For all physically measured quantities, a first-order low-pass Butterworth filter is applied to denoise. The cut-off frequency is individually set and is determined through a spectral analysis based on signals from the idle system.

Action Space

For the choice of the action space $\mathcal{A}$, two established options are considered: *joint space* for controlling the joints and *task space* for controlling the end-effector pose. In this work, we use the joint positions for quadrupeds and end-effector poses for manipulators.

Reward Design

As the reward function governs the behaviour of the policy, ambiguously specifying the reward could lead to reward exploitation and potentially failures. Here, we formulate a set of physically-motivated reward terms r_i for both floating-based and fixed-based systems.

The reward r_t at time t is calculated as weighted sum over the individual reward terms r_i with weights w_i:

$$r_t = \sum_{i=0}^{N} w_i r_i. \tag{5.10}$$

To both regularise and achieve desired values, a Gaussian Radial Basis Function (RBF) $r_i : \mathbb{R} \to (0, 1]$ is used:

$$r_i(x) = \exp(-\kappa \|x_d - x\|^2), \tag{5.11}$$

with desired value x_d and current value x. For regularisation terms, the desired value x_d is set around the operating point. The width κ governs the tolerance δ_{max} of the residual error, and is calculated as $\kappa = -\ln(C)/\delta_{max}^2$ with small associated reward $C \doteq 0.001$ at the boundaries of the tolerance. We use an approximation $C \to 0$ since $\ln(C = 0)$ is infinite.

All single-task experts and multi-expert policies in this work are trained by maximising rewards to follow desired link poses, and regulate joint velocity and joint torque for smoothness in actions. The rewards used in this work can be found in Section 5.5, and a more detailed description for reward design can be found in [67, 171, 66, 175].

Training Procedures

Designing training procedures for the DRL agent is required for both the success and high quality policies. We applied four techniques during the training of the policy: early termination, reference state initialisation (RSI), guided curriculum, and dynamics randomisation.

Early termination stops an episode when meeting an early termination criterion and is therefore used to discard irrelevant and skewed samples. Early termination biases the policy to avoid bad states as the agent cannot collect any further rewards if the episode is terminated. Early termination criteria are set for states that have self-collision or unwanted collisions with the environment. For floating-base systems, once the height of the base falls below a threshold, the episode is terminated.

In order to generalise well across the whole state space and thus training a robust policy, the robot is initialised in reference states that are task-relevant but rarely encountered [144]. Reference states include failure states, local minimum solutions, or imitation frames (see Section 5.3).

For best performance of the policy, curriculum for the learning process is applied. The curriculum aims to guide the policy by gradually increasing the difficulty of the task. In general, if the task difficulty is too high, the policy converges to a locally optimal solution. An increase in the difficulty can be implemented on reward weight, e.g., increasing importance on regularisation weights [66], or in the amount of tasks which the policy needs to complete, e.g., initially standing for a quadruped robot and gradually adding locomotion tasks while withstanding disturbances or an increase in dynamics randomisation [171]. In our setting, the robot is being pushed during training as in [171] to provide additional robustness towards uncertainties.

To increase the robustness of the policy against model uncertainties and to improve the transferability of policies across environments, we applied dynamics randomisation [172] on the robot model. At the beginning of every episode, the parameters of the robot dynamics are uniformly randomly sampled within the range specified in Table 5.1.

Table 5.1: Variations of dynamics randomisation for training.

	Default value	Min	Max
Contact friction	0.7	50%	150%
Joint torque	40Nm	80%	120%
Inertia	link dependent	80%	120%
Mass	link dependent	80%	120%

Training Expert via Imitation Learning

As an alternative to learning via DRL from scratch, we use imitation learning to train a DRL agent by imitating a reference trajectory. In the following, we present a learning scheme for quadruped locomotion that encourages the agent to imitate a reference motion while solving a task as described in Section 5.3.

Four quantities are used for the learning algorithm to imitate: joint positions, joint velocities, relative end-effector positions, and contact states of the feet. The references are time-based trajectories obtained from an optimal trajectory generator for the Centre of Mass (CoM) and the whole-body controller that tracks the CoM trajectory [176, 4].

A policy that imitates a reference demonstration is trained in the learning framework described in Section 5.3. For imitation learning, the reward is modified. The new reward consists of a task reward r_{task}, and an imitation reward $r_{\text{imitation}}$:

$$r_t = w_{\text{task}} r_{\text{task}} + w_{\text{imitation}} r_{\text{imitation}}, \tag{5.12}$$

with weights $w_{\text{task}}, w_{\text{imitation}}$ corresponding to the importance of task completion and imitation quality respectively.

The imitation reward $r_{\text{imitation}}$ is calculated as the weighted sum of four sub-rewards:

$$r_{\text{imitation}} = w_q r_q + w_{\dot{q}} r_{\dot{q}} + w_{\text{eef}} r_{\text{eef}} + w_{\text{contact}} r_{\text{contact}}. \tag{5.13}$$

The joint position r_q, joint velocity $r_{\dot{q}}$, and end-effector position r_{eef} rewards are formulated as Gaussian RBF encouraging similarity between demonstrated and actual robot state. For the contact state, a reward of 1 is assigned if all four contact states match the demonstrated contact state, and is 0 otherwise.

To prevent temporal ambiguity during the imitation of a time series of reference frames, a variable representing time needs to be provided, similar to [67, 144]. Because of the periodic nature of locomotion, a periodic phase vector ζ is formulated as:

$$\zeta = \begin{bmatrix} \sin(\Omega t) \\ \cos(\Omega t) \end{bmatrix}, \tag{5.14}$$

where Ω normalises the period to match the periodicity of the reference time series, and the representation using ζ is compatible with NN instead of using a monotonically increasing time as a variable.

For RSI, the robot's joint position and joint velocities are initialised according to a uniformly random sample from the reference trajectory.

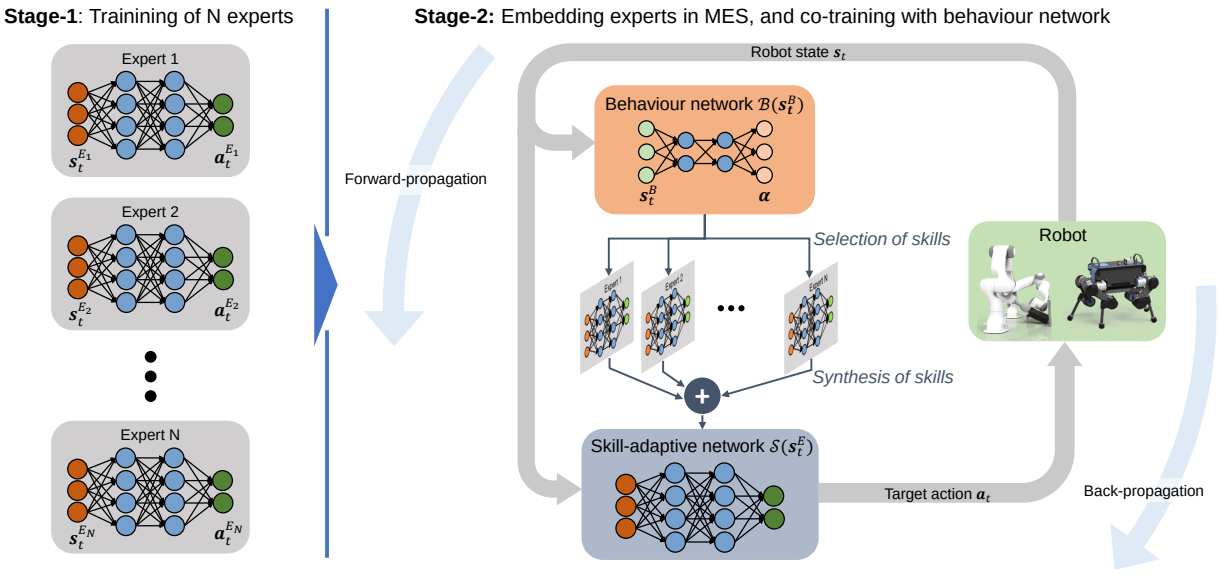


Figure 5.4: Multi-Expert Synthesis trained in a two-stage curriculum. Experts are pre-trained in Stage-1 and co-trained in Stage-2 alongside a Behaviour Network.

5.4 Multi-Expert Synthesis

In the following, we outline Multi-Expert Synthesis (MES) and elaborate how the trained experts are embedded in MES. We further formulate an automatic State Space Selection Process that was used to determine the state space of all policy networks. Lastly, we present how to enforce diversity among experts and tackle the problem of mode-collapse.

Learning Structure

The two-stage training process for MES policies is depicted in Fig. 5.4. In the first stage, individual, single-skilled experts are trained by solving a particular task as described in Section 5.3. In the second stage, the high-level behaviour network is trained alongside the pre-trained experts. While being synthesised by the high-level behaviour network, the pre-trained experts in the second stage are further fine trained which allow all experts to synergise with one another.

Based on the current behaviour state s_t^B , the behaviour network $B(s_t^B), B : \mathbb{R}^{\dim(s_t^B)} \rightarrow \mathbb{R}^N$ continuously synthesises the N task-specific expert skills and builds a skill-adaptive network $S(s_t^E)$. The skill-adaptive network (blue shaded network in Fig. 5.4) is used to infer actions $a_t = S(s_t^E)$ from expert state s_t^E.

For this work, we consider two approaches – MELA and MoE – to achieve MES. The difference between MELA and MoE lies in how the outputs $\alpha = B(s_t^B)$, $\sum_{i=1}^{N} \alpha_i = 1$ from behaviour network B are used as weights to synthesise the expert networks into the skill-adaptive network.

For MELA, the outputs of the behaviour network α blend the network parameters of the experts [2], while for MoE α are used for the weighted sum of the actions a_i of all expert networks.

Multi-Expert learning Architecture (MELA)

The skill-adaptive network's parameters $\theta^{[l]}_{n,\text{SAN}}$ are obtained as the weighted sum of the expert's parameters $\theta^{[l]}_{n,i}$:

$$\theta^{[l]}_{n,\text{SAN}} = \sum_{i=1}^{N} \alpha_i \theta^{[l]}_{n,i}, \tag{5.15}$$

for the i-th expert, neuron n in layer l, and blending weights $\alpha = B(s^B_t)$ with $\sum_{i=1}^{N} \alpha_i = 1$ from the behaviour network B. All operations in the architecture are differentiable that allow backpropagation, and the networks are trained with SAC.

As the skill-adaptive network's parameters are a linear combination of all expert parameters, which have the same amount of neurons and layers. Thus, to accommodate different expert network sizes, the expert network size in Stage-2 is augmented (Fig. 5.5). The amount of layers and neuron per layers are the same as the largest network. To keep the input-output behaviour of the augmented neural network unchanged, all weights $w^{[l]}_{k,j}$ of the newly added neuron connections are initialised to zero. The output of the j-th neuron $h^{[l]}_j$ thus remains unchanged for $w^{[l]}_{k,j} = 0$:

$$h^{[l]}_j = f^{[l]}\left(\sum_{i=0}^{n} w^{[l]}_{i,j} x^{[l-1]}_i + \sum_{k=n+1}^{n_k} w^{[l]}_{k,j} x^{[l-1]}_i + b^{[l]}_j\right), \tag{5.16}$$

with n neurons in layer l, activation function $f(\cdot)$, weight $w^{[l]}_{i,j}$, output $x^{[l-1]}_i$ of the previous layer, and bias b. Examples for network augmentation of locomotion and manipulation policies can be found in Section 5.5 (Fig. 5.5).

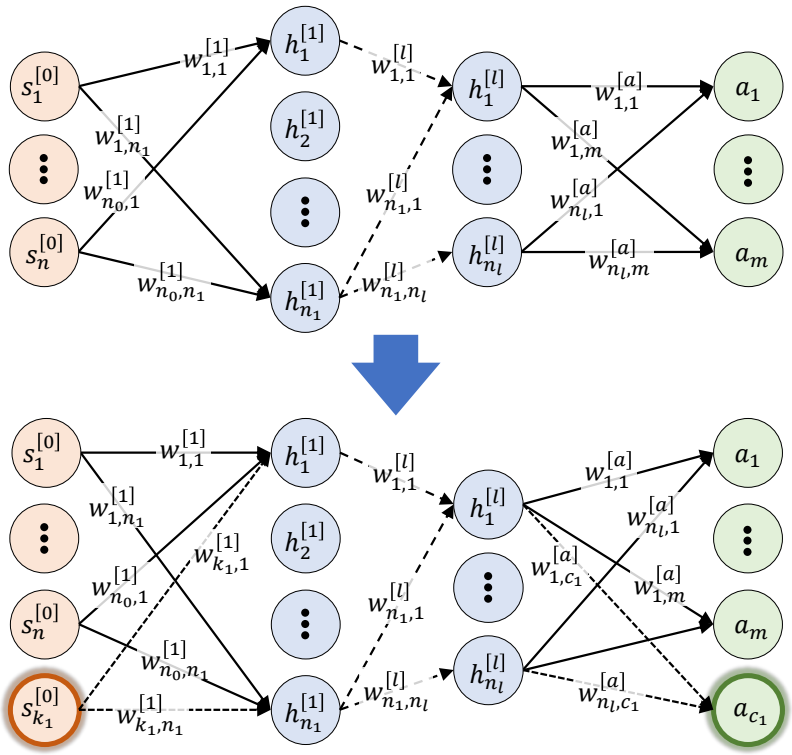


Figure 5.5: Augmented expert network. **Top**: Originally trained expert. **Bottom**: Augmented expert network. All neurons are fully connected including bias terms but omitted for clarity.

Mixture of Experts (MoE)

For MoE, the actions a are calculated as weighted sum from the expert's actions a_i and the behaviour network's output $\alpha = B(s_t^B)$:

$$a = \sum_{i=1}^{N} \alpha_i a_i, \tag{5.17}$$

where N is the number of experts with corresponding weights α_i on the output actions a_i. Consequently, compared to MELA, the dimensionality of the latent space of MoE is much lower by orders of magnitudes.

Automatic State Space Selection

In the following, we present the systematic selection of the state space based on their relevance to the task. The core idea of the state selection process is to check whether the removal of a sub-state space $\mathcal{S}^-$ from state space $\mathcal{S}^*$ has an influence on the value function $V(s), \forall s \in \mathcal{S}$.

The value function $V : \mathbb{R}^n \to \mathbb{R}$ describes the accumulative reward gained in an episode by starting in state s and applying policy $\pi(a|s)$ successively. A properly designed reward will allow the maximisation of accumulated reward to achieve the task completion. If the reduced state space $\mathcal{S}^+ = \mathcal{S}^* \setminus \mathcal{S}^-$ yields a value $V(s^+)$ similar to the complete $V(s^*)$, then $\mathcal{S}^-$ has no influence on the accumulated reward, and can be thus omitted.

More specifically, we have:

$$\begin{gathered} V(s^+) \simeq V(s^*), \quad \forall s^+ \in \mathcal{S}^+, s^* \in \mathcal{S}^*, \\ \Leftrightarrow \\ \mathcal{S}^- \text{is not required for the task,} \end{gathered} \tag{5.18}$$

where $\mathcal{S}^+ \cup \mathcal{S}^- = \mathcal{S}^*, \mathcal{S}^+ \cap \mathcal{S}^- = \emptyset$.

To describe the various state quantities, we use the following notation. Symbols written in bold denote a state space $\mathcal{S} \in \mathbb{R}^n$ with n dimensions. The k-th dimension of the state space $\mathcal{S} = \bigcup_{k=1}^{n} \mathcal{S}_k$ is denoted as $\mathcal{S}_k \in \mathbb{R} \subset \mathcal{S}$, and its members are called state variables $s_k \in \mathcal{S}_k$. In the context of state space reduction, we use state space $\mathcal{S}^* = \bigcup_{k=1}^{n} \mathcal{S}_k$ as the union of relevant $\mathcal{S}^+$ and irrelevant $\mathcal{S}^-$ sub-state spaces. A vector $s \in \mathcal{S}$ is called a state.

In Algorithm 2, we outline how the state space is selected. Algorithm 3 describes the practical implementation of (5.18)) to determine relevant state variables. For selecting relevant sub-state spaces (Algorithm 2), the state space is repeatedly reduced using Algorithm 3. During every iteration, value function $V^\pi(s)$ and policy $\pi(a|s)$ are trained (Section 5.3) using states $s \in \mathcal{S}^i$. If the state space is not further reduced, the task-relevant state space $\mathcal{S}^+$ is found. Otherwise, a new policy and value function is trained using the newly reduced state space $\mathcal{S}^{i+1}$.

In our work, the task-relevant state space is usually found after one iteration. However, when no domain knowledge for the initial state space $\mathcal{S}_0$ can be applied, then all possible state dimensions described

in Section 5.3 are used and more iterations are needed. This is because the learned value function $V(s)$ is not capable of approximating a high-dimensional state's value with the given amount of data. In this case, the state space needs to be successively reduced (line 5 in Algorithm 2) for a better approximation of the state's values.

Algorithm 2: Pseudo code for relevant state selection

```
1 Function selectStateSpace()
2     S^i ← S^0
3     while S^+ not found do
4         Train π(a|s), V(s) with s ∈ S^i.
5         S^{i+1} ← reduceStateSpace(V(s), S^i)
6         if S^i = S^{i+1} then
7             Found task-relevant state space S^+
8         else
9             S^i ← S^{i+1}
```

For the reduction of the state space in Algorithm 3, two thresholds $\delta_{\text{relevant}}, \delta_{\text{required}}$ are set. The *relevance* of sub-state space $\mathcal{S}_k$ is evaluated N times. If the average relevance exceeds threshold δ_{required}, then sub-state space $\mathcal{S}_k$ is *required*. Averaging the relevance is necessary due to the variance in the estimation of the value function $V(s)$.

At every iteration of a state's relevance (line 4 in Algorithm 3), the relevance of every sub-state space $\mathcal{S}_k$ is evaluated by perturbing the corresponding state variable s_k of a measured state s. The state s is measured while rolling out policy $\pi(a|s)$. The state variable s_k is perturbed m times by setting their values to a uniformly sampled value within $[\Delta_{\text{min}}, \Delta_{\text{max}}]$ covering the sub-state space $\mathcal{S}_k$ for $m \to \infty$.

If the percental change of the average perturbed value is small for state variable s_k, the sub-state space $\mathcal{S}_k$ is considered irrelevant to the cumulative reward. If the relevance averaged over N times is below threshold δ_{required}, implying that the sub-state $\mathcal{S}_k$ is irrelevant across the whole state space $\mathcal{S}^*$ for $N \to \infty$, we determine sub-state $\mathcal{S}_k$ as not required.

In MES, state selection (Algorithm 2) is conducted for every expert during Stage-1 and for the behaviour network in Stage-2. Network augmentation (equation 5.16) is used in Stage-2 to unify the network sizes across the experts.

Expert Diversification

During training of multi-expert behaviours, a common problem in multi-expert systems known as expert imbalance occurs. This is because when one expert is over-prioritised by the behaviour network, it results in downgrading of other experts and the inability to learn distinct expert behaviours. For tasks requiring multiple experts as shown in Section 5.5, expert imbalance leads to the task not being completed and a local minimum solution.

Algorithm 3: Pseudo code for selecting relevant state variables

Function reduceStateSpace(V(s), $\mathcal{S}^*$)
  Set threshold δ_{relevant}, δ_{required}
  $\mathcal{S}^-, \mathcal{S}^+ \leftarrow \emptyset$
  for iteration $= 1, ..., N$ **do**
    Measure $s^* \in \mathcal{S}^* \in \mathbb{R}^n$
    requiredStates $\leftarrow$ dict("s_i": []), $i = 1, 2, ..., n$
    for $\mathcal{S}_k \subset \mathcal{S}^*, k = 1, 2, ..., n$ **do**
      $s^*_{\text{perturbed}} \leftarrow s^*$
      perturbedValues $\leftarrow$ list(m)
      for $i = 1, 2, ..., m$ **do**
        $s^*_{\text{perturbed}}[k] \sim U(\Delta_{\min}, \Delta_{\max})$
        perturbedValues[i] $\leftarrow V(s^*_{\text{perturbed}})$
      $\mu = \text{mean}(\text{perturbedValues}) - V(s^*)$
      if $\mu / V(s^*) < \delta_{\text{relevant}}$ **then**
        requiredStates[k].append(0)
      else
        requiredStates[k].append(1)
  for $\mathcal{S}_k \subset \mathcal{S}^*, k = 1, 2, ..., n$ **do**
    if mean(requiredStates[k]) $< \delta_{\text{required}}$ **then**
      $\mathcal{S}^- \cup \mathcal{S}_k$
    else
      $\mathcal{S}^+ \cup \mathcal{S}_k$
  return $\mathcal{S}^+$

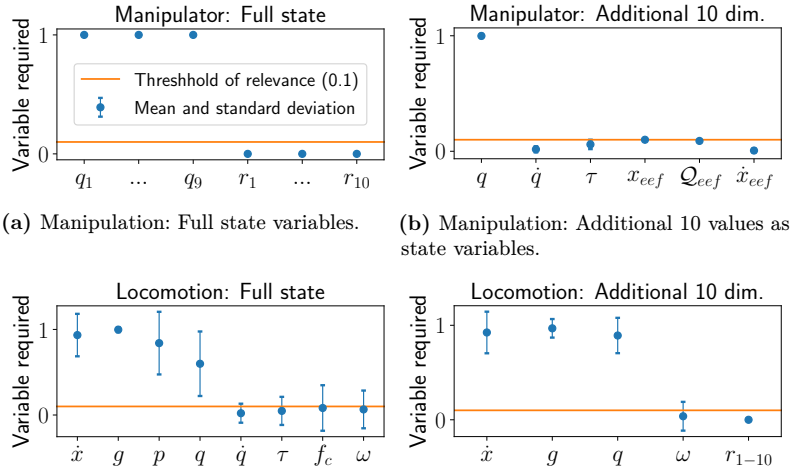
To prevent expert imbalance, we encourage the experts to learn easy-to-discriminate and diverse skills by including a discriminator objective [177] in the DRL objective (5.2):

$$J_{\text{diversity}}(q_\phi) = \log q_\phi(z|s^D), \tag{5.19}$$

with learned discriminator $q_\phi(z|s^D)$ parametrised by weights ϕ and one-hot vector z indicating the skill. The discriminator estimates the likelihood of skill z conditioned on state s^D, and is trained to minimise the cross entropy $H(\hat{z}, z)$ between real skill $\hat{z}$ and predicted skill z:

$$J_{\text{discriminator}} = \frac{1}{N} \sum_{i=1}^{N} H(\hat{z}_i, z_i). \tag{5.20}$$

The diversity objective (5.19) encourages the policy to produce states as distinct as possible, so that the discriminator can easily estimate the skill z based on the state s^D. Note that the state $s^D \in \mathcal{S}^D$ can different from the expert state space $\mathcal{S}^E$. The discriminator state space $\mathcal{S}^D \in \mathbb{R}^6$ used for the locomotion discriminator uses orientation and velocity to discriminate between fall and locomotion. Beside explicitly enforcing skill diversification, it shall be noted that implicit diversity naturally emerges in MELA and RSI [2].



(a) Manipulation: Full state variables.

(b) Manipulation: Additional 10 values as state variables.

(c) Locomotion: Full state variables.

(d) Locomotion: Additional 10 values as state variables.

Figure 5.6: Results of state selection using large state space and 10 additional dimensions for locomotion and manipulation.

5.5 Results

In this section, first we outline the setup of the learning framework. Second, we present results of our proposed state selection process. Third, the expert behaviours of MES are shown. Next, Multi-Expert results on the quadruped and dual-arm robot are reported. Lastly, we show the robustness and ability of MES to generalise across environments.

In particular, we demonstrate that our proposed state selection process finds the optimal state space yielding the highest reward and is used to design the state space for all components of MES. Furthermore, MES learns robust multi-expert policies that are effective from simulation to the real robots and in unseen test scenarios.

Training Setup

Table 5.2: Variations of dynamics randomisation during testing.

	Default value	Min	Max
Contact friction	0.7	30%	170%
Joint torque	40Nm	80%	200%
Inertia	link dependent	50%	150%
Mass	link dependent	50%	150%

All policies were trained in PyBullet [178] on a commercially available computer (CPU: i7-7700K, GPU: Nvidia GTX 1080Ti). The expert and multi-expert policies converged after 1000 and 2000 (Fig. 5.12) epochs respectively. Every epoch consisted of 1000 samples using the sample collection depicted in Fig. 5.2 with 25 samples collected per second.

All networks are two-layered, fully connected Neural Networks with 256 neurons in each layer. Rectified Linear Units (ReLU) were used as activation functions. The standard parameters of SAC were used as in [62].

We performed dynamics randomisation [172, 65] during the training (see Section 5.3) and transferred the policy from PyBullet to both

Gazebo [179] and the real robots. During training in PyBullet a range of values were used for dynamics randomisation as shown in Table 5.1, and a larger range (see Table 5.2) were used for testing in Gazebo, which show that the multi-expert policy is robust in presence of such large physical discrepancies.

Comparison of Different State Observations

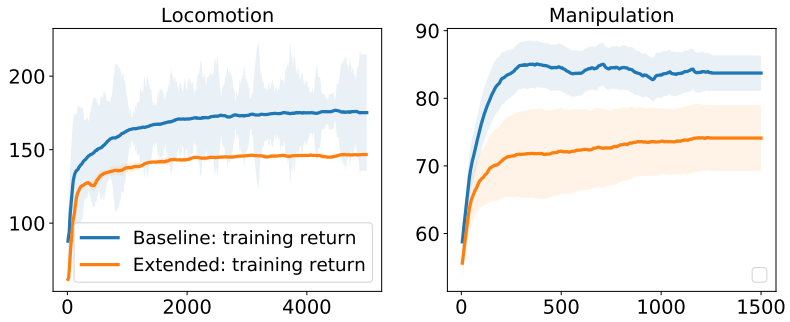


Figure 5.7: Performance for different state space configurations for locomotion (left) and manipulation (right).

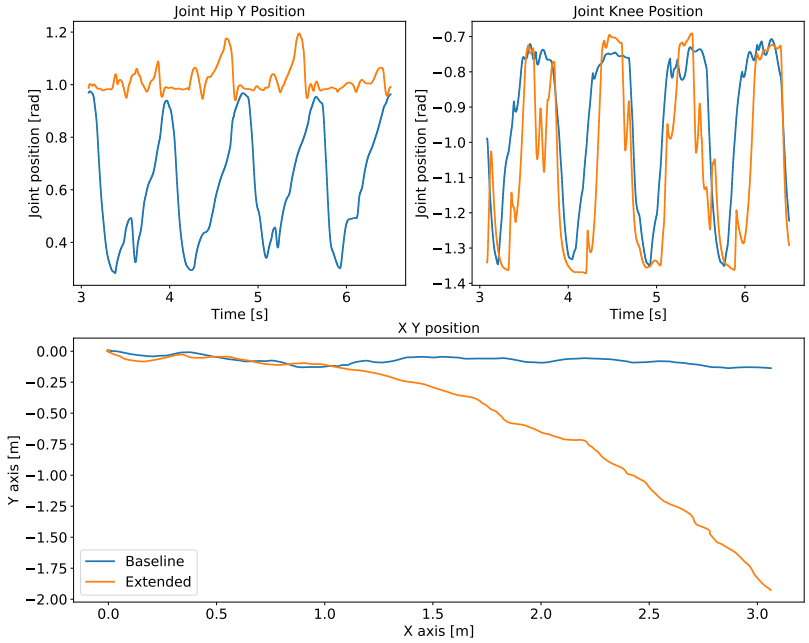
The state selection process described in Section 5.4 was used to design the state space for the experts and gating network for all policies. We demonstrate the process of state selection and its effectiveness by two domains of tasks: quadruped locomotion and reach-grasp manipulation (see Section 5.5).

First, we determine the relevant state variables from a large set of potential state variables. Second, we show that our proposed method can distinguish between completely irrelevant state variables and potentially relevant state variables by adding 10 random variables to the state space. Lastly, we compare the performance of optimally selected state spaces and extended state space containing irrelevant state information, based on policy quantitative performance and qualitative behaviour.

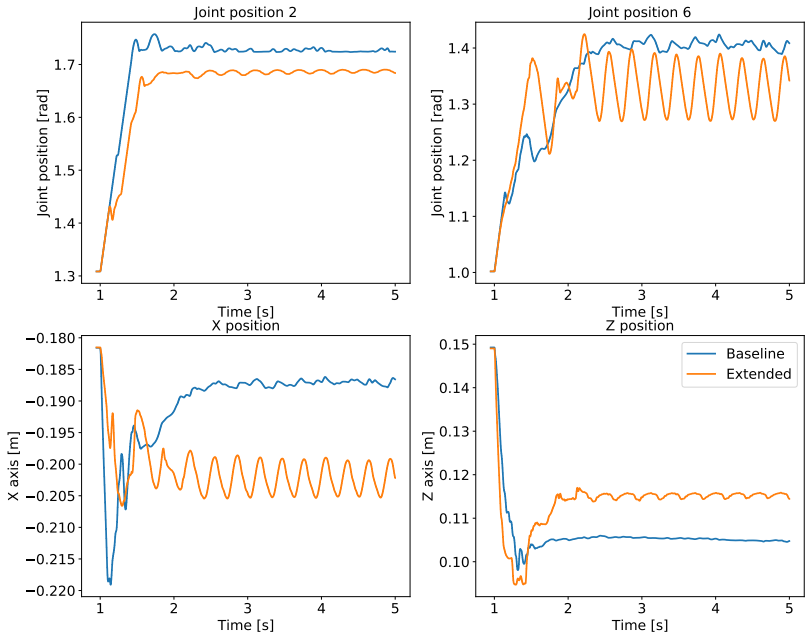
We chose and defined the state variables for initial state space $\mathcal{S}_0$ as: linear base velocity $\dot{x}$, angular base velocity ω, gravity vector g, phase p, joint position q, joint velocity $\dot{q}$, joint torque τ, end-effector force f_c for locomotion, and additionally end-effector position x_{eef} and velocity $\dot{x}_{eef}$, and quaternion $\mathcal{Q}_{eef}$ for manipulation.

After one iteration of Algorithm 2 with threshold $\delta_{\text{required}} = 0.1$, the least relevant state variables were found to be: joint velocity, joint torque, and end-effector force for locomotion (Fig. 5.6 top left) and all state variables but the joint positions for the manipulator (Fig. 5.6 bottom left). By removing the least relevant state variables from the initial, extended state space $\mathcal{S}_0$, we obtain a reduced state space with only task-relevant state variables.

The state selection process was further validated by adding 10 random variables $r_i, i = 1, ..., 10$ to the reduced state space. For every sample, the value of every random variable $s_{\text{extra},i}$ was uniformly sampled $s_{\text{extra},i} \sim U(-1, 1)$. All 10 random variables (Fig. 5.6 top and bottom right) were identified to be irrelevant after one iteration of Algorithm 2.



(a) Locomotion performance: failure to capture periodicity (top) and drift of robot position (bottom) from the extended state space policy (orange), compared to our formulated baseline policy with optimal state space (blue).



(b) Manipulation performance: oscillatory behaviour using the extended state space (orange), compared to our formulated baseline policy (blue).

Figure 5.8: Comparison of policies trained using the optimal baseline state space versus the extended states.

Here, we define our formulation of reduced state space as the **baseline**, and those with all possible states as the **extended** state space including those irrelevant to the task, as found by our algorithms, for both locomotion and manipulation. We comparatively analysed the difference between the *baseline* and the *extended* state space representations in terms of the learning curves and the learned behaviours.

Our baseline state space converged to higher rewards in both locomotion and manipulation tasks (Fig. 5.7). In contrast, the extended state space

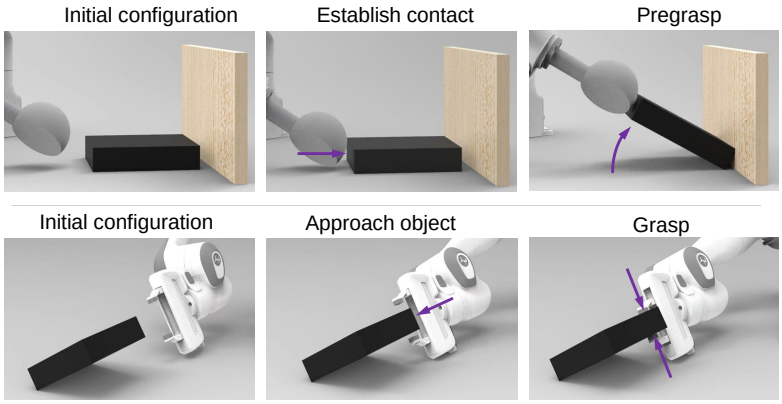
was easily stuck in local minima in almost 60% of the cases, leading to policies with lower return and the incompletion of the task. For a fair comparison, the local minima solutions that did not complete the task were not included in the learning curves.

The difference in performance between baseline and extended state space can also be observed from the policy's behaviours. While the extended state space representation can complete the tasks, our proposed baseline performs better than the extended case. During quadruped locomotion the extended state space representation drifts in the y direction and is not able to encode the periodicity of locomotion (Fig. 5.8a). Besides, the manipulation policy exhibits oscillatory behaviour after grasping the object as shown in Fig. 5.8b.

Expert Behaviours

Using the training procedure in Section 5.3, all experts were trained for the use in Stage-2 of MES (Fig. 5.4). In Fig. 5.9a, two expert behaviours are shown: gait recovery and trotting. Please see the accompanying video for further details.

(a) Expert policies for gait recovery (top) and trotting (bottom).

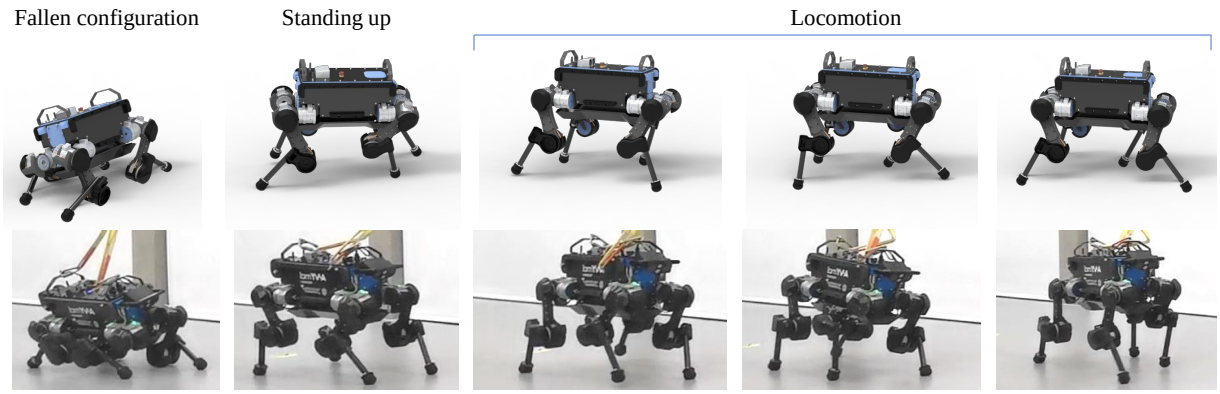


(b) Expert policies for pregrasping (top) and grasping (bottom).

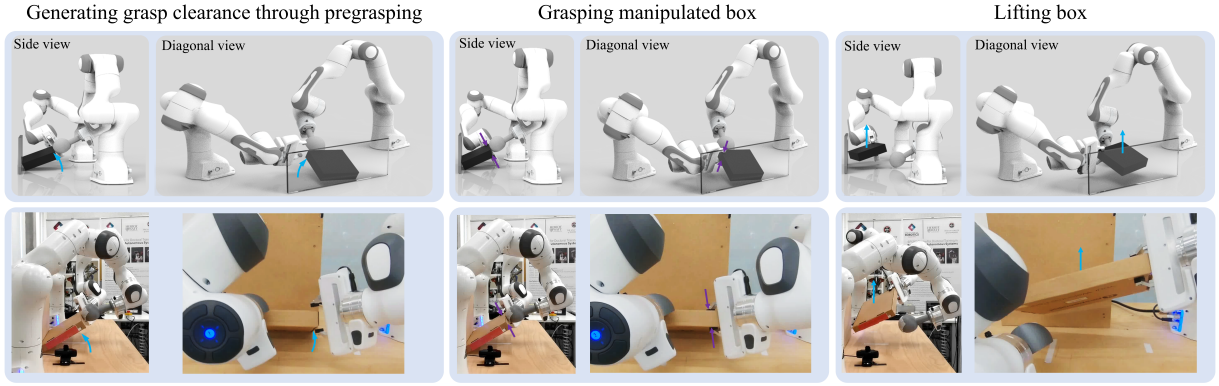
Figure 5.9: Expert policies for locomotion (a) and manipulation (b).

Locomotion Experts

The gait recovery and trotting experts were trained as follows. The training of the gait recovery expert was conducted through directly specifying a reward, with which the maximisation leads to task completion. For trotting additional reference trajectories were provided for



(a) Robust locomotion of gait recovery from a prone position and smooth transitions to trotting.



(b) Dual-arm cooperative manipulation and grasping.

Figure 5.10: Multi-expert policies in simulation and on the real robot.

imitation (see Section 5.3). The weights of the reward terms can be found in Table K.1.

All reward terms were expressed as radial basis function (see Section 5.3) unless stated otherwise. For gait recovery and push recovery, the sagittal velocity x_{vel} was set to zero; for the locomotion gaits, x_{vel} was provided by the imitation data, and the lateral and vertical velocities $y_{\text{vel}}, z_{\text{vel}}$ were zero. The unit gravity vector g^L is the normalised gravity vector in the robot body frame. Regularisation was performed on the joint velocities and joint torques by using a zero vector as target value. For the imitation terms, the target values were provided by the imitation data from a trotting controller.

Early termination was conducted for all experts if the robot was in self-collision. For locomotion, the episode terminated early if any link but the feet was in contact with the ground or if the body height fell below 0.25m. RSI was performed for both locomotion and gait recovery. For locomotion, the robot was spawned in joint states from the reference imitation data. For gait recovery, the robot was spawned in prone and supine body poses with random joint positions.

Manipulation Experts

For dual-arm cooperation, pregrasping and grasping experts were trained (Fig. 5.9b). The state space of both manipulators were determined using Algorithm 2. For grasping, the state space $\mathcal{S}_{\text{grasp}} \in \mathbb{R}^9$ consists of 9 joint positions of the manipulator. The action space consists of the end-effector's pose and the parallel grasper's joint position $\mathcal{A}_{\text{grasp}} \in \mathbb{R}^8$. The pregrasp expert uses end-effector pose as action space $\mathcal{A}_{\text{grasp}} \in \mathbb{R}^6$, and 7 joint positions and the object pitch angle for state space $\mathcal{S}_{\text{pregrasp}} \in \mathbb{R}^8$. Both experts use a two-layered neural network with 256 neurons in each layer.

For pregrasping, the reward has a contact and object orientation term with weights $w_{\text{contact}} = 1$ and $w_{\text{object}} = 3$ respectively. A reward of 1 was assigned when the end-effector was in contact with the object. The residual error in (5.11) was calculated as $\max(\theta_d - \theta, 0)$ with desired orientation $\theta_d = 45°$.

The grasping reward was calculated by the sum of finger contact and end-effector position reward, where weights were $w_{\text{contact}} = 1$ and $w_{\text{eef}} = 1$ respectively. A reward of 1 was assigned if both fingers were in contact with the object. A reward for the end-effector link being close to the desired position p_d was assigned as in (5.11). Early termination was performed in case of self-collision or if any link other than the the end-effector was in contact with the object.

Multi-Expert Results

The previously learned expert skills are now used for MES to achieve robust locomotion and dual-arm manipulation. We choose MELA over MOE as the multi-expert framework because MELA yields better policy performance during domain transfer and is able to diversify experts better (see Section 5.6).

Robust Locomotion

Two experts per skill were initialised for further diversification. Using the network augmentation shown in Fig. 5.5, the expert's state space can be preserved while being embedded in Stage-2 of MES. Early termination was applied in case of self-collision. The pose for RSI was uniformly sampled between prone positions and reference imitation trajectories.

For robust locomotion, the gait recovery expert reward r_{gr} is used if the robot falls, i.e., the threshold in height $p_z < 0.4m$ or body orientation $rpy > 20°$ is exceeded; and the locomotion expert's reward r_{loco} is used, otherwise. For the behaviour network's state space, the state selection process results in sufficient forward velocity. The motion can be seen in Fig. 5.10 and in the accompanying video.

Dual-Arm Cooperation

The dual-arm manipulation shows how MES adapts experts' access to information using the network augmentation technique depicted in Fig. 5.5. For the pregrasping expert, although the object's orientation is relevant during the Stage-1 training, it became less relevant after co-training with a behaviour network in Stage-2. The relevance of the object's orientation was determined by looking at the values of the weights and gradient related to the object orientation, which was almost zero. From the hierarchical structure, this is explainable since the behaviour network can access the object's orientation and thus activate the expert accordingly.

We found that MES consistently adapted the experts' requirement of certain state variables throughout all trained policies. The expert policies for the pregrasping and grasping task only used joint positions of the pregrasper and grasper respectively, while the behaviour network used the object's orientation as state input. Backpropagation through the MES network allowed the experts and behaviour network to share the state information for completing the task. The coordinated motions can be seen in the bottom of Fig. 5.10 and the accompanying video.

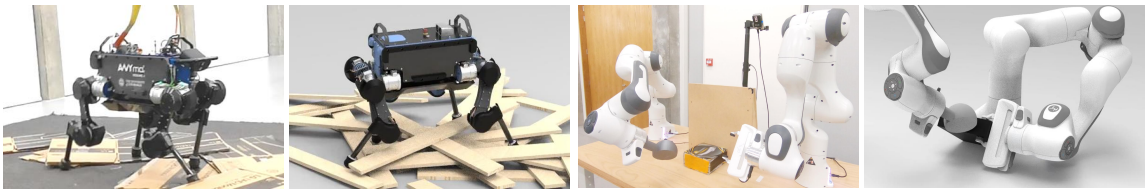
Robustness and Versatility of MES

For validating how robust and versatile the MES approach is, we deployed the multi-expert policies in test scenarios that have never been seen during training (see Fig. 5.11, accompanying video). Specifically, we tested the policy under environmental and hardware uncertainties. The policies' ability to robustly function in unseen environments and the successful transfer from simulation to the real world demonstrate that the learned MES policy is robust and can generalise across environments and domains with varying physics and dynamics properties.

Environmental Uncertainty

The MES policies were tested under varying dynamics parameters from 50% to 150% of the mass and inertia of the robot links. For locomotion, the robot traversed test terrains consisting of a cluster of planks, slippery objects (Fig. 5.11a), and withstood large pushes on the real robot (see accompanying video).

For dual-arm manipulation, the environment was modified by using different objects and support bases. In particular, we replaced the flat wall with the grasper's base as support (Fig. 5.11b right) and a round wall (see video). We replaced the nominal box with a torus (see video) as the object. Despite altering the environment in which the robot interacts with different objects and support walls, the policy can still adapt and complete the task. Furthermore, the successful dual-arm cooperation was shown in real experiments (Fig. 5.11b left), and under disturbances applied on the object (see accompanying video).



(a) Locomotion policy on real and simulated quadruped. **(b)** Manipulation policy on real and simulated robot arms.

Figure 5.11: Robustness under uncertainties and generalisation. **Left:** Deployment of MES on real robots. **Right:** Simulated validation in unseen test settings.

Hardware Discrepancies

The multi-expert policy can robustly complete the task under hardware uncertainties from actuators, sensors, and varying dynamic parameters, such as inertia and mass (see accompanying video). For both actuators and sensors, we tested three settings in simulation: adding Gaussian noise to the signal, setting the signal to zero, and randomizing the signal uniformly. Such setting corresponds to the actuators being corrupted by noise, jammed in a zero position, or receiving jerky commands. The measurements become noisy, zero, and erroneous by using the three settings for the sensors.

Despite the existence of hardware discrepancies, the multi-expert policy was still able to achieve stable trotting and to complete lifting and grasping of the object. This shows the robustness of MELA as a feedback policy to realise different motor tasks, i.e., quadruped locomotion and bimanual manipulation.

5.6 Comparison of MELA and MoE

MoE approaches have been reported to scale poorly for control of a high degree-of-freedom systems [149, 2]. The limited expressiveness of the low-dimensional latent space, i.e., the action space, causes an imbalance in expert behaviour that favours some experts and downgrades others [78, 149].

Task Performance

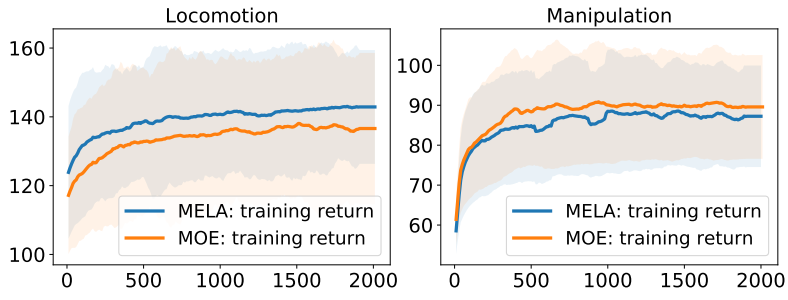
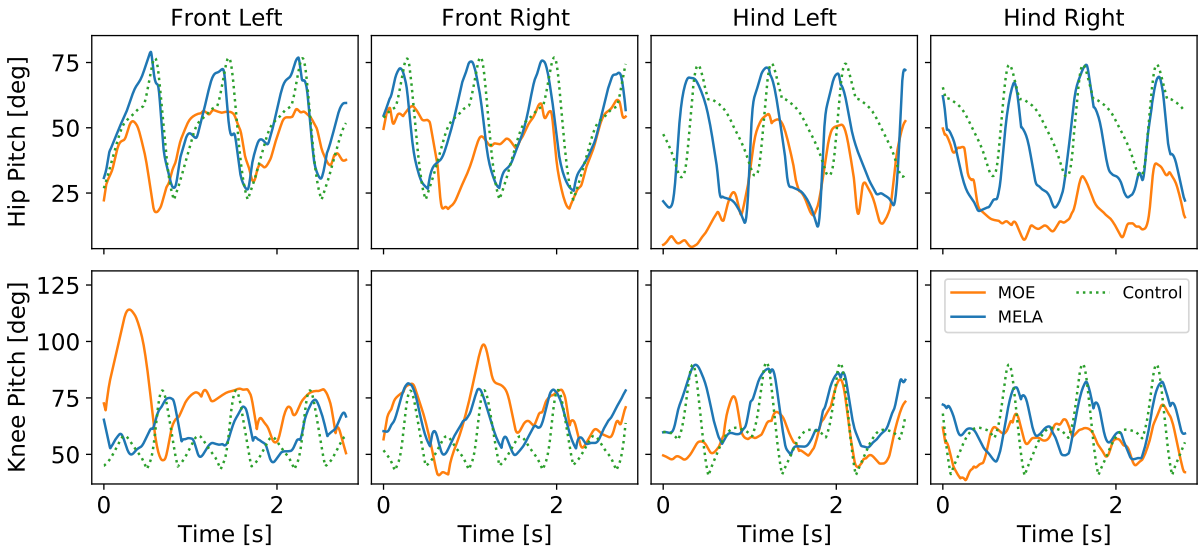
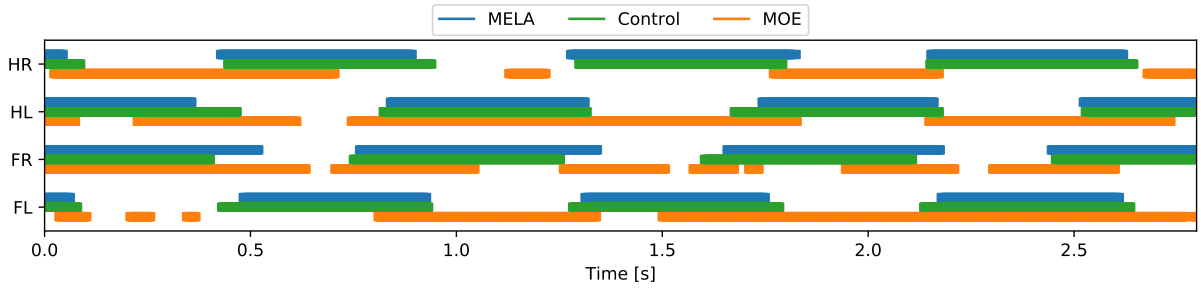


Figure 5.12: Comparison of learning curves of MoE and MELA. The learning curves are averaged over 5 separate training runs and the robot can stand with a reward higher than 100.

We compared MELA with MoE based on the task performance in quadruped locomotion and dual-arm cooperation. The learning curves can be seen in Fig. 5.12.



(a) Joint positions produced by Control, MoE, and MELA.



(b) Gait patterns produced by Control, MoE, and MELA.

Figure 5.13: Comparison of locomotion between control, MoE and MELA. **Top**: Joint position trajectories. **Bottom**: Contact gait pattern of Front Left (FL), Front Right (FR), Hind Left (HL), and Hind Right (HR) feet.

To analyse the transferability and robustness of MELA and MoE across domains, we validated the robust quadruped policy in a different physics simulation, i.e., a simulation to simulation (sim2sim) transfer from pybullet to Gazebo. Gazebo simulator was chosen because the same software infrastructure was used for running the real robot Anymal. The practical and common discrepancies in real world experiments that can cause poor policy performance [66, 145, 172] were introduced in Gazebo, including the mismatch in the physics model, signal noises, feedback latency, friction and damping, and drift in sensory measurements. All quantities used by the MES policies were directly measured and filtered from the robot, or obtained through the state estimation algorithm that runs on the real robot.

While both MoE and MELA can learn robust locomotion in the PyBullet simulation, only MELA was able to transfer the learned policy and perform successful trotting in both a different simulator Gazebo and the real robot. In contrast, MoE was not able to reproduce successful trotting across a different physics simulation or real system and environment, which was shown by a downgraded behavior of a dragging leg that caused a complete fall (see Fig. 5.13 and accompanying video).

In addition to the literature [82, 78, 149], our comparison finds that the

low-dimensional latent space of MoE lacks the complexity for encoding sufficient features robust against physical variations, which is needed for a domain transfer of tasks, such as gait recovery and trotting.

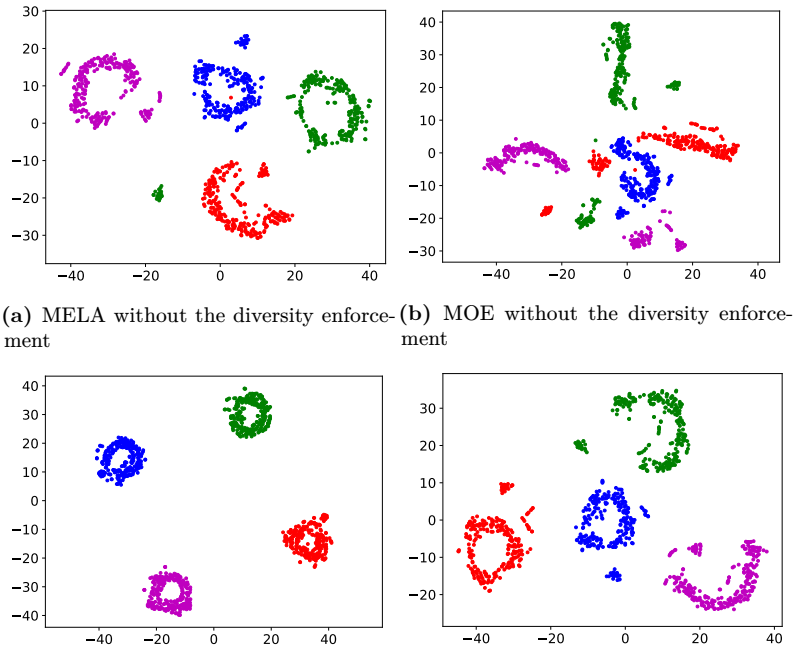
In Fig. 5.13, the qualitative differences and similarities among baseline control, MoE and MELA are shown. The baseline control data was used as imitation reference for both MELA and MoE. The average difference of joint positions between baseline control and MELA was 5°, while MoE deviated by 13° from the baseline control reference. The periodicity and trotting pattern is noticeable in both MELA and baseline control, while MoE was could not reproduce the periodic gait pattern.

Diversity of Skills

We analysed the expert imbalance problem and diversity of skills of MELA and MoE by a t-distributed Stochastic Neighbor Embedding (t-SNE) analysis [180]. T-SNE projects the high-dimensional NN activation on a 2D plane by clustering similar NN activations together but keeping dissimilar data points distant, which can be used to analyse robotic behaviours [6].

The experts' neuron activations (all N experts in Fig. 5.4) during time-step k were stacked as one high-dimensional data point $h_k^i \in \mathbb{R}^{(256+256+12)}$ for all i experts. During one rollout of 250 time steps, 250 data points h_k^i were collected to produce the t-SNE analysis shown in Fig. 5.14.

Fig. 5.14 shows a t-SNE analysis for comparing MELA and MoE with and without the diversity term (5.19). From the distinct clusters and the separation distances (Fig. 5.14c, 5.14d), it can be seen that our proposed diversity technique enforces diversity among experts. MELA shows clustering without the diversity term and has more distinct expert clusters using the diversity term. For MoE, the experts collapse to one indistinguishable cluster if no diversity term is used and form four clusters when diversity is enforced.



(a) MELA without the diversity enforcement

(b) MOE without the diversity enforcement

(c) MELA with the diversity enforcement

(d) MOE with the diversity enforcement

Figure 5.14: Comparison of the t-SNE analysis between MELA and MOE with and without the proposed diversity enforcement, using 4 experts for locomotion (blue and purple) and gait recovery (red and green).

5.7 Conclusion and Future Work

In this work, we proposed: (1) a Multi-Expert Synthesis (MES) framework that can generate motor skills by synthesising expert skills, which is applicable for both robot locomotion and manipulation; (2) an automatic algorithm of selecting relevant physical variables for the state observations of reinforcement learning; and (3) techniques to augment networks and enforce diversify of experts which address the expert imbalance problem in multi-expert approaches.

Both simulation and experiments showed that MES can learn robust quadrupedal locomotion by combining the skills of gait recovery and trotting. Further, MES demonstrated dual-arm manipulation and grasping, where one robot arm pregrasped an object and changed it to a feasible grasp pose, and the other robot arm grasped the object. The robustness of the learned MES policies were rigorously tested by a range of tests in both simulation and real world experiments, which have not been seen during training.

We evaluated two different MES approaches for the locomotion scenario and the results were analysed in terms of gait patterns and diversity of experts using a t-SNE analysis. The analysis suggested that our proposed algorithm for state selection was effective allowing locomotion that exhibits the typical gait patterns of quadrupeds, and that our proposed technique enforcing skill diversity between experts indeed removes expert imbalance.

In future work, we plan to expand the MES structure to incorporate visual perception information to allow robot motions that rely on visual inputs. Furthermore, we intend to learn multi-expert policies that combine experts of different morphologies, e.g., quadruped and bipeds, to control various robots with a unified policy.

Figure 5.15: Potential future application: Valkyrie performing carrying task in NASA's Space Robotics Challenge (SRC) [181].

Conclusion and Outlook 6

6.1 Conclusion

This thesis aimed to enable fast deployment of control algorithms for robotic applications that are both intelligent and versatile in their behaviour as well safe to be deployed in the real world. In this thesis, we proposed methods to combine control with Machine Learning for versatile robotics applications. To this end, we presented (1) designing multi-expert systems as discussed in Chapters 4, 5, and 3, (2) a robust control formulation for Model-Predictive Control (Appendix A) combined with (3) Whole-Body Quadratic Programming (Appendix B) for legged robots. Furthermore, we introduced a (4) fast sample collection procedure for Deep Reinforcement Learning (DRL, Appendix C) and (5) reverse-engineering the learned policy into a controller (Appendix D). Lastly, we developed (5) a method to automatically tune parameters in high-dimensional parameter space (Appendix E).

All frameworks, algorithms, and control architectures were validated on robots; both in simulation and reality. In particular, we showed that

- the hierarchical generative model (Chapter 3) can autonomously learn complex task sequences by mimicking the deep temporal aspects of human motor control. We demonstrate the capabilities of the model on Valkyrie, where she successfully completes a box pickup and delivery task that requires a particular sequence of actions and reasoning of the environment.
- multiple experts can be synthesised into dynamic and fall-resilient locomotion using the Multi-Expert Learning Architecture (MELA) (Chapter 4). We demonstrated the capabilities of MELA on the real robot Jueying, and more extreme, unseen scenarios in simulation.
- Multi-Expert Synthesis (MES) as shown in Chapter 5 can automatically select state space for multiple experts and yields diversity among experts. The successful MES policy was demonstrated on the real robot ANYmal in form of fall-resilient locomotion and in form of dual arm cooperation for the real robot Franka Panda.
- the automatic parameter tuning algorithm proposed in Chapter E can find high-performant parameters for the controller introduced in Appendix A. As a result, the humanoid Valkyrie can robustly locomote over unmodelled, uneven terrain of up to 10° roll and 5° pitch inclination in simulation.
- a DRL policy for trotting of a quadruped and balancing on a humanoid can be trained within an hour and directly deployed on the real robot using our massive parallelisation framework presented in Appendix C. We demonstrate the successful DRL policy on the real robots ANYmal and Valkyrie.

6.2 Outlook

In this work, we showed how learning can yield intelligent, versatile, and robust policies. Further work in the area of Robot Learning can be conducted and are outlined in the following:

Sample efficiency remains a problem.

(1) Sample-efficiency of the learning algorithms remain a problem. Despite the existence of realistic simulators, such as PyBullet [178] or Physx [182], that enable a sim2real transfer, more sample-efficient algorithms would enable real world training and better generalisation.

Meta Learning, Model-Based Reinforcement Learning, and Offline Reinforcement Learning could improve sample-efficiency, but the validity is currently mainly shown in simulation.

For sample-efficient learning, the domains of Meta Learning [183], Model-based Reinforcement Learning (MBRL) [184], and Offline Reinforcement Learning (ORL) [185] are of interest for Robotics. While the viability and effectiveness of most algorithms has currently mainly be shown on the standard MuJoCo benchmark[186], the theoretical nature and results on real world robots [187, 188] are certainly promising.

In Meta Learning, the few-shot learning has shown promising results in Robotics, where robots adapt to similar tasks in simulation [189, 83] and real robots [190]. The sample efficient nature of MBRL compared to model-free RL has been shown in [191]. Furthermore, MBRL was combined with both ORL [192] and Meta Learning [193] to adapt policies online with few samples leading to more general, robust policies and models respectively.

(2) While exploration of actions using the current methods, such as stochastic policies and Reference State Initialisation, is sufficient to learn intelligent policies, they cannot be directly deployed in real-world learning problems, because there is no guarantee of safety and stability during exploration. Thus, for real-world learning safe and stable exploration methods need to be investigated. Possible techniques can be found in [194], [195].

Safe exploration for robotics.

Real-world training would enable policies interacting with liquids and objects that are deformable, fragile, or soft.

(3) Sample-efficient learning methods would allow training robots in the real world. This in turn would allow developing algorithms for tasks that are currently hard to simulate, such as liquid dynamics, interaction with deformable, fragile, or soft objects. Furthermore, better generalisation of the policy is achieved through, if the policy can adapt to new situations with few gradient updates using a little sample amount.

(4) While this thesis presented robust, general policies on real robots, they were not field-tested in real world scenarios. Future work would include applying the techniques developed in this thesis for real world challenges, such as the Subterranean Challenge [196] or DARPA Robotics Challenge [125].

(5) In this work, we presented a state space selection algorithm, that chooses the best states from set of states provided by a domain expert. This line of work can be extended by learning a suitable multi-modal representation for the policy similar to the work in [197]. The aim of this research direction lies in learning a latent representation of task-relevant features, which include multiple modalities, such as visual, prioprioceptive, and tactile information.

Representation Learning of multi-modal features for control policy.

(6) Our work showed promising parallels between Robotics and Neuroscience, where we leveraged the core principles of human motor

control from Neuroscience to develop an implicit hierarchical generative model. Further investigations into deploying well-studied principles from Neuroscience, such as Active Inference [114], would certainly push the frontier of Robotic Intelligence.

Active Inference for Robotics.

(7) Drawing inspiration from Machine Learning , the field of Lifelong Learning is certainly of interest for Robotics as well. Using concepts from Lifelong Learning, a policy could be continuously trained during deployment on the real robot and improve its policy or even learn to adapt to new scenarios, similarly as was demonstrated in MELA (Chapter 4). This idea was developed was presented in [198], and breakthroughs in Machine Learning [199] could enable capabilities similarly to humans.

Lifelong Learning would allow the robot to continuously improve the policy, even after deployment.

Bibliography

[1] K. Yuan, N. Sajid, K. Friston, and Z. Li, "Hierarchical generative modelling for autonomous robots ," Nature Machine Intelligence, under review.

[2] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," Science Robotics, vol. 5, no. 49, 2020.

[3] K. Yuan and Z. Li, "Multi-Expert Synthesis for Versatile Locomotion and Manipulation Skills," Autonomous Robots, under review.

[4] ——, "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.

[5] K. Yuan, Q. Rouxel, Y. Ning, and Z. Li, "Fast Sample Collection Through Massive Parallelisation For Accelerated Deep Reinforcement Learning Development," Springer Nature Applied Sciences, under review.

[6] K. Yuan, C. McGreavy, C. Yang, W. Wolfslag, and Z. Li, "Decoding Motor Skills of Artificial Intelligence and Human Policies: A Study on Humanoid and Human Balance Control," IEEE Robotics & Automation Magazine, 2020.

[7] K. Yuan, I. Chatzinikolaidis, and Z. Li, "Bayesian Optimization for Whole-Body Control of High Degrees of Freedom Robots through Reduction of Dimensionality," IEEE Robotics and Automation Letters, 2019.

[8] R. Bemelmans, G. J. Gelderblom, P. Jonker, and L. De Witte, "Socially assistive robots in elderly care: A systematic review into effects and effectiveness," Journal of the American Medical Directors Association, vol. 13, no. 2, pp. 114–120, 2012.

[9] M. J. Bakari, K. M. Zied, and D. W. Seward, "Development of a multi-arm mobile robot for nuclear decommissioning tasks," International Journal of Advanced Robotic Systems, vol. 4, no. 4, p. 51, 2007.

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

[11] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, and Others, "Team IHMC's lessons learned from the DARPA robotics challenge trials," Journal of Field Robotics, vol. 32, no. 2, pp. 192–208, 2015.

[12] V. Marinoudi, C. G. Sørensen, S. Pearson, and D. Bochtis, "Robotics and labour in agriculture. a context consideration," Biosystems Engineering, vol. 184, pp. 111–121, 2019.

[13] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield et al., "The grand challenges of science robotics," Science robotics, vol. 3, no. 14, p. eaar7650, 2018.

[14] T. Takenaka, T. Matsumoto, T. Yoshiike, and S. Shirokura, "Real time motion generation and control for biped robot - 2nd report: Running gait pattern generation-," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1092–1099, 2009.

[15] M. H. Raibert, Legged robots that balance. MIT press, 1986.

[16] M. T. Mason, Mechanics of robotic manipulation. MIT press, 2001.

[17] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," Science Robotics, vol. 4, no. 26, p. eaau4984, 2019.

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

[19] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," Proceedings of the IEEE, 2016.

[20] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," vol. 2016-November, 2016.

[21] DeepRobotics, "Jueying," http://www.deeprobotics.cn/default/details.

[22] F. Emika, "Franka Panda Robot," https://www.franka.de/technology.

[23] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, no. April, pp. 1405———1411 vol.2, 1991.

[24] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," IEEE Robotics and Automation Letters, vol. 2, no. 1, pp. 10–17, 2016.

[25] 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," 2017.

[26] P. B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," International Conference on Humanoid Robots, 2006.

[27] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online Walking Motion Generation with Automatic Foot Step Placement," Advanced Robotics, 2010.

[28] B. J. Stephens and C. G. Atkeson, "Push recovery by stepping for humanoid robots with force controlled joints," 2010 10th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2010, pp. 52–59, 2010.

[29] M. Krause, J. Englsberger, P. B. Wieber, and C. Ott, "Stabilization of the Capture Point dynamics for bipedal walking based on model predictive control," IFAC Proceedings Volumes (IFAC-PapersOnline), vol. 45, no. 22, pp. 165–171, 2012. [Online]. Available: http://dx.doi.org/10.3182/20120905-3-HR-2030.00165

[30] R. J. Griffin and A. Leonessa, "Model predictive control for dynamic footstep adjustment using the divergent component of motion," Proceedings - IEEE International Conference on Robotics and Automation, vol. 2016-June, pp. 1763–1768, 2016.

[31] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), vol. 2. IEEE, 2003, pp. 1620–1626.

[32] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano, and Others, "The current state and future outlook of rescue robotics," Journal of Field Robotics, vol. 36, no. 7, pp. 1171–1191, 2019.

[33] B. Siciliano and O. Khatib, Handbook of Robotics. Springer, 2008.

[34] 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.

[35] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," Autonomous Robots, vol. 40, no. 3, pp. 429–455, 2016.

[36] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 2589–2594.

[37] 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.

[38] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids). IEEE, 2016, pp. 558–564.

[39] Q. Rouxel, K. Yuan, R. Wen, and Z. Li, "Multi-Contact Motion Retargeting usingWhole-body Optimization of Full Kinematics andSequential Force Equilibrium. ," IEEE/ASME Transactions on Mechatronics., under review, 2022.

[40] R. S. Sutton and A. G. Barto, Introduction to Reinforcement Learning, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[41] C. M. Bishop, Pattern recognition and machine learning. springer, 2006.

[42] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," Journal of Global Optimization, 1998.

[43] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in Adv. in neural information processing systems, 2012.

[44] A. Rai, R. Antonova, S. Song, W. C. Martin, H. Geyer, and C. G. Atkeson, "Bayesian Optimization Using Domain Knowledge on the ATRIAS Biped," International Conference on Robotics and Automation, 2018.

[45] M. Tesch, J. Schneider, and H. Choset, "Using response surfaces and expected improvement to optimize snake robot gait parameters," in Intelligent Robots and Systems, 2011.

[46] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian gait optimization for bipedal locomotion," in Int. Conf. on Learning and Intelligent Optimization, 2014.

[47] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, "Automatic Gait Optimization with Gaussian Process Regression," in IJCAI, 2007.

[48] R. Antonova, A. Rai, and C. G. Atkeson, "Deep kernels for optimizing locomotion controllers," arXiv:1707.09062, 2017.

[49] G. H. Dunteman, Principal components analysis. Sage, 1989.

[50] H. Wold, "Partial least squares," Encyclopedia of statistical sciences, 2004.

[51] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," in Int. Conf. on Robotics and Automation, 2016.

[52] J. Spitz, K. Bouyarmane, S. Ivaldi, and J.-B. Mouret, "Trial-and-error learning of repulsors for humanoid QP-based whole-body control," in International Conference on Humanoid Robotics (Humanoids), 2017.

[53] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton, "High Dimensional Bayesian Optimization using Dropout," in International Joint Conf. on Artificial Intel., 2017.

[54] R. Akrour, D. Sorokin, J. Peters, and G. Neumann, "Local Bayesian Optimization of Motor Skills," in International Conf. on Machine Learning, 2017.

[55] J. C. Bezdek and R. J. Hathaway, "Convergence of alternating optimization," Neural, Parallel & Scientific Computations, 2003.

[56] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," Journal of optimization theory and applications, 2001.

[57] 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.

[58] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine learning, vol. 8, no. 3-4, pp. 229–256, 1992.

[59] 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.

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

[61] 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.

[62] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," International Conference on Machine Learning, 2018.

[63] 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, vol. abs/1509.02971, 2015.

[64] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," arXiv preprint arXiv:1802.09477, 2018.

[65] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning Agile Robotic Locomotion Skills by Imitating Animals," Robotics: Science and Systems, 2020.

[66] 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.

[67] 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.

[68] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 421–436, 2018.

[69] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, and Others, "Learning dexterous in-hand manipulation," The International Journal of Robotics Research, vol. 39, no. 1, pp. 3–20, 2020.

[70] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in International Conference on Machine Learning, 2016, pp. 1329–1338.

[71] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in arXiv preprint arXiv:1709.06560, 2017.

[72] 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.

[73] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," Artificial intelligence, vol. 112, no. 1-2, pp. 181–211, 1999.

[74] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," Journal of artificial intelligence research, vol. 13, pp. 227–303, 2000.

[75] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine, "Latent space policies for hierarchical reinforcement learning," International Conference on Machine Learning, 2018.

[76] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess, "Neural probabilistic motor primitives for humanoid control," International Conference on Representation Learning, 2018.

[77] 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.

[78] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," International Conference on Representation Learning, 2017.

[79] 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.

[80] 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.

[81] R. Sun and T. Peterson, "Multi-agent reinforcement learning: weighting and partitioning," Neural networks, vol. 12, no. 4-5, pp. 727–753, 1999.

[82] 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.

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

[84] L. Liu and J. Hodgins, "Learning to schedule control fragments for physics-based characters using deep q-learning," ACM Transactions on Graphics (TOG), vol. 36, no. 3, pp. 1–14, 2017.

[85] J. Merel, A. Ahuja, V. Pham, S. Tunyasuvunakool, S. Liu, D. Tirumala, N. Heess, and G. Wayne, "Hierarchical visuomotor control of humanoids," International Conference on Representation Learning, 2018.

[86] D. Clever, M. Harant, K. Mombaur, M. Naveau, O. Stasse, and D. Endres, "Cocomopl: A novel approach for humanoid walking generation combining optimal control, movement primitives and learning and its transfer to the real robot HRP-2," IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 977–984, 2017.

[87] V. C. V. Kumar, S. Ha, and C. K. Liu, "Expanding motor skills using relay networks," in Conference on Robot Learning. PMLR, 2018, pp. 744–756.

[88] Y. Lee, S.-H. Sun, S. Somasundaram, E. S. Hu, and J. J. Lim, "Composing complex skills by learning transition policies," in International Conference on Learning Representations, 2018.

[89] Y. Lee, J. Yang, and J. J. Lim, "Learning to coordinate manipulation skills via skill behavior diversification," in International Conference on Learning Representations, 2019.

[90] A. H. Qureshi, J. J. Johnson, Y. Qin, T. Henderson, B. Boots, and M. C. Yip, "Composing task-agnostic policies with deep reinforcement learning," arXiv preprint arXiv:1905.10681, 2019.

[91] N. Hogan, "Planning and execution of multijoint movements," Canadian Journal of Physiology and Pharmacology, vol. 66, no. 4, pp. 508–517, 1988.

[92] D. A. Rosenbaum, J. Vaughan, H. J. Barnes, and M. J. Jorgensen, "Time course of movement planning: selection of handgrips for object manipulation." Journal of Experimental Psychology: Learning, Memory, and Cognition, vol. 18, no. 5, p. 1058, 1992.

[93] N. Li, T.-W. Chen, Z. V. Guo, C. R. Gerfen, and K. Svoboda, "A motor cortex circuit for motor planning and movement," Nature, vol. 519, no. 7541, pp. 51–56, 2015.

[94] K. Svoboda and N. Li, "Neural mechanisms of movement planning: motor cortex and beyond," Current opinion in neurobiology, vol. 49, pp. 33–41, 2018.

[95] C. J. Honey, T. Thesen, T. H. Donner, L. J. Silbert, C. E. Carlson, O. Devinsky, W. K. Doyle, N. Rubin, D. J. Heeger, and U. Hasson, "Slow cortical dynamics and the accumulation of information over long timescales," Neuron, vol. 76, no. 2, pp. 423–434, 2012.

[96] J. D. Murray, A. Bernacchia, D. J. Freedman, R. Romo, J. D. Wallis, X. Cai, C. Padoa-Schioppa, T. Pasternak, H. Seo, D. Lee et al., "A hierarchy of intrinsic timescales across primate cortex," Nature neuroscience, vol. 17, no. 12, pp. 1661–1663, 2014.

[97] J. Diedrichsen, R. Shadmehr, and R. B. Ivry, "The coordination of movement: optimal feedback control and beyond," Trends in cognitive sciences, vol. 14, no. 1, pp. 31–39, 2010.

[98] K. J. Friston, T. Parr, and B. de Vries, "The graphical brain: belief propagation and active inference," Network Neuroscience, vol. 1, no. 4, pp. 381–414, 2017.

[99] M. Baltieri and C. L. Buckley, "Generative models as parsimonious descriptions of sensorimotor loops," arXiv preprint arXiv:1904.12937, 2019.

[100] J. Merel, M. Botvinick, and G. Wayne, "Hierarchical motor control in mammals and machines," Nature communications, vol. 10, no. 1, pp. 1–12, 2019.

[101] J. H. Jackson, "On certain relations of the cerebrum and cerebellum (on rigidity of hemiplegia and on paralysis agitans)," Brain, vol. 22, no. 4, pp. 621–630, 1899.

[102] T. Parr, J. Limanowski, V. Rawji, and K. Friston, "The computational neurology of movement under active inference," Brain, 2021.

[103] G. Pezzulo, F. Rigoli, and K. J. Friston, "Hierarchical active inference: a theory of motivated control," Trends in cognitive sciences, vol. 22, no. 4, pp. 294–306, 2018.

[104] P. Perrier, D. J. Ostry, and R. Laboissière, "The equilibrium point hypothesis and its application to speech motor control," Journal of Speech, Language, and Hearing Research, vol. 39, no. 2, pp. 365–378, 1996.

[105] A. G. Feldman and M. F. Levin, "The equilibrium-point hypothesis–past, present and future," Progress in motor control, pp. 699–726, 2009.

[106] M. M. Botvinick, Y. Niv, and A. G. Barto, "Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective," Cognition, vol. 113, no. 3, pp. 262–280, 2009.

[107] L. Aitchison and M. Lengyel, "With or without you: predictive coding and bayesian inference in the brain," Current opinion in neurobiology, vol. 46, pp. 219–227, 2017.

[108] R. A. Adams, S. Shipp, and K. J. Friston, "Predictions not commands: active inference in the motor system," Brain Structure and Function, vol. 218, no. 3, pp. 611–643, 2013.

[109] E. Bizzi, F. A. Mussa-Ivaldi, and S. Giszter, "Computations underlying the execution of movement: a biological perspective," Science, vol. 253, no. 5017, pp. 287–291, 1991.

[110] E. Marder and D. Bucher, "Central pattern generators and the control of rhythmic movements," Current biology, vol. 11, no. 23, pp. R986–R996, 2001.

[111] S. Grillner, P. Wallén, K. Saitoh, A. Kozlov, and B. Robertson, "Neural bases of goal-directed locomotion in vertebrates—an overview," Brain research reviews, vol. 57, no. 1, pp. 2–12, 2008.

[112] M. Jueptner, C. Frith, D. Brooks, R. Frackowiak, and R. Passingham, "Anatomy of motor learning. ii. subcortical structures and learning by trial and error," Journal of neurophysiology, vol. 77, no. 3, pp. 1325–1337, 1997.

[113] M. A. Sommer, "The role of the thalamus in motor control," Current opinion in neurobiology, vol. 13, no. 6, pp. 663–670, 2003.

[114] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo, "Active inference: a process theory," Neural computation, vol. 29, no. 1, pp. 1–49, 2017.

[115] H. Hasselt, "Double q-learning," Advances in neural information processing systems, vol. 23, pp. 2613–2621, 2010.

[116] 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.

[117] B. Siciliano, O. Khatib, and T. Kröger, Springer handbook of robotics. Springer, 2008, vol. 200.

[118] J. B. Rawlings, D. Q. Mayne, and M. Diehl, Model predictive control: theory, computation, and design. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[119] 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, 2007.

[120] T. Drew, J. Kalaska, and N. Krouchev, "Muscle synergies during locomotion in the cat: A model for motor cortex control," Journal of Physiology, vol. 586, 2008.

[121] M. Mischiati, H. T. Lin, P. Herold, E. Imler, R. Olberg, and A. Leonardo, "Internal models direct dragonfly interception steering," Nature, vol. 517, 2015.

[122] 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, 2020.

[123] H. Markram, "The blue brain project," 2006.

[124] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," 2013.

[125] "Darpa robotics challenge (drc)." [Online]. Available: https://www.darpa.mil/program/darpa-robotics-challenge

[126] C. G. Atkeson, B. P. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, and M. DeDonato, "What Happened at the DARPA Robotics Challenge and Why," 2015.

[127] N. A. Bernstein, "The co-ordination and regulation of movements: Conclusions towards the study of motor co-ordination," 1967.

[128] M. L. Latash, "Stages in learning motor synergies: A view based on the equilibrium-point hypothesis," Human Movement Science, vol. 29, 2010.

[129] J. Ramos and S. Kim, "Dynamic locomotion synchronization of bipedal robot and human operator via bilateral feedback teleoperation," Science Robotics, vol. 4, 2019.

[130] D. Dimitrov, A. Sherikov, and P. B. Wieber, "A sparse model predictive control formulation for walking motion generation," 2011.

[131] H. W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," vol. 11, 2015.

[132] J. D. Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," 2018.

[133] 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, 2015.

[134] M. Neunert, M. Stauble, 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, 2018.

[135] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body Motion Planning with Simple Dynamics and Full Kinematics," Humanoids, 2014.

[136] 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, 2018.

[137] 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, vol. 5, 2020.

[138] A. Cully, J. Clune, D. Tarapore, and J. B. Mouret, "Robots that can adapt like animals," Nature, vol. 521, 2015.

[139] E. O. Neftci and B. B. Averbeck, "Reinforcement learning in artificial and biological systems," 2019.

[140] K. Bouyarmane, S. Caron, A. Escande, and A. Kheddar, "Multi-contact motion planning and control," 2017.

[141] M. Posa, "Optimization for control and planning of multi-contact dynamic motion," in PhD Thesis.

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

[143] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning," ACM Transaction on Graphics, 2017.

[144] 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, p. 143, 2018.

[145] 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," in Robotics: Science and Systems, 2018.

[146] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, "Using deep reinforcement learning to learn high-level policies on the atrias biped," vol. 2019-May, 2019.

[147] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. V. D. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," 2019.

[148] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," 2003.

[149] 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.

[150] A. G. Feldman, V. Goussev, A. Sangole, and M. F. Levin, "Threshold position control and the principle of minimal interaction in motor actions," 2007.

[151] E. Bizzi, N. Hogan, F. A. Mussa-Ivaldi, and S. Giszter, "Does the nervous system use equilibrium-point control to guide single and multiple joint movements?" Behavioral and Brain Sciences, vol. 15, 1992.

[152] F. L. Moro, N. G. Tsagarakis, and D. G. Caldwell, "Efficient human-like walking for the compliant humanoid coman based on kinematic motion primitives (kmps)," 2012.

[153] A. T. Spröwitz, M. Ajallooeian, A. Tuleu, and A. J. Ijspeert, "Kinematic primitives for walking and trotting gaits of a quadruped robot with compliant legs," Frontiers in Computational Neuroscience, vol. 8, 2014.

[154] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," ACM Trans. Graph., 2017.

[155] S. Starke, H. Zhang, T. Komura, and J. Saito, "Neural state machine for character-scene interactions," ACM Transactions on Graphics, vol. 38, 2019.

[156] J. W. Hurst and A. A. Rizzi, "Series compliance for an efficient running gait," IEEE Robotics and Automation Magazine, vol. 15, 2008.

[157] C. Gilbert, "Visual control of cursorial prey pursuit by tiger beetles (cicindelidae)," Journal of Comparative Physiology - A Sensory, Neural, and Behavioral Physiology, vol. 181, 1997.

[158] J. M. Camhi and E. N. Johnson, "High-frequency steering maneuvers mediated by tactile cues: Antennal wall-following in the cockroach," Journal of Experimental Biology, vol. 202, 1999.

[159] J. Borràs, C. Mandery, and T. Asfour, "A whole-body support pose taxonomy for multi-contact humanoid robot motions," Science Robotics, vol. 2, 2017.

[160] 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.

[161] A. R. Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, "Setting up a reinforcement learning task with a real-world robot," 2018.

[162] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2016.

[163] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2261–2268, 2018.

[164] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," Journal of Machine Learning Research, vol. 16, no. 1, pp. 1437–1480, 2015.

[165] M. Mitchell, An introduction to genetic algorithms. MIT press, 1998.

[166] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95-International Conference on Neural Networks, vol. 4. IEEE, 1995, pp. 1942–1948.

[167] S. Ha and C. K. Liu, "Evolutionary optimization for parameterized whole-body dynamic motor skills," in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016, pp. 1390–1397.

[168] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[169] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," arXiv preprint arXiv:1906.08253, 2019.

[170] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in Machine learning proceedings 1990. Elsevier, 1990, pp. 216–224.

[171] 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.

[172] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1–8.

[173] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in Aaai, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[174] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and Others, "Soft actor-critic algorithms and applications," arXiv preprint arXiv:1812.05905.

[175] Z. Sun, K. Yuan, W. Hu, C. Yang, and Z. Li, "Learning Pregrasp Manipulation of Objects from Ungraspable Poses," International Conference on Robotics and Automation, 2020.

[176] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 3359–3365.

[177] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," International Conference on Representation Learning, 2018.

[178] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation in robotics, games and machine learning." [Online]. Available: https://pybullet.org/

[179] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3. IEEE, 2004, pp. 2149–2154.

[180] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," Journal of machine learning research, vol. 9, no. Nov, pp. 2579–2605, 2008.

[181] NASA, "NASA's Centennial Challenges: Space Robotics Challenge ," https://www.nasa.gov/directorates/spacetech/centennial_challenges/space_robotics/about.html.

[182] NVIDIAGameWorks, NVIDIA PhysX SDK 4.1. [Online]. Available: https://github.com/NVIDIAGameWorks/PhysX

[183] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," arXiv preprint arXiv:2004.05439, 2020.

[184] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," Journal of Intelligent & Robotic Systems, vol. 86, no. 2, pp. 153–173, 2017.

[185] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," arXiv preprint arXiv:2005.01643, 2020.

[186] OpenAI, MuJoCo - Continuous control tasks, running in a fast physics simulator. [Online]. Available: https://gym.openai.com/envs/#mujoco

[187] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn et al., "Actionable models: Unsupervised offline reinforcement learning of robotic skills," arXiv preprint arXiv:2104.07749, 2021.

[188] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in Conference on Robot Learning. PMLR, 2020, pp. 1101–1112.

[189] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," International Conference on Machine Learning, 2017.

[190] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," arXiv preprint arXiv:1803.11347, 2018.

[191] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," arXiv preprint arXiv:1907.02057, 2019.

[192] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma, "Mopo: Model-based offline policy optimization," arXiv preprint arXiv:2005.13239, 2020.

[193] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in Conference on Robot Learning. PMLR, 2018, pp. 617–629.

[194] M. Pecka and T. Svoboda, "Safe exploration techniques for reinforcement learning–an overview," in International Workshop on Modelling and Simulation for Autonomous Systems. Springer, 2014, pp. 357–375.

[195] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," arXiv preprint arXiv:1801.08757, 2018.

[196] "Darpa subterranean challenge." [Online]. Available: https://www.subtchallenge.com/

[197] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 8943–8950.

[198] S. Thrun and T. M. Mitchell, "Lifelong robot learning," Robotics and autonomous systems, vol. 15, no. 1-2, pp. 25–46, 1995.

[199] D. L. Silver, Q. Yang, and L. Li, "Lifelong machine learning systems: Beyond learning algorithms," in 2013 AAAI spring symposium series, 2013.

[200] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," Automatica, vol. 36, no. 6, pp. 789–814, 2000. [Online]. Available: http://control.disp.uniroma2.it/galeani/CO/materiale/automatica2000.pdf

[201] A. W. Winkler, "Optimization-based motion planning for legged robots," Ph.D. dissertation, ETH Zurich, 2018.

[202] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in 2014 IEEE-RAS international conference on humanoid robots. IEEE, 2014, pp. 279–286.

[203] M. Vukobratović and B. Borovac, "Zero-Moment Point — Thirty Five Years of Its Life," International Journal of Humanoid Robotics, vol. 01, no. 01, pp. 157–173, 2004. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0219843604000083

[204] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3D Walking Based on Online Optimization," International Conference on Humanoid Robots, 2013.

[205] R. Tedrake, S. Kuindersma, R. Deits, and K. Miura, "A closed-form solution for real-time optimal gait generation and feedback stabilization," 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 2015.

[206] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and foothold optimization for quadruped locomotion," Proceedings - IEEE International Conference on Robotics and Automation, pp. 5308–5313, 2017.

[207] M. Morisawa, S. Kajita, F. Kanehiro, K. Kaneko, K. Miura, and K. Yokoi, "Balance control based on Capture Point error compensation for biped walking on uneven terrain," IEEE-RAS International Conference on Humanoid Robots, pp. 734–740, 2012.

[208] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics," IEEE International Conference on Intelligent Robots and Systems, pp. 4420–4427, 2011.

[209] S. Kajita, M. Morisawa, K. Harada, and K. Kaneko, "Biped Walking Pattern Generator allowing Auxiliary ZMP Control," International Conference on Intelligent Robots and Systems, 2006.

[210] B. J. Stephens, "Push Recovery Control for Force-Controlled Humanoid Robots," Biomechanics, no. June, p. 177, 2011.

[211] J. A. Castano, A. Hernandez, Z. Li, C. Zhou, N. G. Tsagarakis, D. G. Caldwell, and R. De Keyser, "Implementation of Robust EPSAC on dynamic walking of COMAN Humanoid," IFAC Proceedings Volumes, vol. 47, no. 3, pp. 8384–8390, 2014.

[212] J. A. Castano, A. Hernandez, Z. Li, N. G. Tsagarakis, D. G. Caldwell, and R. De Keyser, "Enhancing the robustness of the EPSAC predictive control using a Singular Value Decomposition approach," Robotics and Autonomous Systems, vol. 74, pp. 283–295, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2015.09.001

[213] E. W. Cheney and D. R. Kincaid, Numerical Mathematics and Computing. Brooks/Cole Publishing Co., 2007.

[214] J. A. Castano, Z. Li, C. Zhou, N. Tsagarakis, and D. Caldwell, "Dynamic and reactive walking for humanoid robots based on foot placement control," International Journal of Humanoid Robotics, vol. 13, no. 02, p. 1550041, 2016.

[215] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion," IEEE Transactions on Robotics, vol. 31, no. 2, pp. 355–368, 2015.

[216] S. Caron, Q. C. Pham, and Y. Nakamura, "ZMP Support Areas for Multicontact Mobility Under Frictional Constraints," Transactions on Robotics, 2017.

[217] L. Sentis and O. Khatib, "Control of free-floating humanoid robots through task prioritization," International Conference on Robotics and Automation, 2005.

[218] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no. 7553, pp. 436–444, 2015.

[219] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in 2015 IEEE international conference on robotics and automation (ICRA). IEEE, 2015, pp. 4397–4404.

[220] D. Kang and J. Hwangho, SimBenchmark - Physics engine benchmark for robotics applications: RaiSim vs. Bullet vs. ODE vs. MuJoCo vs. DartSim. [Online]. Available: https://leggedrobotics.github.io/SimBenchmark/

[221] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, and Others, "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.

[222] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," arXiv preprint arXiv:1907.02057, 2019.

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

[224] E. Coumans and Others, "Bullet physics library," Open source: bulletphysics. org, vol. 15, no. 49, p. 5, 2013.

[225] gRPC Authors 2020, gRPC - A high-performance, open source universal RPC framework. [Online]. Available: https://grpc.io/

[226] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," arXiv preprint arXiv:1606.01540, 2016.

[227] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[228] 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.

[229] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in International conference on robotics and automation. IEEE, 2017.

[230] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 634–641.

[231] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," Journal of neuroscience, vol. 5, no. 7, pp. 1688–1703, 1985.

[232] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, and Others, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," Science, vol. 362, no. 6419, pp. 1140–1144, 2018.

[233] P. Zaytsev, W. Wolfslag, and A. Ruina, "The boundaries of walking stability: viability and controllability of simple models," IEEE Transactions on Robotics, 2018.

[234] C. McGreavy, K. Yuan, D. Gordon, K. Tan, W. Wolfslag, S. Vijayakumar, and Z. Li, "Unified Push Recovery Fundamentals: Inspiration from Human Study," submitted to Robotics and Automation Letters, accessible through https://cam586.github.io/mcgreavy_2019_RA-L.pdf, 2019.

[235] 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.

[236] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," The International Journal of Robotics Research, vol. 33, no. 7, pp. 1006–1028, 2014.

[237] D. Gordon, G. Henderson, and S. Vijayakumar, "Effectively quantifying the performance of lower-limb exoskeletons over a range of walking conditions," Frontiers in Robotics and AI, 2018.

[238] J. A. Nyakatura, K. Melo, T. Horvat, K. Karakasiliotis, V. R. Allen, A. Andikfar, E. Andrada, P. Arnold, J. Lauströer, J. R. Hutchinson, and Others, "Reverse-engineering the locomotion of a stem amniote," Nature, 2019.

[239] H. Forssberg, "Ontogeny of human locomotor control I. Infant stepping, supported locomotion and transition to independent locomotion," Experimental Brain Research, vol. 57, no. 3, pp. 480–493, 1985.

[240] C. Zhou, Z. Li, X. Wang, N. Tsagarakis, and D. Caldwell, "Stabilization of bipedal walking based on compliance control," Autonomous Robots, 2016.

[241] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," International Journal of Humanoid Robotics, 2005.

[242] C. Zhou, X. Wang, Z. Li, and N. Tsagarakis, "Overview of Gait Synthesis for the Humanoid COMAN," Journal of Bionic Engineering, vol. 14, no. 1, pp. 15–25, 2017.

[243] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," International Journal of Robotics Research, vol. 30, no. 2, pp. 236–258, 2011.

[244] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in International Conference on Learning and Intelligent Optimization, 2011.

[245] Z. Tang, C. Zhou, and Z. Sun, "Humanoid walking gait optimization using GA-based neural network," in Int. Conference on Natural Computation, 2005.

[246] C. Niehaus, T. Röfer, and T. Laue, "Gait optimization on a humanoid robot using particle swarm optimization," in Second Workshop on Humanoid Soccer Robots, 2007.

[247] H. Dallali, P. Kormushev, Z. Li, and D. Caldwell, "On global optimization of walking gaits for the compliant humanoid robot, COMAN using reinforcement learning," Cybernetics and Information Technologies, 2012.

[248] C. Zhou, C. Fang, X. Wang, Z. Li, and N. Tsagarakis, "A generic optimization-based framework for reactive collision avoidance in bipedal locomotion," in International Conference on Automation Science and Engineering (CASE), 2016.

[249] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," arXiv:1603.08785.

[250] C. E. Rasmussen and C. K. I. Williams, Gaussian Processes in Machine Learning. MIT Press, 2006.

[251] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," Journal of Machine Learning Research, 2002.

[252] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," arXiv:0912.3995, 2009.

[253] S. Caron, Q.-C. Pham, and Y. Nakamura, "Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas," in Int. Conference on Robotics and Automation, 2015.

[254] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with Gaussian processes," in International Conference on Machine Learning, 2015.

Appendices

Appendix Preface

The following Appendix chapters are further contributions of this chapters. To clearly distinguish between multi-expert learning approaches (main part of this thesis) and methods that enable these, the enabling methods are in the Appendix.

In the Appendix A – E, our contributions in the domain of control theory and learning for robotics, and methods enabling multi-expert learning are presented. The content includes expert generation and include: MPC, Joint Control, Deep Reinforcement Learning, Reverse Engineering of AI policies, and automatic parameter tuning. The following Appendix chapters constitute an edited version of the publications [4, 5, 6, 7].

Stability Control A

The following section introduces the concept of Model-Predictive Control (MPC) for mid-level stability control, and presents an improved formulation for MPC of floating-based systems [4]. Please note, that the proposed formulation for MPC is applicable to both bipeds and quadrupeds as both dynamics can be simplified to a Linear Inverted Pendulum formulation.

MPC achieves two goals: open-loop planning and closed-loop feedback control. Planning involves finding an optimal trajectory (blue curve in Fig. A.1) under constraints, e.g., physical, dynamics, and stability constraints. The constraint optimisation nature of MPC allows tracking a given reference trajectory (green curve in Fig. A.1) through physically feasible motions while guaranteeing stability. Feedback control is achieved by continuously solving the constrained optimisation problem over a given prediction horizon in a closed loop fashion using the newest state information.

It can be shown that in an MPC scheme the closed loop is stable even under uncertainties such as model error and external disturbances [200]. The target reference can originate from various sources such as Trajectory Optimisation (TO) [201], foot step planning [202], or in form of actions from Neural Networks. The media to this chapter can be found on https://www.youtube.com/watch?v=7uwtorW_kzE.

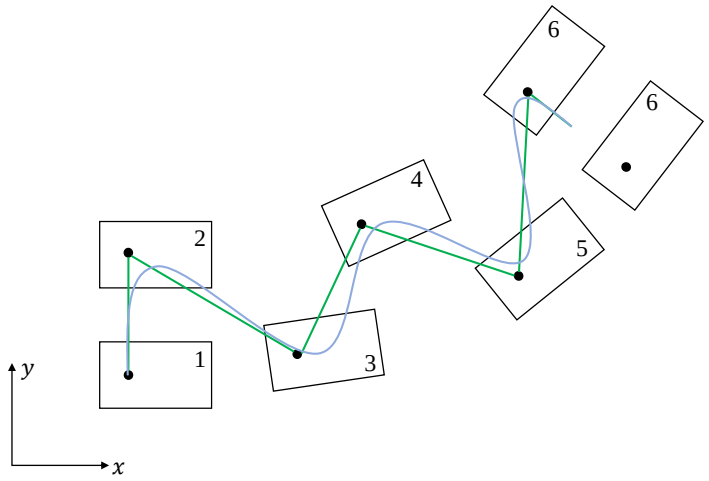


Figure A.1: Enumerated sequence of support foot placement. The blue curve represents the COM trajectory. The green curve is the reference trajectory.

A.1 Model-Predictive Control

The performances achieved by legged robots during the DARPA Robotics Challenge (DRC) suggest that more improvements in control shall still be made [126]. The common approach for locomotion among all high ranked DRC humanoids is to split motion planning and stabilizing control into two separate parts [11, 35]. First, an optimal walking pattern under the given environmental constraint is found and used as reference in a feed-forward manner. Then a closed-loop control tracks this reference for the execution, i.e. stabilization of the planned motion pattern under disturbances and uncertainties. This decoupled use of feed-forward and feedback control leads to suboptimal performances, especially in the presence of disturbances.

This motivates the use of Model Predictive Control (MPC) to remedy this problem [200]. MPC achieves closed-loop control by continuously solving the feed-forward optimization using the feedback of current states and constraints. In MPC, high-level tasks can be embedded in the cost function of the optimization problem, and stability of the motion is ensured by designing suitable constraints. This study aims to formulate a coherent model for MPC suitable for both online/offline planning as well as real-time feedback control against external perturbations.

Zero-Moment Point (ZMP) based approaches generate stable gaits by keeping the ZMP within the Support Polygon (SP) [203]. This is usually enforced by either closely tracking the ZMP, or by constraining the ZMP in an optimization problem. When the solver time is secondary for planning, more complete and complex whole-body models could be used to plan for a long horizon resulting in a small deviation from the optimal trajectory. In contrast, MPC solves the optimization problem using simple models, and stability around the planned trajectory is guaranteed by optimizing at a high frequency. The most common simple model is the Linear Inverted Pendulum Model (LIPM) [23]. Despite its limitations on the Centre of Mass (COM) height and the necessity of coplanar contacts between feet and ground, it is extensively used in optimization for dynamic locomotion: a Differential Dynamic Programming (DDP) approach was proposed in [204], and the work in [205] used the LIPM to calculate the closed-form solution of an LQR problem. Lastly, the LIPM dynamics can also be used to represent the ZMP constraint [206] and to design feedback controllers [35, 207].

In the past decades, the importance of MPC for dynamic locomotion has increased. First an unconstrained MPC scheme was proposed in [31] for walking pattern generation of bipedal humanoids. Improved disturbance rejection was then achieved by constraining the ZMP [26]. Furthermore MPC was used for automatic foot placement [27], Push Recovery [28], and Capture Point (CP) tracking [29, 30].

We propose an MPC framework that particularly distinguishes the external perturbation in our formulation making it real-world applicable, and also has dual-uses for Motion Planning and Feedback Control. Our formulation is able to generate a stable COM motion given Centre of Pressure (COP) or CP references while satisfying the ZMP constraints. Our contributions are summarized as follows:

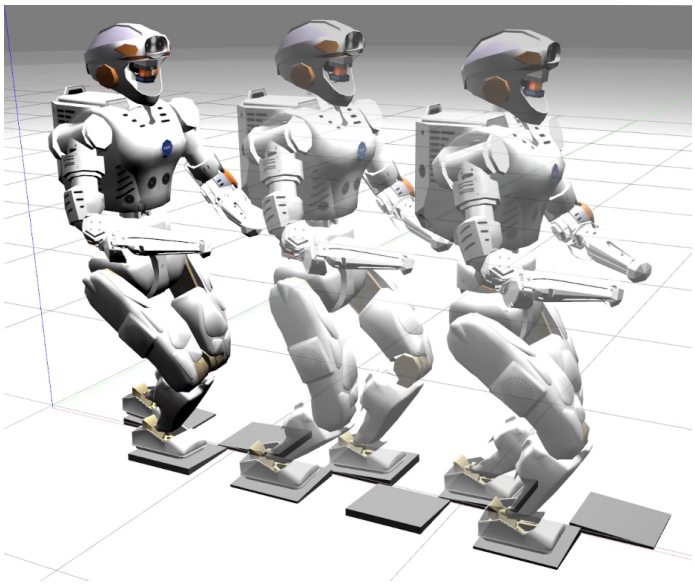


Figure A.2: Physics simulation in Gazebo of NASA's Valkyrie walking over uneven terrain using the proposed MPC framework.

- Analysis of numerical and physical issues in existing MPC frameworks;
- A proposed formulation that solves the numerical and physical problems analyzed above;
- Integration of the proposed MPC with whole-body control for real-world application (Fig. A.2);
- Benchmarking of the proposed MPC framework against existing ones.

In Section A.2, the different use of dynamics models for optimization are revisited and compared, and a better formulation is proposed to explicitly differentiate the COM acceleration caused by external pushes. An MPC framework for Walking Pattern Generation and Feedback Control is proposed in Section A.3. Then the numerical stability of different optimization formulations are analysed, evaluated, and discussed in Section A.4. A comparison of these controllers in the presence of disturbances is then studied in Section A.5. Lastly we conclude our study in Section A.6.

A.2 Dynamic Models

The ability to model the robot's dynamics allows control design to provide stability and robustness to uncertainties. The idea of the LIPM is based on the assumption that the dominant dynamics of a biped robot can be described by a single inverted pendulum [23] (Fig. A.3), which is fast-solvable for optimization problems.

Linear Inverted Pendulum Model: COP as Control Input

For a constant COM height z_c, the dynamics of the Inverted Pendulum become linear and the states of the COM are determined by a second

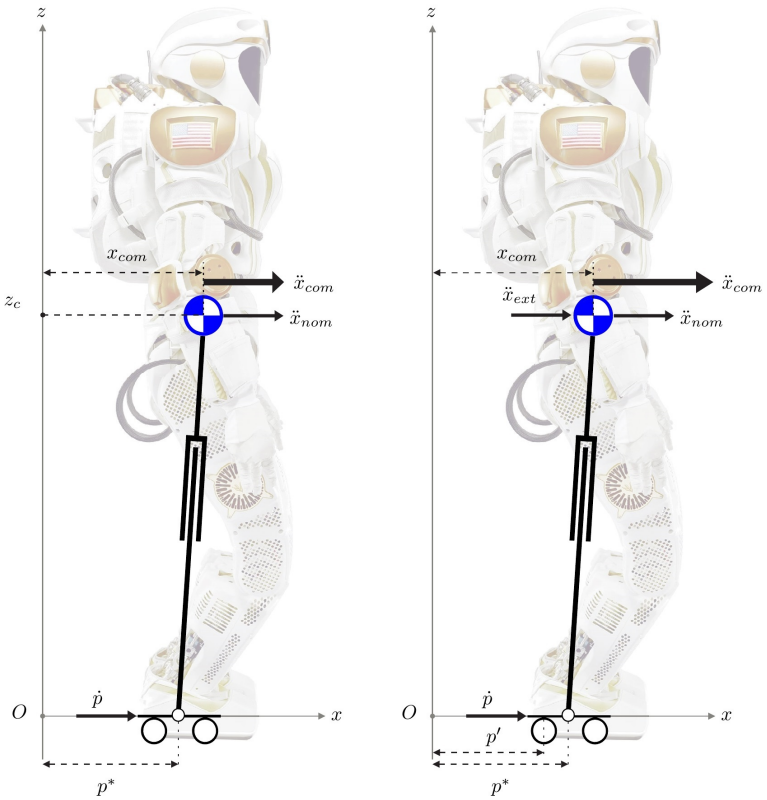


Figure A.3: Representation of NASA's Valkyrie's dynamics as an LIPM. The real COP p^* is controlled by COP velocity $\dot{p}$ through a torque-control based tracking controller. The resultant COM acceleration $\ddot{x}_{com}$ is composed by the acceleration generated by the real COP and the external disturbance $\ddot{x}_{ext}$ (left: no disturbance, right: with disturbance). The fictitious ZMP p' is represented by the CTM.

order differential equation [31]:

$$\ddot{x} = \frac{g}{z_c}(x - p_x), \tag{A.1}$$

with COM position x, velocity $\dot{x}$, acceleration $\ddot{x}$, height z_c, gravity constant g, and ZMP p_x. The LIPM describes the motion of the COM propelled by the Centre of Pressure (COP). Due to symmetry of the LIPM motion, only the x component is analysed; the y component behaves analogously.

In the literature the LIP based model (A.1) was represented with both a two-dimensional $x = [x, \dot{x}]^T$ [29, 208], and a three-dimensional state space vector $x = [x, \dot{x}, p]^T$ [209, 28]. The two-dimensional state space version is used for Capture Point (CP) tracking with the output $y = [1, \sqrt{z_c/g}]x$. The three-dimensional version was used both for control design [209, 28] and state estimation [210]. The ZMP's dynamics exhibit an integrator behaviour using the rate change $\dot{p}$ directly as input into the system.

Cart-Table Model: Jerk as Control Input

A common way to plan a stable gait is to predefine a ZMP trajectory within the Support Polygon that will be then tracked closely, where the Cart-Table Model (CTM) [31] serves as an inverse dynamics model of the LIPM (A.1) that enables accurate ZMP tracking by the output feedback of the simulated ZMP p given COM position x and acceleration $\ddot{x}$:

$$p = x - \frac{z_c}{g}\ddot{x}. \tag{A.2}$$

Although the CTM is suitable for gait planning, it has two limitations: 1.) its input-output causality does not reflect the causality of the actual robot: the real system does not have a real control input that can be arbitrarily manipulated at the COM. In fact, the real system behaves like an Inverted Pendulum, where the COM is driven by the COP. 2.) Due to the reversed causality the COP (ZMP) position is calculated wrongly in the case of disturbance. The real COP p^*, which can be measured and directly controlled, generates the nominal COM state $x_{nom} = [x_{nom}, \dot{x}_{nom}, \ddot{x}_{nom}]$ (Fig. A.3 left). For the undisturbed system, COM state x_{com} and the COM motion generated by COP x_{nom} are identical, i.e. $x_{com} = x_{nom}$. However, in case of an external disturbance (Fig. A.3 right) that causes acceleration $\ddot{x}_{ext}$, the COM state evolves according to the dynamics generated by the resulted acceleration $\ddot{x}_{com} = g(x_{com} - p^*)/z_c + \ddot{x}_{ext}$, while the COP p^* is controlled by a COP tracking controller and independent from the external push. The CTM uses the resultant COM state to rule back the cause of this acceleration, i.e. where the COP/ZMP is. However, once the external acceleration is injected, the CTM no longer accurately captures this behaviour and thus yields a wrong COP calculation as $p' = x_{com} - z_c(\ddot{x}_{GRF} + \ddot{x}_{ext})/g = p^* - z_c\ddot{x}_{ext}/g$.

Proposed formulation

The fact that the COM state does not generally reflect the COP of a real system due to disturbance motivates the incorporation of the COP as an additional state in order to correctly model the real physical process. By differentiating (A.1), we can see that the *jerk* is determined by COM velocity and rate change of COP $\dot{p}$:

$$\dddot{x} = \frac{g}{z_c}(\dot{x} - \dot{p}). \tag{A.3}$$

Also, by rearranging (A.3), the rate change of COP $\dot{p}$ can be represented by the COM state $\dot{x}$ and the COM *jerk* $\dddot{x}$ resulted by the influence of the COP position:

$$\dot{p} = \dot{x} - \frac{z_c}{g}\dddot{x}. \tag{A.4}$$

In theory, both $\dot{p}$ and $\dddot{x}$ are mutually exchangeable control inputs to represent the newly introduced dynamics (A.4) of p:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \frac{g}{z_c} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -\frac{g}{z_c} \\ 1 \end{bmatrix} \dot{p} \tag{A.5}$$

$$y_{COP} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} x \tag{A.6}$$

$$y_{CP} = \begin{bmatrix} 1 & \sqrt{\frac{z_c}{g}} & 0 & 0 \end{bmatrix} x. \tag{A.7}$$

Our proposed equivalent formulation is:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ -\frac{z_c}{g} \end{bmatrix} \dddot{x} \tag{A.8}$$

$$y_{COP} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} x \tag{A.9}$$

$$y_{CP} = \begin{bmatrix} 1 & \sqrt{\frac{z_c}{g}} & 0 & 0 \end{bmatrix} x. \tag{A.10}$$

To sum up, the COM dynamics can be controlled by two different control efforts: controlling the COP p (A.1) or its derivative $\dot{p}$, or controlling the jerk $\dddot{x}$ (A.2). Both models require a COP controller that is able to track the desired COP, which is compatible with modern torque controlled hardware. The different formulations that were used in this study are summarised in Table A.1.

Table A.1: Definition of formulation A, B, C, D', D as LIPM and E as CTM with their control inputs and origins.

#	Model	Control Input	Origin
A	LIPM: $x = [x, \dot{x}]^T$	p	[208]
B	LIPM: $x = [x, \dot{x}]^T$	p	[29]
C	LIPM: $x = [x, \dot{x}, p]^T$	$\dot{p}$	[28], [209]
D'	LIPM: $x = [x, \dot{x}, \ddot{x}, p]^T$	$\dot{p}$	Proposed (A.5)
D	LIPM: $x = [x, \dot{x}, \ddot{x}, p]^T$	$\dddot{x}$ (representing $\dot{p}$)	Proposed (A.8)
E	CTM: $x = [x, \dot{x}, \ddot{x}]^T$	$\dddot{x}$	[31], [26]

We opt for formulation D and will show that the proposed state space formulation (A.8) has two main advantages compared with existing ones: **1.** analysis in Section A.4 will show that the proposed formulation is numerically more stable than the LIPM dynamics based state space formulations A, B, C, D', and thus the proposed formulation D can be used for both short and long prediction horizons; **2.** the disturbance rejection is greatly improved by introducing the new state p which will be shown in Section A.5.

A.3 Model-Predictive Control Framework

MPC enables loop closure by optimizing over a predefined prediction horizon N at a given frequency $1/T$ considering constraints. This section formulates a constrained optimization problem which is then incorporated into an MPC framework consisting of an Model-Predictive Controller, Whole-Body Controller, and a State Estimator.

Optimization Problem Formulation

The optimization problem, which the MPC is continuously optimizing over, is formulated as a Quadratic Programming (QP) problem due to its convex property and the existence of fast QP solvers. For this, the previously introduced continuous state space systems is discretised at a sampling time T. For any discrete Linear Time-Invariant (LTI) system

$$x_{k+1} = A_d x_k + B_d u_k, \quad z_k = C_d x_k \tag{A.11}$$

the outputs $Z_{k+1} = [z_{k+1}, ..., z_{k+N}]^T$ at each time step can be recursively stacked up to the prediction horizon N:

$$\begin{aligned} Z_{k+1} &= \begin{bmatrix} C_d A_d^1 \\ \vdots \\ C_d A_d^N \end{bmatrix} x_k + \begin{bmatrix} C_d A_d^0 B_d & 0 & 0 \\ C_d A_d^1 B_d & \ddots & 0 \\ C_d A_d^{N-1} B_d & \cdots & C_d A_d^0 B_d \end{bmatrix} U_k \\ Z_{k+1} &= A_t x_k + B_t U_k. \end{aligned} \tag{A.12}$$

The original cost function minimizing tracking error $e_z(U) = Z(U) - Z_{Ref}$ and control input U:

$$J = (Z - Z_{ref})^T Q (Z - Z_{ref}) + U^T R U, \tag{A.13}$$

can be rewritten into a quadratic optimization formulation:

$$\min_U \quad \frac{1}{2} U^T H U + f^T U \tag{A.14}$$

$$s.t. \quad AU \leq B. \tag{A.15}$$

The column matrix $U \in \mathbb{R}^{N \times 1}$ contains all control actions over the prediction horizon N. The matrix $H \in \mathbb{R}^{N \times N}$ and $f \in \mathbb{R}^{N \times 1}$ are obtained by rewriting the cost function (A.13):

$$H = B_t^T B_t + \frac{R}{Q} \mathbb{1}_N, \quad f = B_t^T (A_t x_k - Z_{ref}), \tag{A.16}$$

with identity matrix $\mathbb{1}_N$ of prediction horizon size N.

The ZMP constraints for (A.15) is obtained from (A.12):

$$Z_{min} \leq Z_{k+1}(U_k) \leq Z_{max} \tag{A.17}$$

$$\begin{bmatrix} -B_t \\ B_t \end{bmatrix} U \leq \begin{bmatrix} A_t x_k - Z_{min} \\ -(A_t x_k - Z_{max}) \end{bmatrix}. \tag{A.18}$$

Furthermore, for real world applications, the rate of change for the COP $\dot{p}_k = \dot{x}_k - z_c \ddot{x}_k / g$ plays a crucial role. If the optimal controller produces an unrealizable COP trajectory, the system will not be able to track it. Hence, a constraint of the rate change of the COP according to the physical system is necessary as:

$$\Delta p_{min} \leq \dot{x}_k - \frac{z_c}{g} u_k \leq \Delta p_{max} \tag{A.19}$$

$$\begin{bmatrix} -\frac{z_c}{g} + B_t \\ \frac{z_c}{g} - B_t \end{bmatrix} U \leq \begin{bmatrix} \Delta p_{max} - A_t x_k \\ -(\Delta p_{min} - A_t x_k) \end{bmatrix}, \tag{A.20}$$

where the output $z_k = \dot{x}_k$ is the velocity of the COM. The maximal admissible values of the COP rate change are determined by the individual robot. For Valkyrie the maximal rate change is approximately $\pm 2 m/s$. Constraint (A.20) will be used for all CTM based QP formulations with input $u_k = \ddot{x}_k$. For LIPM based formulations, the rate change is the input $u_k = \dot{p}_k$ that can be directly constrained:

$$\Delta p_{min} \leq u_k \leq \Delta p_{max}. \tag{A.21}$$

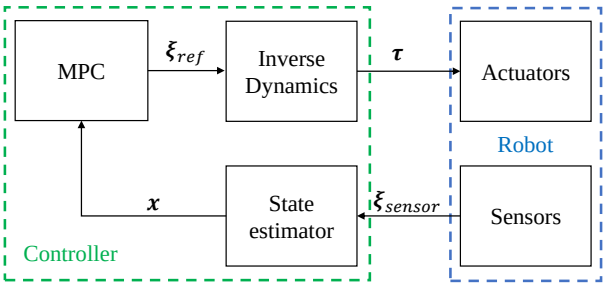


Figure A.4: Overview of the MPC framework.

MPC framework

The MPC framework (Fig. A.4) consists of three parts: MPC, Inverse Dynamics, and State Estimator. First, the sensor information ξ_{sensor} from the sensors is passed into a state estimator [210] outputting state x. Next, the Model-Predictive Controller solves the QP problem (A.12) for the optimal control input U using the state x obtained from the state estimator. The first element of the optimal control input $U(1) = u_k$ is used to forward simulate the dynamics (A.11) for the reference trajectory ξ_{ref}. Lastly, an Inverse Dynamics low-level controller maps the reference COM and COP trajectories from the MPC to joint torques τ which are then tracked on the actuators.

The torque commands τ, along with joint accelerations $\ddot{q}$ and ground reaction forces λ, are calculated as the solution $X = [\ddot{q}, \tau, \lambda]^T$ from a whole-body QP optimization problem (further details are described in the next Chapter, Section B.1):

$$\min_X \; X^T H X + f^T \tag{A.22}$$

$$s.t. \quad A_{eq} X + B_{eq} = 0 \tag{A.23}$$

$$A_{ineq} X + B_{ineq} \geq 0. \tag{A.24}$$

The cost function (A.22) is calculated as a weighted sum over Cartesian space acceleration tracking of COM and body links, such as swing foot and hands, COP tracking, and regularization of X. The equality constraints (A.23) are determined by the equations of motion:

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \end{bmatrix} + h(q, \dot{q}) = 0, \tag{A.25}$$

with inertia matrix $M(q)$, selection matrix S, stacked Jacobian matrices $J^T(q)$ of the contact links, and nonlinear effects $h(q, \dot{q})$. Torque limits, friction constraints, and COP constraints are considered in the inequality constraints (A.24).

A.4 Analysis of Numerical Stability Related with Prediction Horizon

Guaranteeing numerical stability is an important aspect for digital control, as numerical brittleness can induce oscillations and instabilities into systems, where no noise or uncertainties exist. In optimization, if

the matrices are ill-defined, nominal systems can diverge and unbounded responses can be evoked by small changes [211, 212]. Specifically, well defined matrices A_t, B_t (A.12) are important for the prediction matrix H in the cost function (A.14). Numerical issues caused by the long prediction horizon were avoided by a Singular Values Decomposition (SVD) of the optimization matrix in [211, 212]. Consequently, the constraints become softened and thus do not guarantee stability. In the following, the influence of the Prediction Horizon N on the numerical stability of different state space formulations will be analyzed, and the study will show that the numerical issues can be prevented by an appropriate choice of the dynamic model.

Numerical Stability

For determining the numerical stability and sensitivity the condition number $\kappa(A) = \|A\|\|A^{-1}\|$ of a matrix A will be used as an indicator. Using the Euclidean norm the calculation of the condition number becomes $\kappa(A) = \sigma_{max}(A)/\sigma_{min}(A)$, where $\sigma_{max}, \sigma_{min}$ are the maximal and minimal Singular Values (SV) respectively. As a general rule, condition numbers $\kappa(A)$ exceeding $1/\epsilon$, where ϵ is the number precision, can be considered to be an indicator for ill-defined matrices and therefore numerical brittle systems. The larger the condition number becomes, the more digit precision is lost during numerical operations, such as inverting, multiplying, etc. [213]. The condition number is exponentially influenced by the prediction horizon N. The larger N becomes, the more elements of A_d^N get exponentiated as in (A.12), resulting in large SV and therefore larger condition numbers. Due to this reason, it is desirable that the inter-multiplication of rows and columns in A_d are smaller than 1 to prevent the condition number to grow exponentially large. In the CTM formulation, values in the system matrix A and control matrix B are either 0 or 1. However, this is not the case for the LIPM formulation.

Fig. A.5 shows the influence of the prediction horizon N on the condition number of different optimization formulations. For the matrices A_d, B_d in equation (A.12) the 5 formulations in Table A.1 are considered. The matrix C_d is determined by whether COP or CP is tracked. Apart from formulation A, B, which due to its two-dimensional state space is only able to track CP alone, all other formulations are able to track both CP and COP. It can be seen that the Condition Numbers (CN) are identical for formulation A and B until the prediction horizon N exceeds 5s,where the CN exceeds $1/\epsilon$ and thus fluctuates due to numerical inaccuracies.

This phenomenon can be observed in all LIPM based state space representations: the function of CN over prediction horizon follows an exponential curve, but after the CN exceeds $1/\epsilon$, the CN fluctuates, while still increasing exponentially. An exception is the COP tracking formulation C: the matrices are well-conditioned for output matrix $C_d = [0, 0, 1]$ and state vector $x = [x, \dot{x}, p]^T$. The CTM formulation E, and the proposed formulation have well-defined matrices, and the CN only exceeds $1/\epsilon$ after more than a 100s.

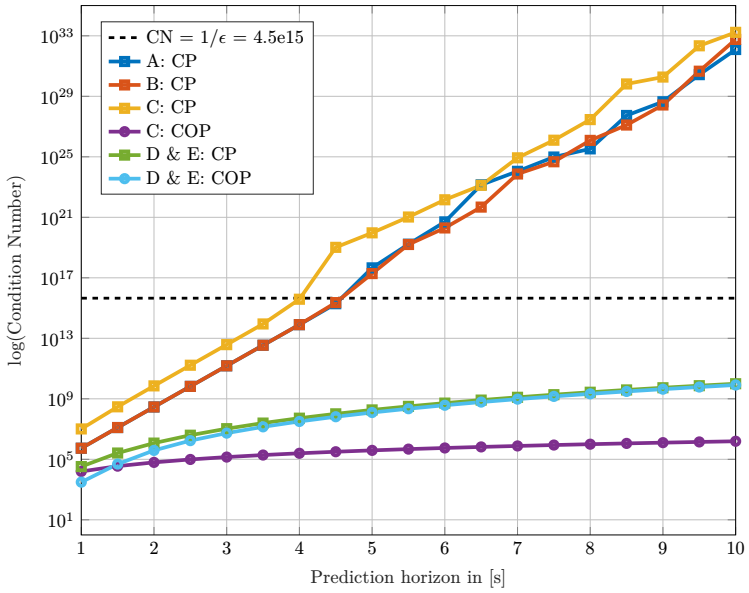


Figure A.5: Logarithmic representation of the condition number (CN) over different prediction horizons. Squared points are CN for CP tracking, circled points are CN for COP tracking. The CN for formulation E and proposed D overlap in both CP and COP.

Open Loop Stability for Planning

To further evaluate whether the state space formulations are suitable for planning, i.e. optimization over a long prediction horizon, the Eigenvalues (EV) were analysed. As in Table A.2, the prediction horizon does not influence the EV of the CTM based representations. For LIPM based representations however, the prediction horizon influences the EV after $NT = 5.5$s making these models unsuitable for long term planning as the open-loop case is highly unstable.

In Fig. A.6, the effect of a prediction horizon $T_{prev} = 5.5s$ for solution U of the planning problem (A.14) is shown exemplary (all LIPM formulations exhibit this behaviour) for CP tracking using formulation C and our proposed formulation D. The yellow line is the COP rate change trajectory that achieves perfect tracking given the CP reference.

It can be seen that the optimal solution using the LIPM (blue line) performs poorly, due to oscillation and deviations from the reference. This is caused by large prediction horizons, which can be observed by analysing the last row of the prediction matrix B_t in (A.12). It describes the contribution of each input $u_k, k = [1, ..., N]$ to the final output z_N. As shown in Fig. A.6, the control inputs of LIPM based formulation have almost no effect on the final state after $2s$. In contrast, our proposed formulation and formulation A (red line) are unaffected by the prediction horizon. All inputs u_k influence the final state with decreasing importance as k increases.

The comparative analysis shows that LIPM based models are not suitable for long term planning. If the loop is closed by the optimal solution of such models with large prediction horizons (empirically maximum tested: $4s$), the computed control input (COP) oscillates and ultimately diverges the system. This is caused, beside oscillations, by the ill-defined prediction matrices B_t result in inaccurately calculated feedback gains K, and therefore destabilize the system.

Summarized, all CTM based optimizations are suitable for long prediction horizons and therefore gait planning. The gait planner generates

Formulation		Prediction Horizon NT				
		1.5s	2.5s	3.5s	4.5s	5.5s
A: CP	CN	1e07	7e09	3e12	2e15	**2e19**
	EV	0.97	0.97	0.97	0.97	**68.74**
B: CP	CN	1e07	7e09	3e12	2e15	**2e19**
	EV	1.03	1.03	1.03	1.03	**8.81**
C: CP	CN	3e08	2e11	9e13	**1e19**	**1e21**
	EV	0.97	0.97	0.97	**1873.3**	**216.3**
C: COP	CN	4e04	1e05	2e05	3e05	5e05
	EV	**1.03**	**1.03**	**1.03**	**1.03**	**1.03**
D: CP	CN	3e05	4e06	2e07	1e08	3e08
	EV	1.00	1.00	1.00	1.00	1.00
D: COP	CN	5e04	2e06	1e07	7e07	2e08
	EV	1.00	1.00	1.00	1.00	1.00
E: CP	CN	3e05	4e06	2e07	1e08	3e08
	EV	0.97	0.97	0.97	0.97	0.97
E: COP	CN	5e04	2e06	1e07	7e07	2e08
	EV	0.97	0.97	0.97	0.97	0.97

Table A.2: Condition Number (CN) and Eigenvalue (EV) over different prediction horizons NT for the formulations presented in Section A.2, and COP or CP tracking.

a COM trajectory that follows the ZMP or the CP reference. Due to the well-defined prediction matrices A_t, B_t, the proposed formulation D and E can generate walking patterns to up to 100s. Given a ZMP or CP reference, an one-shot solution of the optimization can then be used as a nominal gait planning. The limitation of existing CTM based MPC schemes will be studied in the next section.

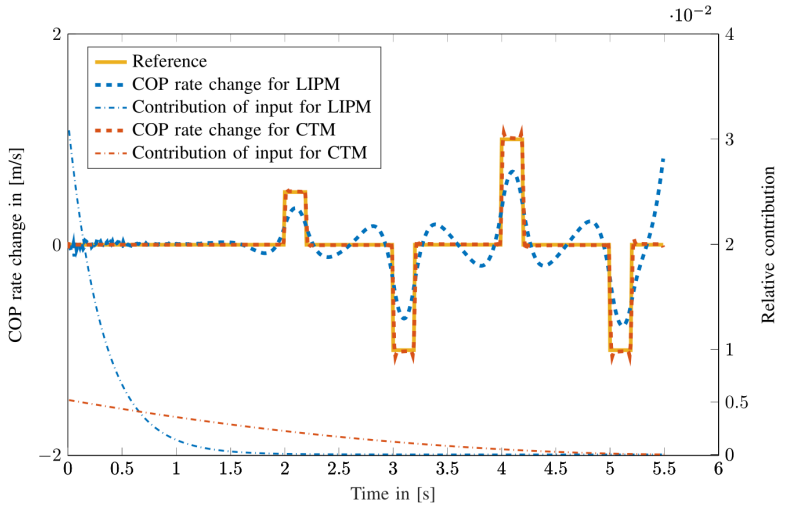


Figure A.6: COP rate change trajectories (left y-axis) for LIPM and CTM. The relative contribution (right y-axis) of input u_k to the final output z_N is in dashed lines for LIPM (blue) and CTM (red).

A.5 Simulation Benchmarks

This section presents the comparison study of closed loop behaviour for different formulations C, E, and our proposed formulation D in COP and CP tracking tasks while applying a variety of impulse and constant disturbances.

Simulation setup

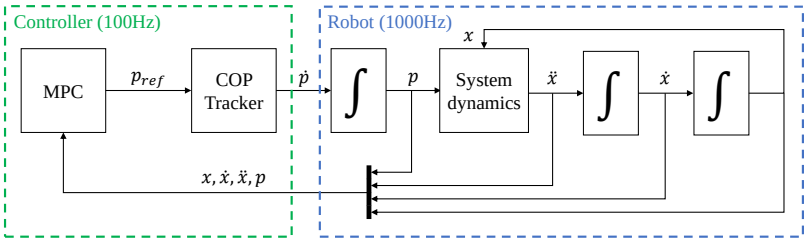


Figure A.7: Overview of the simulation setup in MATLAB.

For simulating the physics, two different platforms are chosen: Gazebo and MATLAB. The validity of the proposed MPC framework (Section A.3) for gait planning and feedback control is shown in the real world simulator Gazebo (Fig. A.2). The framework is able to handle unmodelled inclined terrain of up to 5° pitch and 10° roll inclination. The results can be seen on https://youtu.be/7uwtorW_kzE.

For benchmarking the disturbance rejection abilities a numerical simulation is conducted in MATLAB. The robot's dynamics (Fig. A.7) are numerically integrated at $T_{sim} = 1\text{ms}$, and the closed-loop MPC is running at a frequency of $f_{mpc} = 100Hz$. The COM dynamics follow the "COP→ acceleration" (A.1) causality determined by the LIPM (Fig. A.3):

$$\ddot{x}_{com} = \frac{g}{z_c}(x_{com} - p^*) + \ddot{x}_{ext}. \tag{A.26}$$

Formulation C expresses identical dynamics as formulation A and B, and can therefore be seen as a representative extension of the latter. The objective (A.13) of the optimization is tracking a predetermined ZMP or CP trajectory under constraints. For numerically sensitive systems,

correct selection of parameters is crucial: short prediction horizons or over-constrained inputs ($R < 10e^{-6}$) may result in instability. The prediction horizon was chosen as $N = 2.5s$, the weights as $Q = 1, R = 10e^{-6}$. The system is considered to be unstable, if the states diverge, the ZMP exceeds the SP (e.g. Fig. F.1a), or no solution of the optimization problem is found, i.e. the COP rate change constraint is significantly violated.

Disturbance Rejection

To determine the capability of disturbance rejection for the different formulations with different tracking objectives, four types of disturbances are studied: three impulses at different impact times, and one constant disturbance. All types of disturbances have relevance in the real application: impulse disturbances can be considered as pushes, or sudden impacts caused by collision, e.g. improper swing foot landing. Constant acceleration disturbances can be caused by an external load, biased COM position from the state estimation errors [210], or model uncertainties. The maximal rejectable disturbance of the individual models is detailed in Table A.3.

Alternatively for formulation E, instead of applying the erroneously calculated ZMP, it is possible to simply apply clipping of the output ZMP/COP when it exceeds the SP as the work in [214]. Even though clipping can prevent the system from getting unstable due to the ZMP exceeding the SP, the system can still become unstable due to divergent motion, or insufficient COP rate change. This approach is still inferior to our proposed formulation in theory, because the constraints are not correctly represented in the formulation and hence the clipped control effort is inevitably suboptimal, which leads to roughly 50% less disturbance rejection ability compared to that of our proposed formulation.

Table A.3: Maximal rejectable disturbance for CP or COP tracking. The ratio in the brackets is normalized by the maximal disturbance (bold) for the specific disturbance type and formulation.

	Disturbance duration in seconds			
Formulation	0–10	2.4–2.45	2.6–2.65	2.8–2.85
C (CP)	0.36 (0.31)	2.04 (0.91)	2.57 (0.92)	2.02 (0.80)
C (COP)	0.18 (0.15)	**2.23 (1.00)**	**2.78 (1.00)**	2.23 (0.89)
D (CP)	0.50 (0.43)	2.04 (0.91)	2.57 (0.92)	2.29 (0.91)
D (COP)	**1.17 (1.00)**	2.16 (0.97)	2.76 (0.99)	**2.51 (1.00)**
E (CP)	0.42 (0.36)	0.30 (0.13)	0.37 (0.13)	0.34 (0.14)
E (COP)	0.56 (0.48)	0.35 (0.16)	0.42 (0.15)	0.44 (0.18)
E clip. (CP)	0.35 (0.30)	1.42 (0.64)	1.40 (0.50)	1.40 (0.56)
E clip. (COP)	0.56 (0.48)	1.39 (0.62)	1.39 (0.50)	1.37 (0.55)

Performance Analysis

Comparison of Models

Under constant disturbance our proposed formulation performs twice as good as the others (Table A.3). Our proposed MPC withstands more than double the amount of disturbance compared to the other formulations in COP tracking tasks, and withstand averagely 34% more perturbation in CP tracking tasks. Compared to other formulations, the disturbance is handled better, and therefore the COP rate change only hits the constraints at much larger perturbations. Formulation C fails due to the violation of the COP rate change constraints, and the ZMP in formulation E violates the SP constraints earlier (Fig.F.3).

For constant disturbances, the proposed formulation D is able to distinguish between disturbance-generated and self-generated accelerations and therefore correctly calculating the COP that respects the physical constraints. By doing so, it is able to counterbalance the constant disturbance, track the COP reference with zero steady state error after 1 second by offsetting the COM away from the nominal trajectory and leaning against the constant push (cf. behaviour after 7s). In contrast, formulation E generates the COP away from the desired reference with steady state error while keeping the COM at the nominal trajectory yielding.

Formulation C shifts both COP and COM away from the nominal trajectory, because only the input $\dot{p}$ is used to track the COP reference without considering the COM state. However, convergence is only guaranteed by including the COM states in the cost function or constraints. For short disturbances, e.g. impulses, deviations between predicted COM state and real disturbed COM state do not influence the performance of the controller. But for long, constant deviations, ignoring the COM state leads to decreased performance. This can be seen in Fig. F.3(b). It can be further elaborated by turning the constant disturbance off, and observing how the steady state error vanishes, as predicted COM state and real COM state coincide again (cf. attached video).

For impulse disturbances (Fig. F.1), the time of impact is almost irrelevant with respect to the ratio (see Table A.3). Although satisfying the internal model of the ZMP constraint, formulation E becomes unstable because of the violation of the SP constraint due to an erroneously calculated ZMP (cf. Section A.2). Both the proposed formulation D and C can handle the ZMP dynamics correctly with almost the same performances that are better than formulation E.

In summary, the disturbance rejection capability is not fully maximized in the CTM based MPC formulation E. LIPM based MPC formulations perform similar to the proposed formulation for impulse disturbances, but reject less performance during constant disturbances.

Comparison of Control Objectives

Although our former study shows that the COP tracking has good performance against disturbances, decomposing the COM's motion into

a stable and unstable parts and controlling the unstable part of divergent component or capture point is known for additional advantages [215].

Setting the control objective of minimizing CP, i.e. CP tracking, makes the system fully utilizing the stability margin while being disturbed by quickly moving the COP to the boundaries as possible. In contrast, an objective of minimizing COP error, i.e., COP tracking, limits the deviation of COP away from the reference, thus a slower COP response in the presence of disturbance, so a problem may arise if a second disturbance occurs.

The CP tracking recovers faster and can therefore possibly handle a new impulse, whereas the COP tracking recovers slower being vulnerable to a new push. The response of an impulsive push using the proposed formulation can be seen Fig. F.2. After getting disturbed for 50ms at $t = 2.20$s, the CP tracking has COP settles back at $t = 2.5$s, while for the COP tracking the COP is still at 30% of its stability margin. Besides, the COP tracking scheme reinforces smaller COP errors due to its objective and thus demands larger COP rate change; while the CP scheme allows small harmless variations of COP as long as CP is tracked and COP is feasible, thus requires smaller COP rate change.

A.6 Conclusion

This chapter presented a formulation for Model-Predictive Control based on the Linear Inverted Pendulum, which has an improved performance over previous formulations and is suitable for both gait planning and feedback control of legged locomotion. It resolves two previously unattended problems: (1) numerical stability in the case of long prediction horizons, making the proposed method more suitable for gait planning; (2) enhanced disturbance rejection attributed to the incorporation of COP as a new state variable, and therefore correctly including the COP in the feedback loop. Our study showed that the proposed formulation was able to reject larger disturbances than the previous LIP formulations. Furthermore, in combination with a whole-body controller, the proposed MPC framework was able to blindly traverse unknown inclined terrain of up to 5° and 10° for pitch and roll inclination respectively.

Joint Control | B

The goal of low-level joint control is to calculate joint torques that realise the provided commands as close as possible while maintaining stability. In this work, the commands can come from an MPC controller as described in Appendix A, or from actions provided by a Neural Network policy.

This chapter presents two approaches to achieve low-level joint control: Inverse Dynamics based whole-body control, and joint impedance control. While whole-body control guarantees stability, their reliance on the model and constraints make whole-body control only suitable in well modelled environments. In contrast, low-level joint controls do not have inherent stability guarantees and rely heavily on well-designed, stabilising reference trajectories, e.g., through MPC or DRL.

Whole-Body Quadratic Programming or Joint Impedance Control for low-level joint control.

B.1 Whole-Body Torque Control

Whole-Body Quadratic Programming optimises over joint acceleration, torque, and contact forces.

The whole-body controller is formulated as Quadratic Programming (QP) problem due to the existence of fast solvers exploiting the convexity, sparsity, and linearity of constraints of QP problems. A given cost function under equality and inequality constraints is optimised over a state vector $X = [\ddot{q}, \tau, \lambda]^T$ for a robot with N Degree of Freedom (DOF) as in [204]. With the joint accelerations $\ddot{q} \in \mathbb{R}^N$, joint torques $\tau \in \mathbb{R}^N$, and contact forces $\lambda \in \mathbb{R}^{6K}$ for K end-effectors the QP problem is formulated as:

$$\min_x \frac{1}{2} X^T H X + f^T X \tag{B.1}$$

$$s.t. \quad A_{ineq} X + b_{ineq} \leq 0, \tag{B.2}$$

The construction of the matrices H, f, A, b will elaborated in the following. Any tracking task can be formulated as $1/2\|y - y_{ref}\|^2 = 1/2\|AX - b\|^2$, which yields the matrices $H = A^T A$, $f = -A^T b$ for the cost function of the QP problem. Multiple tasks are stacked in A, b and weighed by w_i, where $i = 0, ..., n$ is the i-th task:

$$A = \begin{bmatrix} w_0 A_0 \\ w_1 A_1 \\ \vdots \\ w_n A_n \end{bmatrix}, b = \begin{bmatrix} w_0 b_0 \\ w_1 b_1 \\ \vdots \\ w_n b_n \end{bmatrix} \tag{B.3}$$

Control Objectives

Tasks are defined in the form $1/2\|A_i x - b_i\|^2_{w_i}$ and stacked in (B.3). In the following tasks, which will be implemented in order to track trajectories given by the higher levels, will be introduced.

Tracking in Cartesian Space:

For tracking links (hand, feet, COM) in Cartesian space, a PD law is used to generate appropriate accelerations that will then be tracked by the QP. For a given desired Cartesian space trajectory x_d with desired velocity $\dot{x}_d$ and acceleration $\ddot{x}_d$, the PD law using the gains K_P, K_D is:

$$\ddot{x}_d^* = K_p(x_d - x) + K_d(\dot{x}_d - \dot{x}) + \ddot{x}_d, \tag{B.4}$$

with real (measured) states $x, \dot{x}$. The resulting acceleration $\ddot{x}^*$ following the PD law will be used as reference acceleration which the QP will track. The relation between velocity in task and joint space is established by the Jacobian $J(q)$. Its time derivative yields:

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q, \dot{q})\dot{q}, \tag{B.5}$$

which can then be rewritten into the form $\|A_i X - b_i\|$:

$$\begin{aligned} A_i &= \begin{bmatrix} J(q) & 0 & 0 & 0 \end{bmatrix} \\ b_i &= \ddot{x}_d^* - \dot{J}(q, \dot{q})\dot{q} \end{aligned} \tag{B.6}$$

Cartesian space tracking is achieved by using (B.6) with the appropriate Jacobian matrix $J(q)$, which relates the link velocity to the respective joint angle velocities.

Centre of Pressure Tracking:

The reference COP tracking is conducted in the local foot frame. Therefore the mid-level COP trajectory, which are generated in a world-frame, need to be transformed into the local foot frame by a homogeneous transformation and the COP is calculated by

$$p = \begin{bmatrix} -\tau_y / f_z \\ \tau_x / f_z \end{bmatrix}, \tag{B.7}$$

where τ_x, τ_y, f_z are the torque and force components of the contact forces of the feet. COP tracking is achieved by the following task matrices:

$$A_i = \begin{bmatrix} 0 & 0 & \begin{bmatrix} 0 & 0 & p_{x,L}^* & 0 & 1 & 0 \\ 0 & 0 & p_{y,L}^* & -1 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & p_{x,R}^* & 0 & 1 & 0 \\ 0 & 0 & p_{y,R}^* & -1 & 0 & 0 \end{bmatrix} & * & * \end{bmatrix} \tag{B.8}$$

$$b_i = 0, \tag{B.9}$$

where $*$ indicates that in the case of usage of hands, the COP tracking may be calculated the same way as for the feet contacts.

State tracking and Regularization:

Additional to the aforementioned tracking tasks, the optimisation variable X can directly track a desired state $X_d^* = [\ddot{q}_d^*, \tau_d^*, F_d^*]$ via:

$$A_i = \mathbb{I} \tag{B.10}$$

$$b_i = \begin{bmatrix} \ddot{q}_d^* & \tau_d^* & F_d^* \end{bmatrix}^T. \tag{B.11}$$

If no desired value is needed, b_i can be chosen to be zero. In this case, this task will have a regularizing effect, providing low values of the state components.

Weight Distribution:

A desired weight distribution $w^* = F_{zl}/(F_{zl} + F_{zr})$ can also be achieved by:

$$A_i = \begin{bmatrix} 0 & 0 & S \end{bmatrix} \tag{B.12}$$

$$b_i = 0, \tag{B.13}$$

where S is a row vector of zeros but for the F_z components, which has the value $1 - w^*$ and $-w^*$ in the left and right foot z component respectively.

Solving the QP problem (B.1) yields a state vector of joint accelerations, joint torques and the resulting contact forces, which are coherent according to (B.14). The joint torques will be applied on the joints for tracking of the desired high-level trajectories.

Physical Constraints

For whole-body control physical coherence between the states needs to be ensured. This is achieved by using the whole-body dynamic motion equations of the robot:

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ F \end{bmatrix} + h(q, \dot{q}) = 0, \tag{B.14}$$

where $M(q)$ is the inertia matrix, $h(q, \dot{q})$ are non-linear forces (gravitational, centrifugal, coriolis), S is a selection matrix for the torques τ, and $J^T(q)$ is the transpose of the Jacobian matrix for the respective contact link.

In addition to physical correctness of the solution, locomotion specific inequality constraints are used to guarantee, that the robot will not fall over. These are constraint imposed on the feet: friction and COP constraints. In addition to the commonly used constraints an additional constraint is imposed on the z component of the torque to prevent the well-known problem of slippage in the yaw-direction [216]. Every contact force F_i, with i $= L, R$ either left or right foot, needs to satisfy

the following inequality constraints for non-slippage and balance of the robot:

$$|f_x| \leq \mu f_z \tag{B.15}$$
$$|f_y| \leq \mu f_z \tag{B.16}$$
$$f_z > 0 \tag{B.17}$$
$$|\tau_x| \leq \mathcal{Y} f_z \tag{B.18}$$
$$|\tau_y| \leq \mathcal{X} f_z \tag{B.19}$$
$$\tau_{min} \leq \tau_z \leq \tau_{max}, \tag{B.20}$$

with

$$\tau_{min} = -\mu(X+Y)f_z + |\mathcal{Y} f_x - \mu\tau_x| + |\mathcal{X} f_y - \mu\tau_y|$$
$$\tau_{max} = +\mu(X+Y)f_z - |\mathcal{Y} f_x + \mu\tau_x| - |\mathcal{X} f_y + \mu\tau_y|.$$

Slippage in x, y direction is prevented by constraining the force in the respective direction (B.15), (B.16) for a given friction coefficient μ. Unilateral forces (no suction of the feet to the ground) are guaranteed by (B.17). The COP is constraint to be within a bounding box, defined by the dimensions $\mathcal{X}, \mathcal{Y}$ in (B.18), (B.19) and yaw slippage is prevented by (B.20).

Tuning Parameters for Joint Control

The accuracy of the individual task is decided by the respective weights. The larger the weights, the higher the precision of respective tasks becomes. Rather than predetermining the priority of the tasks, which may change for different circumstances (locomotion tasks are less important during manipulation, and vice versa) and finding solutions in multiple null-spaces [217], weighing the importance of the individual task in a non-hierarchical fashion will lead to more flexibility.

The choice of weights can be conducted either by manual design or by learning. Manual design implies the usage of hand-tuned parameters for different tasks, and choosing the appropriate set of parameters while interpolating between the weights during the transition phase. Alternatively, instead of tuning the parameters by hand an optimisation problem can be formulated that minimizes the error for the given task. Due to the fact, that interaction with the environment cannot be modelled and therefore the cost function cannot be analytically determined, the cost function is treated as black box. The cost for a given parameter set is set by the environment and black box optimisation methods, such as Bayesian Optimisation [43], can be used to minimize the cost function for a given task.

In Chapter E, a high-dimensional Bayesian Optimisation approach is presented that automatically tunes the parameters for the whole-body controller. The parameter set includes both objective weights w_i (B.3) and PD parameters K_p, K_d (B.4).

B.2 Joint Impedance Control

Joint impedance control describes the control of every individual DoF joint, and decouples the influences of various joints. The typical control objective is on joint position, but can also be on joint velocity or joint acceleration. In order to realise the joint state, a joint torque is required to actuate the joint. The output of the joint controller is thus always torque. Next, a joint torque loop that transforms the joint torques into currents that ultimately actuates the joint.

See Chapter 4 for how Joint Impedance Control is used with Deep Reinforcement Learning to achieve dynamic motions on robots.

Due to the ease of implementation as well as their ability to track the reference signal well, a joint Proportional-Derivative (PD) Controller is implemented:

$$\tau = K_p(q_{ref} - q) - K_D\dot{q}, \tag{B.21}$$

with joint torque τ, reference joint position q_{ref}, joint position q, and joint velocity $\dot{q}$. The PD controller modulates both stiffness and damping through tuning the proportional and derivative gain respectively leading to an impedance behaviour on the joints based on the magnitude of the gains.

Accelerated Deep Reinforcement Learning

C

This chapter provides an overview of DRL that is used throughout this work. Further, this chapter shows how parallelisation can be leveraged to accelerate the training and development time. For learning methods in Robotics, such as Deep Reinforcement Learning, the most time-consuming component is the collection of data and samples. In the following, we propose a fast sample generation procedure through parallelisation of environments leveraging the multi-threaded architecture of modern computer hardware and utilising the parallel nature of Graphics Processing Units (GPUs).

We show how these capabilities can be used in conjunction with state of the art on- and off-policy Deep Reinforcement Learning algorithms to solve a variety of Robotics problems. In particular, we conduct benchmarks on MuJoCo environments achieving up to 1200 faster than real time solving the benchmarks within minutes. Furthermore, we train a trotting and balancing policy for the quadruped ANYmal and biped Valkyrie respectively, and deploy the policies on the real robots.

Through our proposed parallelisation framework, we achieve simulation speed 200 times faster than real time, and can successfully train and deploy the trotting and balancing policies on the real robots within 1 hour.

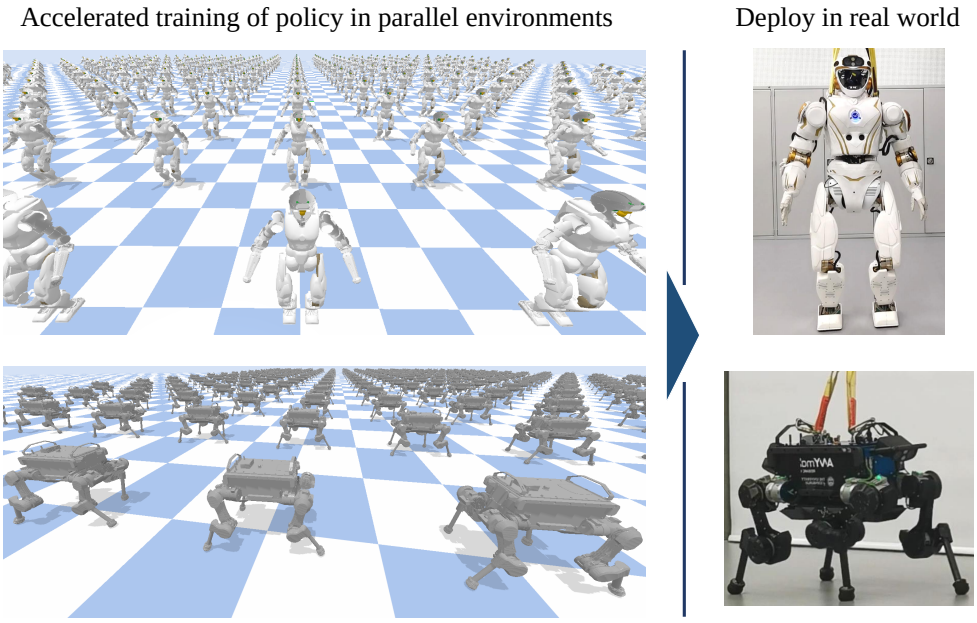


Figure C.1: Accelerated training of policies through simulation in parallel environments. Our proposed parallelisation and training procedure learns humanoid balancing and quadruped trotting within 1 hour real time and can be directly transferred on the real robot.

C.1 Introduction

The success of Deep Learning and subsequent revolution in the fields of Computer Vision, Natural Language Processing and Data Science can be mainly attributed to Deep Neural Networks (DNN), which are powerful function approximators with high expressiveness due to their large amount of parameters [218]. But with increasing complexity and number of neuron connections, a larger amount of data and computational power has become necessary to successfully train them. Despite success in the fields of supervised learning arising from DNN's and the availability of large amounts of data to train these, Robotics has not yet reached the same stage: generating and collecting large amounts of data require either substantial computational expense in simulation or unrealistic collection times prone to potential damage on real hardware.

Realistic physics simulators enable sim2real transfer.

With increasingly realistic and performant physics simulators [219, 220] as well as methods to bridge the simulation to reality gap (sim2real) [66, 65], using simulated data for training in Robotics has become a valid alternative to real world samples. Gathering simulated data to train DNN through Deep Reinforcement Learning (DRL) and applying sim2real methods has allowed policies to be deployed in reality to achieve inter alia Fall Recovery [66], and animal-like behaviours [65].

This work presents an approach of collecting robotic data at large by leveraging distributed architectures. In particular, the data collection process utilises the rapid improvement of computation power and software capable of exploiting the parallel nature of GPU's to perform traditionally expensive computations in parallel. For the high-degrees of freedom ANYmal robot, our presented approach collects data 200 times faster than real time on a consumer-grade computer (Intel i7 8700K CPU, one Nvidia Geforce RTX 2080Ti) allowing the training of robot policies and partly offsetting the data hungry nature of the current DRL algorithms. We intend to open-source our code after publication of this paper.

We will show how the parallel sample collection procedure can be used together with DRL algorithms for both on- and off-policy methods to train the quadruped ANYmal [221] to trot and the humanoid robot Valkyrie [10] to maintain balance. Despite the continuous, high-dimensional nature of the robots, our fast sample collection procedure in a realistic physics simulator enables training of the policies within 10 minutes real time and is robust enough to be directly deployed on the real robots ANYmal and Valkyrie (Fig. C.1). Lastly, we show that the MuJoCo benchmark tasks can be solved within minutes by our proposed approach, which is an order of magnitude faster than traditional approaches [222].

Our summarised contributions are:

- Proposing and discussing technical steps to allow parallel sample collection of robotic data for Machine Learning approaches achieving a real time speedup by a factor of several hundred on a consumer-grade computer.

- Presenting a learning framework that combines parallel sample collection with Deep Reinforcement Learning for both on- and off-policy methods.
- Training a trotting and balancing policy within 1 hour for ANYmal and Valkyrie respectively and deployment of the policies on the real robot platforms.
- Solving MuJoCo benchmark tests within minutes using our proposed approach.

In the following, we first discuss related work in Section C.2. Next, we present our parallelisation structure in Section C.3, and discuss key design choices for the successful implementation. In Section C.4, we outline how on- and off-policy DRL algorithms can be used with our proposed parallelisation framework and discuss their advantages and disadvantages. The environments which will be parallelised and trained in are described in Section C.5. Results including training on the MuJoCo benchmarks and the real world deployment of the policies on ANYmal and Valkyrie are presented in Section C.6. Lastly, we draw a conclusion in Section C.7.

C.2 Related Work

Recent algorithmic advances in Reinforcement Learning [18] incorporated Deep Neural Networks (DNN) as function approximators. Consequently, continuous control tasks as prevalent in robotics can be solved by directly optimising the actor through policy gradient methods [58].

DRL algorithms can be categorised into two families: on-policy and off-policy algorithms. While on-policy algorithms evaluate and improve the same policy from which the sampled data originates from, off-policy algorithms do not require the samples to originate from the same policy as the trained, target policy. Current state of the art on-policy algorithms for continuous control include Trust Region Policy Optimization [59], Proximal Policy Optimization [60], and Asynchronous Advantage Actor-Critic [61]. For off-policy algorithms, Soft-Actor Critic [62], Deep Deterministic Policy Gradient (DDPG) [63], and Twin Delayed DDPG [64] have seen successes in controlling robotic systems.

These algorithms have been applied in robotics to imitate animal locomotion [65], achieve fall recovery [66], yield human-like locomotion [67], perform grasping in cluttered environments [68], perform dexterous manipulation [69], and achieve remarkable results in robotic benchmarks [70, 71].

Beside advancements in DRL, the success in continuous control tasks can be attributed to the emergence of realistic and performant physics engines. The most commonly used physics engines for simulating DRL agents are MuJoCo [223] and Bullet3 [224]. Further frequently used physics engines include ODE in conjunction with Gazebo [179], and Physx [182]. Various benchmarks for the physics engines exists [219, 220].

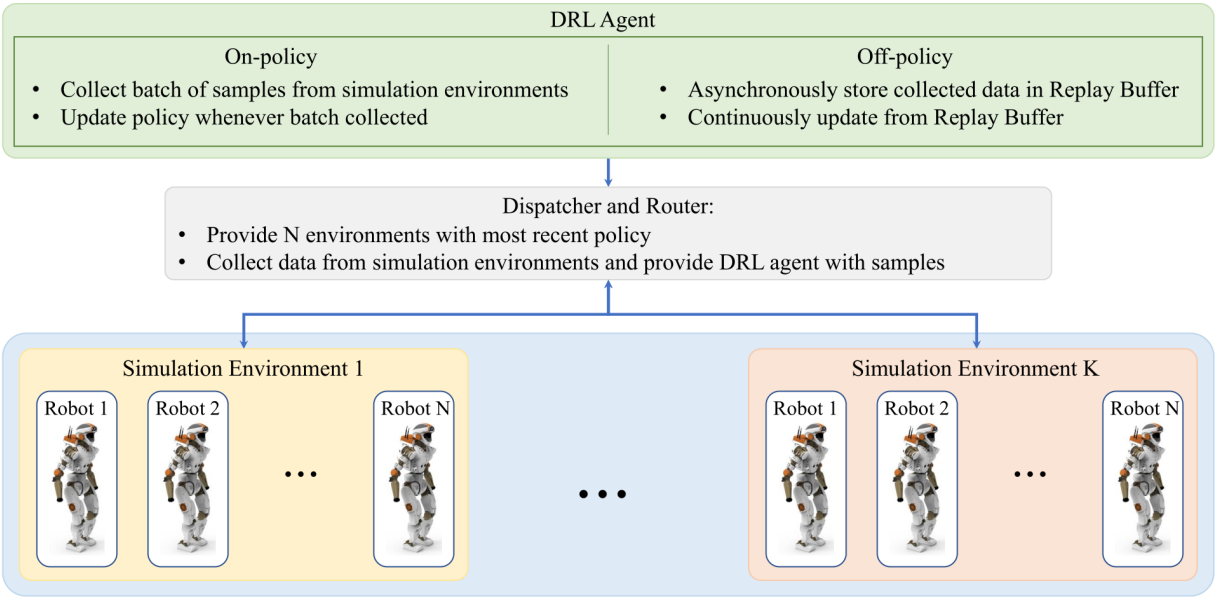


Figure C.2: Proposed distributed simulation and learning architecture.

C.3 Parallelisation Structure

To speed up the training of complex agent, we wish to generate policy samples consisting of state s_t, action a_t, reward r_t, and next state s_{t+1} as fast as possible while following the most up-to-date policy. However, despite being widespread, the most used physic engines in Robotics such as Bullet3, ODE, or MoJuCo are implemented as a single thread only able to use one core of the computer's CPU.

This section outlines how to leverage the technological progress achieved within the gaming industry in order to fully exploit the multi-threaded architecture of modern computer hardware. The physics engine PhysX developed by NVIDIA is optimized to take advantage of the computing capabilities of both CPUs and GPUs. Large matrix operations arising during the simulation of big dynamical structures and collision resolution can typically be handled by the specialized parallel hardware of the GPUs.

Architecture Considerations and Challenges

While migrating from a sequential to an asynchronous design, several factors in the software implementation have to be considered:

(1.) The data transfer between the main memory, the CPU and the GPU must be minimized.
(2.) The GPU computing load shared between simulation and (neural network) policy evaluation has to be balanced.
(3.) When multiple policy evaluations run in parallel, one possible long sequence (e.g., where others are subject to early terminations) should not stall the whole pipeline.
(4.) The policy can not be updated after each simulation sequence and has to be learned asynchronously.

(5.) The optimal trade-off between the number of independent simulation environments and the number of robots per environment has to be found in order to address (2.).
(6.) To prevent the quadratic nature of the collision checking, the robot instances simulated within the same environment have to be logically isolated. If this can be guaranteed, the performance of increasing the number of robot per environment can scale linearly.

In this work, we address these challenges as follow. To solve (1.), a single simulation environment is configured to run several robot instances at the same time. Batch operations reduce the need for individual configuration and alleviate the GPU/CPU communication. The updates of the policies have to be conducted asynchronously, and ideally on a separate GPU to tackle (2.). Issue (3.) is ensured through appropriate threading mechanisms. Regarding (4.), for off-policy algorithms, the policy is being updated asynchronously (DRL agent box in Fig. C.2), while the simulation environments are sampling data with the most recent policy provided by the Dispatcher and Router. Challenge (5.) is solved through a grid-search to find the optimal trade-off (c.f., Fig. C.4). Logic isolation is conducted both on a software side and through ensuring that the robots have a distance towards each over to prevent collision between robots in the same Simulation Environment.

Distributed Simulation Architecture

Following the architecture considerations and challenges, we proposed the distributed architecture depicted in Fig. C.2. Each simulation environment is independently running the PhysX engine on GPU. In these environments, up to 1000 robots or 64000 body links are simulated at the same time while being placed in the world and logical isolated such that any interaction between them is avoided.

In every simulation environment, every robot evaluates the same version of the neural network policy provided by the Dispatcher and Router. The generated training samples are collected and temporarily stored in memory. When the required amount of samples has been gathered, the samples are routed to the DRL agent and the newest up-to-date policy is retrieved by the environment. By being asynchronous, multiple simulation environments can be deployed in parallel on several GPU's and machines scaling up the amount of sample generated concurrently.

For off-policy algorithms using a replay buffer, the replay buffer gets asynchronously filled with new training samples, while the DRL algorithm continuously updates the policy. To adapt to this distributed scheme, we exemplarily show in Section 11 how the classic off policy SAC can been reformulated to support both batches and asynchronous updates of the replay buffer.

Technical Details

The communication between DRL agent, Dispatcher and Router, and individual simulation environments is implemented by the gRPC (gRPC Remote Procedure Calls) framework [225]. By relying on the HTTP

(TCP) network protocol, multiple components and workers can be deployed on different servers enabling the scalability of the proposed architecture.

Improved stability during collisions and penetrations is enabled through the Temporal Gauss-Seidel (TGS) in PhysX instead of a Projected Gauss-Seidel (PGS) solver method. To accommodate the most common Python and machine learning API's used in OpenAI gym [226] and off-the-shelf DRL algorithms, the PyBullet API [178] is used as frontend using Physx as backend physics engine.

C.4 Deep Reinforcement Learning

The collected data is used for training task-solving policies via Deep Reinforcement Learning. In this work, we present how the collected data can be used on two families of Reinforcement Learning algorithms: on-policy and off-policy algorithms. While on-policy algorithms use the same policy for both sample collection and training, off-policy algorithms use a behaviour policy to collect the samples while training a different target policy for task completion. The type of learning algorithm thus changes the nature of how the collected samples can be used during training.

In the following, we outline Proximal Policy Optimization (PPO) and Soft-Actor Critic (SAC) as representative example for on-policy and off-policy algorithms respectively, and discuss how fast sample collection can be used to accelerate the training of these policies.

On-policy algorithms

PPO is a policy-gradient method with training stability properties. Training stability is achieved by avoiding large parameter updates between the old and new policy $\pi_\theta(a_t|s_t)$ parametrised by θ_{old} and θ respectively. This is achieved by clipping the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ if it exceeds a pre-defined interval $[1-\epsilon, 1+\epsilon]$ with hyperparameter ϵ.

The policy's parameters are updated via stochastic gradient descent by minimising the following loss function J_{PPO}:

$$J_{\text{PPO}}(\theta) = \mathbb{E}[\min(r(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{C.1}$$

with estimated advantage function $\hat{A}_t$ at timestep t.

A policy π_θ is trained through algorithm 4. First, N actors are used to collect batches of samples (line 2 in algorithm 4) with every actor collecting T timesteps. Then, the policy is updated by minimising (C.1) through Adam [227] for K epochs using minibatches sampled from the collected batches of size NT. The advantage estimate calculated in line 4 of algorithm 4 is obtained through Generalized Advantage Estimation (GAE) [228], and requires rolling out a policy for T timesteps.

For methods that use advantage estimates, or any kind of temporal-difference learning $TD(\lambda)$ [168], the policy needs to roll out at least

Algorithm 4: Pseudocode for PPO

```
for iter=1,2,... do
    for actor=1,2,...,N do
        Rollout policy π_{θ_old} for T timesteps;
        Compute advantage estimates Â_t, t = 1, ..., T;
    Calculate θ as argmin J_PPO(θ) using batch of size NT.;
    θ_old ← θ;
```

T timesteps. For these methods, the computational bottleneck lies in the speed at which T rollouts can be conducted. While the rollouts of N actors can be parallelised, every actor needs to collect T timesteps to compute the advantage in sequence. Thus, the minimum amount of time required for a batch of size NT is limited by the length T of one rollout.

Off-policy algorithms

SAC aims to optimise a maximum entropy objective $J_{\mathrm{SAC}(\pi)}$:

$$J_{\mathrm{SAC}}(\pi) = \sum_{t=0}^{T} \mathbb{E}[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))], \tag{C.2}$$

with reward r for a given state s_t and action a_t at time t, temperature parameter α and policy entropy $\mathcal{H}(\pi)$. Using the parametrisation trick, the parameters θ for policy π_θ are obtained by minimising $J_\pi(\theta)$:

$$J_\pi(\theta) = \mathbb{E}[\log \pi_\theta(a_t|s_t) - Q_\phi(s_t, a_t)]. \tag{C.3}$$

The action value function $Q_\phi(s_t, a_t)$ is obtained by minimising the bellman residual $J_Q(\phi)$:

$$J_Q(\phi) = \mathbb{E}[\tfrac{1}{2}(Q_\phi(s_t, a_t) - \hat{Q}(s_t, a_t))^2], \tag{C.4}$$

with bellman equation $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}[V_\psi(s_{t+1})]$ and discount factor γ. An estimation of value function V_ψ is obtained through minimising $J_V(\psi)$:

$$J_V(\psi) = \mathbb{E}[\tfrac{1}{2}(V_\psi(s_t) - \mathbb{E}[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t|s_t)])^2]. \tag{C.5}$$

Lastly, the temperature parameter α can be automatically adjusted by optimising over the objective:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t}[-\alpha \log \pi_t(a_t|s_t) - \alpha \mathcal{H}_0], \tag{C.6}$$

with predefined minimum policy entropy threshold $\mathcal{H}_0$. It was shown in [62] that this object maximises the expected return while satisfying a minimum entropy constraint $\mathcal{H}(\pi_t) \geq \mathcal{H}_0$.

The training process of policy π_θ is outline in algorithm 5

Contrary to on-policy algorithms (Section C.4) which need to obtain samples originated from rolling out the current policy, the samples for

Algorithm 5: Pseudocode for SAC

```
1  D ← ∅ , initialise replay buffer;
2  for iter=1,2,... do
3      while collected samples < batch size do
4          Sample and perform action a_t ∼ π_θ;
5          Collect d = {s_t, a_t, r(s_t, a_t), s_{t+1}};
6          Store sample d in replay buffer D ← D ∪ d;
7      for policy update=1,...,K do
8          Sample batch from replay buffer D for update;
9          Update policy π_θ via (C.3);
10         Update action value function Q_φ via (C.4);
11         Update value function V_ψ via (C.5);
```

off-policy updates are obtained from a replay buffer. Thus, the samples stored in the replay buffer can be collected by rolling out a behaviour policy that is different from the target policy enabling parallel sample collection.

The batch sample collection (line 3 in algorithm 3) can thus be independently conducted until the amount of collected samples equates the required batch size. Consequently, the sample collection time for off-policy algorithms is not bottle-necked by a minimum amount of timesteps T as for on-policy algorithms, and a vast amount of samples can be collected in parallel for the replay buffer. Furthermore, the same samples can be used multiple times in the replay buffer which typically leads to more sample-efficiency for off-policy algorithms.

Although off-policy algorithms are not bottle-necked by the collection speed, the convergence time is impeded by the speed at which K policy updates can be conducted. Typically the amount of policy updates is of the same magnitude as the amount of samples collected, i.e., every time a sample is collected, the policy is also updated.

Asynchronous Updates

We adapt the standard SAC (algorithm 5) to utilise our proposed parallel structure (algorithm 6). In contrast to classical SAC, the sample collection is independent from the policy update. The sample collection and filling of the replay buffer is continuously conducted with the most recent policy, while the policy is being updated in parallel.

Algorithm 6: Pseudocode for Asynchronous SAC

$\mathcal{D} \leftarrow \emptyset$, initialise replay buffer;
Run sample collection and policy update in parallel threads;
while policy not converged **do**
    Retrieve latest policy π_θ;
    Sample and perform action $a_t \sim \pi_\theta$;
    Collect $d = \{s_t, a_t, r(s_t, a_t), s_{t+1}\}$;
    Store sample d in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup d$;
while policy not converged **do**
    Sample batch from replay buffer $\mathcal{D}$ for update;
    Update policy π_θ via (C.3);
    Update action value function Q_ϕ via (C.4);
    Update value function V_ψ via (C.5);

C.5 Environments

In the following, we present the environments in which the DRL agents are trained. The presented environments are single instances and are parallelised as outlined in Section C.3. In this work, we improved the sample collection time for high-degrees of freedom robots, namely the quadruped ANYmal [221] and the humanoid Valkyrie [10], and for several MuJoCo robots, namely Ant, HalfCheetah, Hopper, and Walker.

High-degrees of Freedom Robots

The Valkyrie robot (Fig. C.1 top left) is a 23 Degrees of Freedom (DoF) robot. We implement the training procedure as outlined in [171]. The state and action space, reward are identical as in [171].

The ANYmal quadruped (Fig. C.1 bottom left) consists of 12 DoF. The state space $\mathcal{S} \in \mathbb{R}^{21}$ consists of linear and angular base velocity, base orientation, and joint positions. The action space $\mathcal{S} \in \mathbb{R}^{12}$ is in joint impedance control space. The training procedure and reward design to imitate a given trotting reference data set is identical to [67].

MuJoCo benchmarks

For HalfCheetah, Walker, and Hopper (Fig. C.3), the state space consists of joint positions and joint velocities yielding 12, 12, and 6 dimensions of the state space respectively. Ant additionally uses end-effector forces in its state yielding a state space of 20 dimensions. The action space uses direct torque control. The reward of all robots consists of a forward reward and a survival reward, and penalty on energy and contact. For further details, please refer to [186].

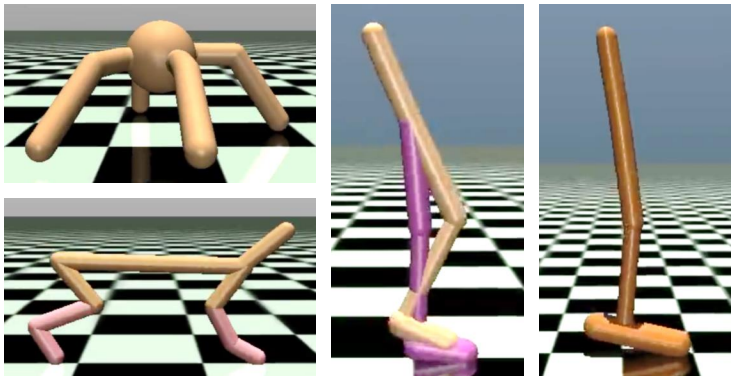


Figure C.3: Various MuJoCo environment for benchmarking: Ant (Top Left), HalfCheetah (Bottom Left), Walker (Middle), Hopper (Right).

C.6 Results

In the following, we present the results obtained by training policies in parallelised environments. All training is conducted on a commercially available computer using single Nvidia GeForce RTX 2080Ti GPU and an Intel i7 8700K CPU. All results are averaged over 5 individual runs.

Real-Time Factor

The aggregated Real-Time Factor (RTF) expresses how much faster data can be collected in simulation than in reality and is calculated as ratio $\text{RTF} = t_{\text{sim}}/t_{\text{real}}$ between simulated time and real time spent collecting the data. The simulated time $t_{\text{sim}} = N_{\text{samples}}/f_{\text{samples}}$ is determined as amount of samples N_{samples} divided by the sampling frequency f_{samples}.

As outlined in Section C.3, a trade-off between the amount of simulation environments and robots in one simulation environment need to be made in order to achieve the highest possible RTF. In Fig. C.4, we show how these parameters are found via a grid-search across the two parameters for the ANYmal environment. Due to software limitations, no more than 1000 robots can be loaded in a simulation environment. For every combination, we collect 10^5 samples at a 25Hz sample frequency which equate to 4000s real time experience. We found the optimal combination at 20 simulation environments and 250 robots per simulation environment leading to an RTF of 202 that collects data of 4000s experience in 19s.

This grid-search approach was used to determine the maximal possible RTF for various environments ("Max RTF" in Table G.1).

For a consistent comparison of the RTF across on- and off-policy algorithms and various environments, we sample the at 25 for a duration of 10s leading to 250 samples per episode. A comparison of real wall clock time to train task solving policies in 6 different environments using PPO and SAC can be found in Table G.1.

For the PPO algorithm, 5000 samples ($T = 250, N = 20$) are collected into a batch which is then used to update the policy. Due to the necessity of rolling out $T = 250$ timesteps to accurately estimate the advantage for PPO, the sample collection time is limited not by the amount of samples to collect but how long it requires to rollout 250 timesteps.

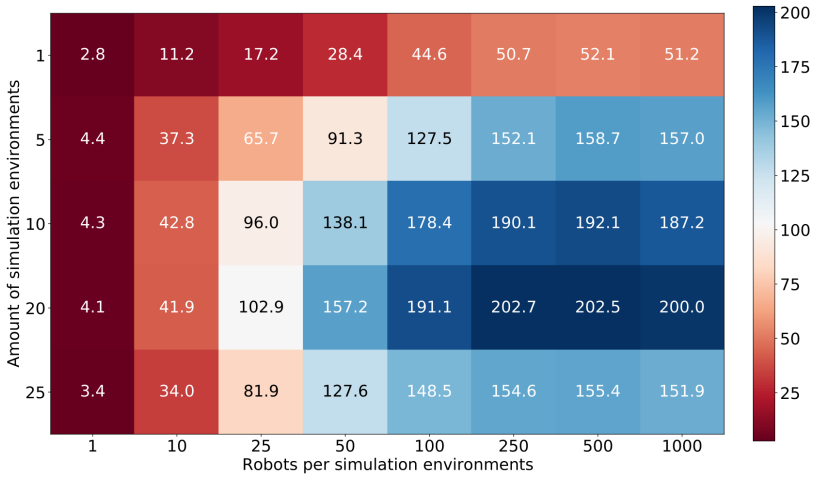


Figure C.4: RTF influence over amount of robots per simulation environment and amount of simulation environments.

Thus, the time to collect 5000 samples is similar to the time to collect 250 samples as the collection of 5000 samples can be parallelised on 20 robots, each simultaneously rolling out 250 timesteps. Larger batch sizes than 5000 would yield a higher RTF and less variance in the data, but no reduction in the sample collection time. Empirically, we found 5000 to be a suitable batch size for PPO, which coincides in magnitude with the batch sizes used in literature [60, 222].

Contrary to PPO, where the majority of time is spent collecting samples, SAC fully leverages the parallelisation capabilities by collecting samples in parallel and storing these in the Replay Buffer (c.f., "SAC RTF" in Table G.1). However, the sample-efficient nature of off-policy algorithms like SAC requires several policy updates on the same data resulting in $K >= 1$ updates per sample. In this benchmark, we update the policy for every sample collected yielding 2500 training updates for 2500 samples collected. Thus, the amount of time spent training is constant across the robot environments as can be seen in the "Train time [s]" column for SAC in Table G.1.

The decision whether to use on- or off-policy algorithm depends on the type of environment. A trade-off is required between time spent for sample-collection bottle-necked by the minimum amount of sequential timesteps for advantage estimation and time spent for updating the Neural Networks. On-policy algorithms are advantageous for simple robotic environments, where the minimum rollout time is less than the time spent for training in off-policy algorithms. However, for more complex robots like for quadrupeds or bipeds, off-policy algorithms are more suitable as the rollout time vastly overshadows the time spent to train.

Deployment on Real Robots

To show that the samples collected are indeed realistic, we transfer the motions from simulation to the real robots. In particular, we implement trotting on ANYmal and balancing on Valkyrie (Fig. C.6). Averaged over 5 training runs, both policies converge after 200 epochs which take 55 minutes in real time (Fig. C.5).

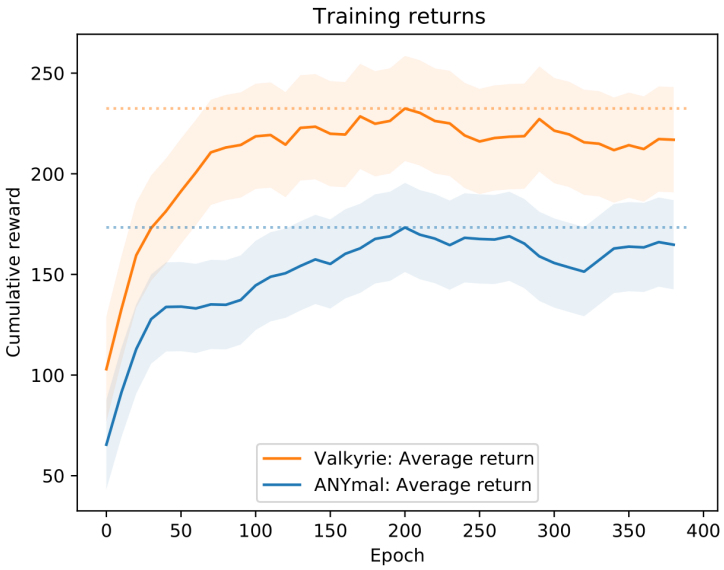


Figure C.5: Learning curve for ANYmal and Valkyrie. Policies converge after 200 epochs and 55 minutes real time training.

The joint position targets provided by the policy are tracked via a Proportional-Derivative controller for both robots. Both robots are torque controlled and use their default joint position controller to calculate the corresponding joint torques. For Valkyrie, the joint torques are obtained through solving an Inverse Dynamics problem that track the joint positions as close as possible. For ANYmal, the joint torques are obtained through directly modulating the stiffness and damping on the joint positions and velocities respectively.

Valkyrie is able to balance in Double Support Phase and withstand constant and impulse external pushes. The policy finds a solution where the Centre of Pressure (CoP) is modulated in order to maintain balance and counteract external forces. The ANYmal robot is able to imitate the trotting behaviour from a given trotting reference data set. The classical alternating diagonal line support phases can be observed in Fig. C.6.

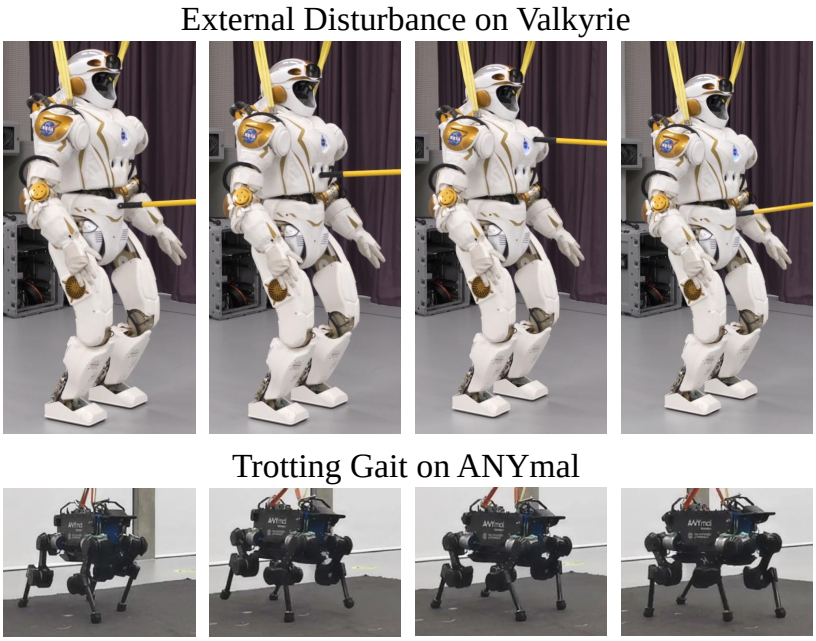


Figure C.6: Snapshots of Valkyrie withstanding various external disturbances (Top) and ANYmal performing trotting gait (Bottom).

C.7 Conclusion

In this work, we proposed a fast sample collection procedure through parallelisation of environments that achieved sample collection on high-degrees of freedom robots like ANYmal and Valkyrie 202 and 120 times faster than real time respectively.

We outlined how parallelisation can be combined with state of the art on- and off-policy Deep Reinforcement Learning algorithms, and showed that a successful policy for trotting on ANYmal and balancing on Valkyrie can be achieved within 1 hour of training. We then directly deploy the train policy on the real robots.

Lastly, we presented results on the MuJoCo benchmarks, solving the problems within minutes of training and achieving a real-time factor of up to 1200 faster than real time.

Decoding Motor Skills of AI and Human Policies

D

D.1 Scientific Motivation

From the advancement in computers, computer-aided design for mechanical and electronic engineering, architecture and many other engineering fields emerged. Foreseeing a similar development curve and technology wave, we forecast a new emerging discipline in the near future that uses learning-aided approaches for catalysing control development, alongside other similar applications such as in medicine discovery. In this study, we propose a new paradigm of using a machine learning approach to facilitate a quicker, more efficient and effective control development, as a different approach of leveraging the power of machine learning in addition to other options that intent to use learning directly in real-world applications.

Machine Learning and Deep Reinforcement Learning (DRL) in particular have reached an advanced stage to produce powerful policies with better autonomous performances than many state-of-the-art control and planning approaches in robot locomotion [66], robotic manipulation [229], and even the control of complex morphological machines [230]. Notably, DRL's ability to solve complex problems with a relatively short development time is especially attractive, which is empowered by training policies that maximise the cumulative reward through the exploration of the action and state space, rather than using prior knowledge of the models about the robot, the world, and their interactions.

To leverage the capabilities of DRL, we first develop a DRL-based control framework to learn rich motor skills of push recovery for humanoid robots. The complexity in whole-body balancing arises in challenges such as multi-contact coordination based on multi-sensory inputs, state transitions between fully- and under-actuated situations, switching policies, and generalising to external disturbances on any body parts, while accounting for all edge cases that a designer has difficulty to consider beforehand. In such a setting, manually designing the individual control strategies and finding a reliable switching mechanism requires both substantial development time, mathematical rigour, and

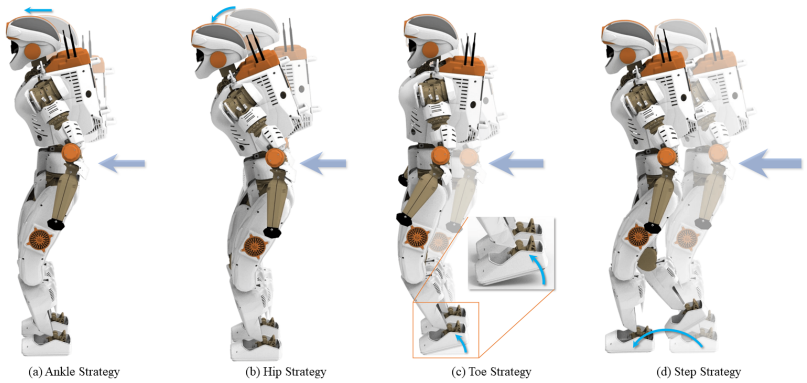


Figure D.1: Human-like Push Recovery strategies emerging from Deep Reinforcement Learning. The discovered behaviours serve as a guideline for the design of certifiable and safe controllers that replicate advantageous strategies from AI policies.

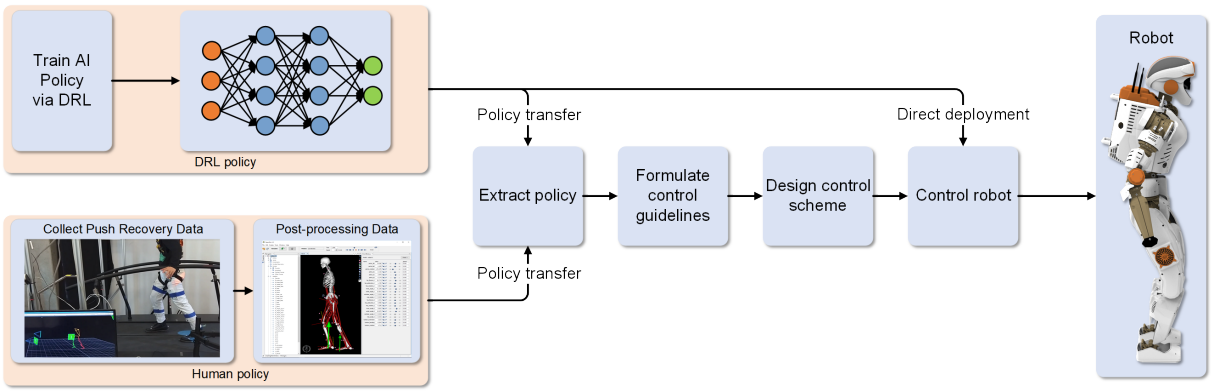


Figure D.2: Proposed control design approach by policy transfer from AI policies to real robotic platforms. By understanding the underlying concepts of the learned policies, we can reverse-engineer a controller that closely resembles the behaviour of the AI policy while being transparent and safe.

code implementation. On the other hand, through a well-designed DRL framework and task-specific training procedures, a robust policy can be learned automatically by interacting with the environment, requiring only computational power. In particular, as shown in Fig. D.1, our **learned policy exhibits human-like push recovery behaviour** with four typical push recovery strategies emerging naturally: ankle, hip, toe, and stepping strategy.

Though the learned control policy could possibly be deployed on the real robotic system, the lack of explainability and analytical reasoning of the Neural Network makes it unsuitable for safety-critical applications in real world. Furthermore, due to the demand of large data and sample-inefficient nature of DRL algorithms, complex policies are typically trained in simulation, which cannot guarantee the same performance while transferred directly to the real system [66], and the challenge of reality gap raises concerning about both the safety and performance.

To benefit from both the **safety and interpretability for the control policy** and the **versatility and adaptability from learning**, we propose to take advantage of DRL to quickly discover versatile, deployable policies and solutions for very difficult problems, and then study, analyse and extract the principles of those policies as guidelines for developing engineered controllers in a reliable manner. By doing so, we utilise the AI-solutions for rapid control development (Fig. D.3) to design safe and certifiable controllers which can be verified and deployed on real-world robots (Fig. D.2).

While classical control development is based on gradually building knowledge that increases the performance incrementally, using a template policy will provide disruptive, innovative solutions that will escalate performance (green line, Fig. D.3). DRL is able to achieve good performance by a number of iterations in the DRL learning framework. However, the achieved performance is still comparatively low to what tuning in control can do. Combining both approaches to “kick-start” the iteration process helps to design good controllers. After knowing the system and the controller, it is straightforward to improve upon due to the fact that we are then able to understand why the performance is lower than the optimum, whereas in the case of DRL, there is little influence from human engineers to improve the performance but reshap-

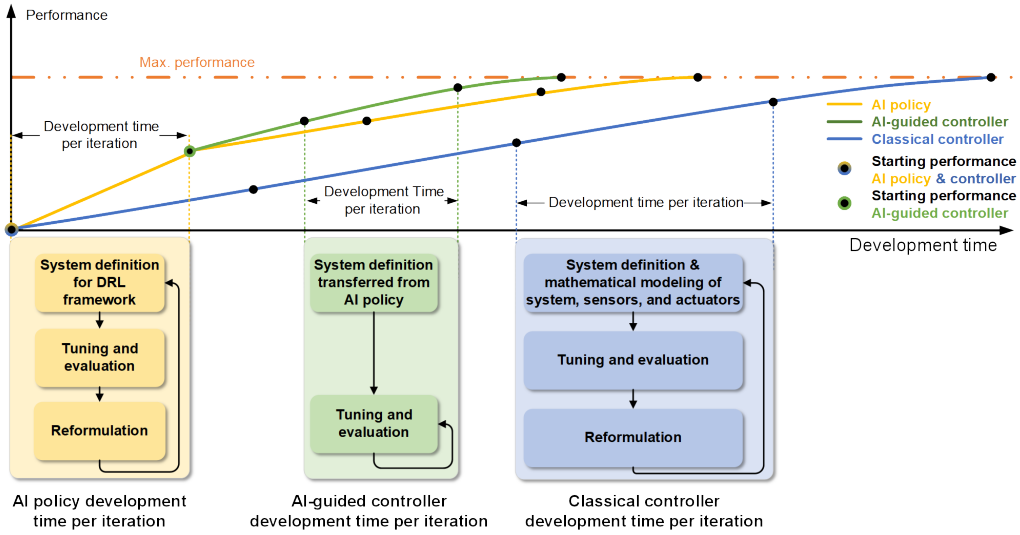


Figure D.3: Qualitative depiction of the performance of the controllers over iterations: leveraging the solution provided by an AI policy decreases the development time of controllers compared to AI policy and traditional controller development while starting at a higher initial performance. The width between every black dot indicates the relative development time of the individual control design approaches with the shortest development time per iteration for the AI-guided controller.

ing the reward and/or altering the learning framework, and relying on the exploration being sufficiently large to achieve high performance.

In this paper, we are motivated to study a viable approach to infer underlying principles of an AI policy by studying its perception-action relation, i.e., to some extent, reverse-engineer an equivalent controller in terms of functionality based on a black-box policy. This methodology is not only applicable to AI policies, but also to any black-box policies, such as a human policy. Without knowing exactly how push recovery policies are realised by Artificial Neural Network (ANN) or biological human Neural Network, we can still analyse the behaviour at the functionality level by studying their input-output relationship.

Based on evidence of optimality in human manipulation tasks [231], we hypothesise that policies for push recovery in humans and humanoid are both optimal control process that follows certain optimal criteria that can be quantified. Following this hypothesis, we analyse and utilise input-output data collected from both humanoid and human policies, and propose a Minimum-Jerk Model-Predictive Control (MJMPC) Framework that is able to quantitatively reflect both the AI and human push recovery policies. The engineered controller has high similarity (Coefficient of Determination more than 90%) with the collected data, and also exhibits the same human-like push recovery strategies, which emerge from the proposed MJMPC without the need of manual switching between the strategies.

Furthermore, a comparison between humanoid and human balancing is conducted to show the characteristics of the learned humanoid behaviour. This comparison will show that DRL algorithms are very powerful to learn a policy (e.g., balancing) within a short development and training time that may require humans years to learn. In contrast, in order to design an engineered controller from scratch with similar performance, months or even years are needed for developmental iterations, mainly because of the high-redundancy and a diversity of control

actions, which are yet challenging to resolve the physical optimality on a high Degree of Freedom (DoF) robot. In this regard, the learning approach is very attractive because of the significant reduction of manual effort, and the learning architecture requires only the design of input-output and rewards. This article shed some light on a new paradigm: the recent high-profile successes in DRL suggest new high-quality value in learning methods that the discovered policies can be used as a basis for speeding up the development of robotic controllers (Fig. D.2). As an outcome in this push recovery study, we obtain a certifiable, analysable optimal controller that does not require any state machine or switching mechanism, while exhibiting human-like push recovery strategies, such as ankle, hip, toe, and stepping strategy all in a coherent optimisation process.

D.2 Generating Complex Motions through Deep Reinforcement Learning

To use DRL-policies as a basis for analysis, these policies must reach a certain performance threshold that ideally surpasses traditional control approaches both in the types of motions it can generate and the amount of disturbances that it can withstand. DRL has been shown to be capable of learning locomotion and fall recovery policies that surpasses traditional control approaches for quadruped robots in terms of power efficiency, and versatility of motion [66].

In this section, we present a hierarchical learning framework for achieving versatile behaviours during push recovery for humanoid robots as proposed in [171]. The learned policy exhibits a wide range of balancing strategies that are comparable to human push recovery. In particular, the learned policy is able to withstand external disturbances by modulation of the Centre of Pressure (Ankle strategy), Angular Momentum (Hip strategy), Centre of Mass height (Toe strategy), and Support Polygon (Stepping strategy) and surpasses traditional control methods with respect to the disturbances that it can withstand.

All motions are trained for deployment on NASA's humanoid Valkyrie [10], a 44 DoF bipedal humanoid robot designed for operating in disaster response scenarios and extraterrestrial planetary space missions such as unmanned pre-deployments on Mars. Valkyrie is built to operate in human-engineered environments standing 1.87m tall and weighing 129kg with ranges of motion similar to humans. Locomotion and manipulation of Valkyrie are enabled through 25 series-elastic actuators in arms, torso, and legs; their respective joint limits are described in Table I.1. For sensing, Valkyrie has proprioceptive and exteroceptive sensors consisting of a multitude of gyroscopes, accelerometers, load cells, pressure sensors, sonar, LIDAR, depth cameras, and stereo sensors.

For training the policy, the physics simulator PyBullet [178] was used, which is able to simulate physics faster than real-time expediting the training process, and simulates collisions, and soft and rigid dynamics by loading objects defined in the Unified Robot Description Format (URDF). The Valkyrie URDF model closely replicates the real robot

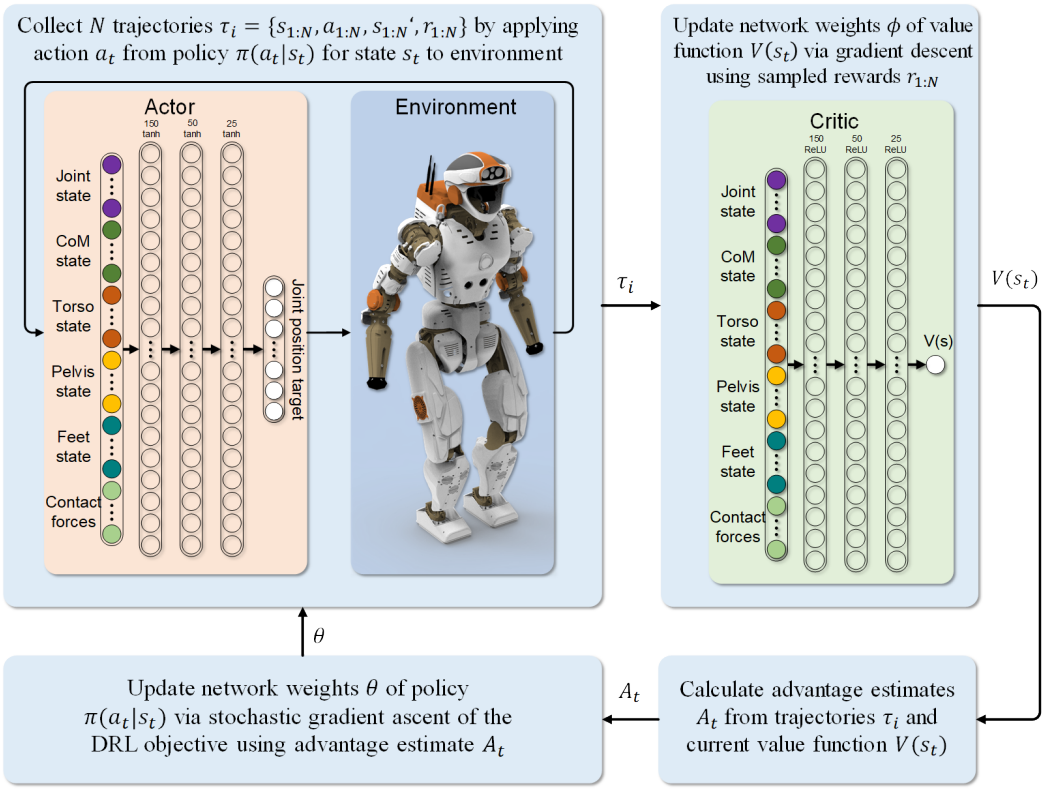


Figure D.4: Learning framework for Deep Reinforcement Learning of a Push Recovery policy.

and is provided by NASA including physical quantities, such as inertia, mass distribution, sensor noise, friction, and damping [10].

Learning Framework

Our DRL framework (Fig. D.4) has an actor-critic architecture, in which both the actor $\pi_\theta(a|s)$ and the critic $V_\phi(s)$ are Neural Networks parametrised by the weights θ and ϕ respectively.

The AI policy $\pi_\theta(a|s)$ is trained through a policy gradient method called Trust-Region Policy Optimization (TRPO) [59]. Policy gradient methods directly model and optimise the policy that will yield the highest reward $J(\theta)$:

$$J(\theta) = \sum_{s\in\mathcal{S}} d^\pi(s) V^\pi(s) = \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) Q^\pi(s,a), \qquad \text{(D.1)}$$

with stationary distribution $d^\pi(s)$, state value function $V^\pi(s)$, and action value function $Q^\pi(s,a)$ under policy π_θ. Using the gradient $\nabla_\theta J(\theta)$, gradient ascent is used to find an optimal parameter set θ that maximises the reward.

Because the true value functions $V^\pi(s), Q^\pi(s,a)$ are not known, a critic $V_\phi(s)$ estimating the true value function $V^\pi(s)$ will be trained. The critic $V_\phi(s)$ is updated by minimising the expected loss $L_V(\phi)$ via gradient descent:

$$\min_\phi L_V(\phi) = \min_\phi \mathbb{E}[(V_\phi(s_t) - G_t)^2], \qquad \text{(D.2)}$$

with value function estimation $V_\phi(s_t)$ and return G_t. The return $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k$ is the discounted cumulative reward using the discount factor $\gamma \leq 1$ and reward r_t.

Using the value function estimation $V_\phi(s_t)$, the actor's parameters θ are updated by maximising the reward (D.1) via stochastic gradient ascent. Additionally, TRPO improves the training stability through trust-region optimisation - constraining the KL divergence during a policy update preventing large policy changes that could cause instability - and reduces the variance of the policy gradient estimate using an advantage estimation A_t:

$$\begin{aligned} \max_\theta J(\theta) &= \max_\theta \mathbb{E}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_t(s_t)] \\ \text{subject to} \quad & \mathbb{E}[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)||\pi_\theta(\cdot|s))] \leq \delta, \end{aligned} \tag{D.3}$$

with updated policy $\pi_\theta(a|s)$, old policy $\pi_\theta(a|s)$, advantage estimate $A_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l (r_t + V(s_{t+1}) - V(s_t))$, discount factor γ, variance/bias trade-off parameter $\lambda \in [0, 1]$, KL divergence $D_{\text{KL}}(\cdot||\cdot)$, and trust region δ.

During the training process, by sampling the action a_t from the stochastic policy $\pi_\theta(a_t|s_t)$ and performing this action, the environment returns the resulting reward r_t, and the corresponding state s_{t+1}. During every update step during training, so-called SARS tuples $\{s_t, a_t, r_t, s_{t+1}\}$ are collected in a batch $\mathcal{D}$. Upon reaching a certain batch size N - the size of the batch is a hyper-parameter trading off the variance of the data against the speed of training - the actor and critic are then updated through gradient ascent and descent respectively. While a stochastic policy provides exploration, additionally, by applying random forces to the pelvis, the policy's ability to withstand large pushes is increased due to a larger coverage of the state space in the stored experience data. With the additional push experience during training, the policy will be able to generalise better to push disturbances during runtime.

Inferring actions from the AI policy

By training in the presented learning framework and interacting with the environment, the AI policy experiences what reward certain actions in different states will yield. Using these experiences in form of SARS tuple trajectories $\tau = \{s_{1:N}, a_{1:N}, r_{1:N}, s'_{1:N}\}$ and reward maximisation through gradient ascent, the policy is incentivised to perform actions that will lead to high-value states while avoiding low-value states. Consequently the AI policy $\pi(a|s)$ learns how to optimally act in state s_t by performing actions a_t to maximise a human-defined reward function r.

The reward function is designed as:

$$r = r_{\text{pose}} + r_{\text{CoM pos}} + r_{\text{CoM vel}} + r_{\text{GRF}} + r_{\text{contact}} + r_{\text{power}}. \tag{D.4}$$

The reward function is designed such that high rewards are given if torso pose r_{pose}, Centre of Mass (CoM) position $r_{\text{CoM pos}}$, CoM velocity $r_{\text{CoM vel}}$, and ground contact force r_{GRF} (equally distributed ground reaction forces) remain close to the nominal values. A radial basis

function $r_i = \exp(-\alpha_i(x_{\text{target}} - x)^2)$ is used for torso pose, CoM position, CoM velocity, and ground contact force to encourage the robot being close to a target position. Furthermore, a penalty (negative value) is given proportional to the power consumption r_{power}, and if the upper body is in contact with ground or the foot is not in contact with the ground r_{contact}. For the contact penalty, a constant value is subtracted if there was no ground contact of the foot or an upper body contact with the ground.

The high-level AI policy generates actions in form of target joint angle references at a frequency of $25Hz$ by forward-propagating through the actor network using the current state as input. The robot performs its motions with upper body joints locked in their nominal position. The action space $\mathcal{A} \in \mathbb{R}^{11}$ thus consists of joint positions for torso pitch, hip roll, hip pitch, knee pitch, ankle roll, and ankle pitch. The target joint angles are given to a low-level Proportional Derivative (PD) controller operating at 500Hz to generate the joint torques that are ultimately applied to control the robot (Top of Fig. D.6).

The state space $\mathcal{S} \in \mathbb{R}^{47}$ consists of joint position and velocity of the actuated joints, pelvis states (translational and angular velocity, orientation), CoM states (translational velocity and position in local frame), ground contact force, torso position (in local frame), and foot position (in local frame). The state is sampled at a frequency of 500Hz and filtered by a first-order Butterworth filter with a cut-off frequency of 10Hz.

More details regarding the formulations of the learning framework can be found in [171].

Behaviours of the AI Policy

Training a robust push recovery policy in the presented learning framework requires 6-8 hours of simulation time on a commercial desktop PC (Intel i7 6700K, Nvidia Titan X, Tensorflow) and equates to 1-2 days in real time. The learned push recovery policy demonstrates human-like push recovery strategies such as ankle, hip, toe, and stepping strategy which emerge at different levels of disturbance (Fig. D.1). The policy performs well in the presence of external disturbances and large sensor noises due to the filtering and sufficient amount of exploration.

Notably, the AI is able to generalise to external disturbances it was not trained for. In Figure D.5 snapshots of Valkyrie are shown during which an impulse push on the shin is performed. The policy is able to generalise to this untrained case by generating a stepping behaviour. This generalisation ability further extends to the ability of recovering from an impact during landing from 0.55m height.

The performance of the policy learned by the AI is compared with push recovery controllers in terms of maximum impulse disturbance. The AI demonstrates the ability, comparable to state-of-the-art push recovery controllers, to withstand a wide range of impulse disturbances. The details of the comparison are presented in [171]. In Section D.4 we will further show that the motions are human comparable both quantitatively and qualitatively.

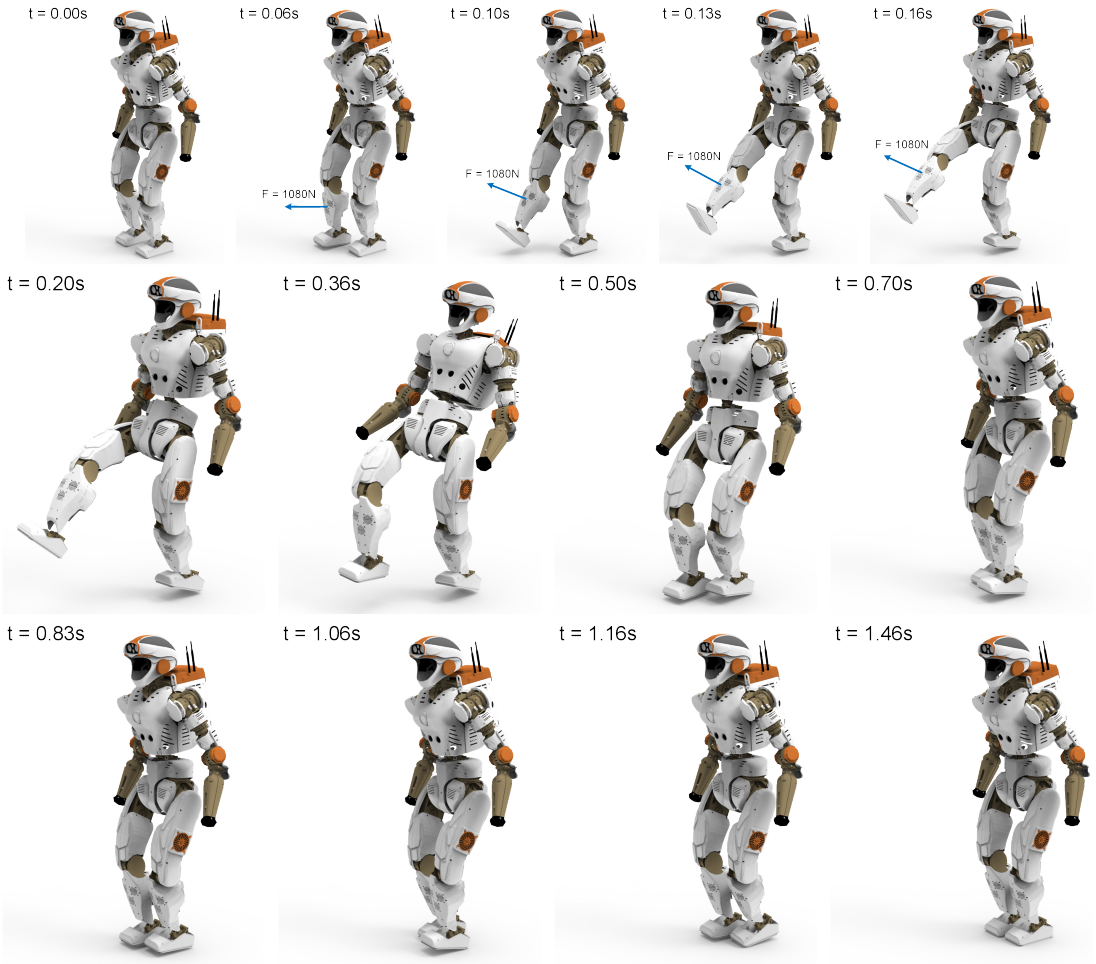


Figure D.5: Valkyrie recovering from an unexpected disturbance in a test scenario which is never encountered during training (impulse at the shin of 108Ns). The learned policy naturally evolves and generates a stepping behaviour. Top row: a nominal starting pose (left), and the motion during the disturbance (being pulled at the shin); two bottom rows: recovery reactions where Valkyrie takes 3 steps backwards and successfully recovers balance.

D.3 Understanding the Fundamental Principles From the AI-Policy

The idea of using AI-generated policies as inspiration for solving hard problems has taken flight in other fields, most notably in games such as Go, Chess and Shogi [232] for which the AI policy achieved super-human level performance. Analysing the concepts and solution process of the AI allows humans to construct better solutions and improve the way humans approach these problems: AlphaGo, a super-human policy that beat 18-time world champion Lee Seedol 4:1, has subsequently been analysed to provide humans with insights on why and how AlphaGo won. The release of AlphaGo Teach enabled Go-players to analyse strategies, and changed the way in which humans played the game: AlphaGo Teach helped to quantify the value of starting the game (suggesting that the current compensation of 6.5 points for not starting puts the non-starting player in an advantage), redefined conventional openings, and utilised unconventional moves that went against conventional human wisdom and were previously deemed as disadvantageous.

Inspired by the performance of the AI policy, this section aims to

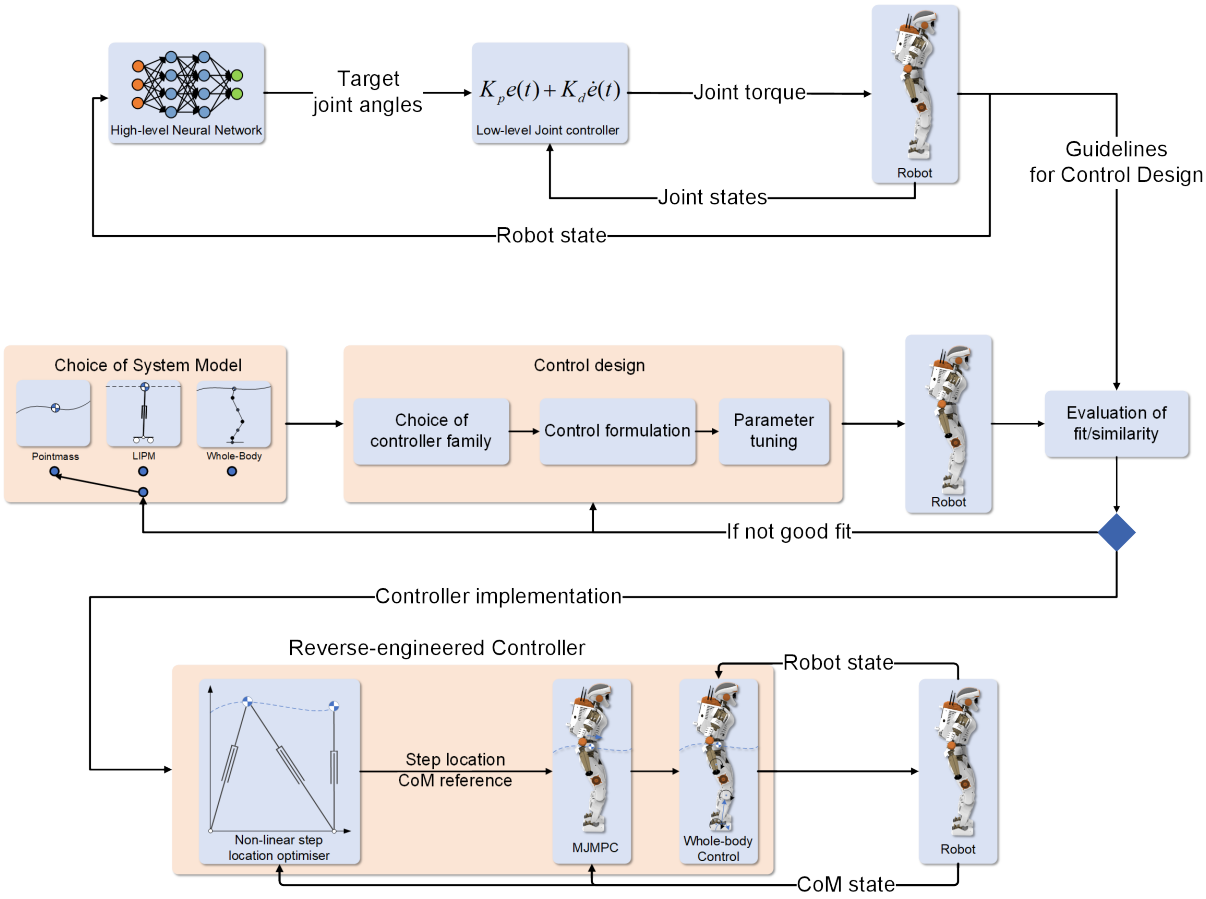


Figure D.6: Process for AI-aided control design. Top: Hierarchical control system for push recovery. The high-level neural network is learned as depicted in Fig. D.4. Mid: Design process of the engineered controller. Bottom: Implementation of the designed controller into a whole-body control framework.

present an approach to reverse-engineer the policy to obtain a unified controller exhibiting the same push recovery strategies as the AI policy. By reverse-engineering the AI policy, its versatility is maintained while enabling us to analyse and modify the controller with respect to stability and safety.

We aim to find a controller which is able to stabilise the system, and has a close similarity and fit to the DRL policy. To this end, a choice of system model and the controller type need to be made. In the following, we will show that the DRL policy can be accurately replicated with point-mass dynamics that is controlled by a minimum jerk controller. This indicates that the NN may internally use a point-mass template and try to optimise the jerk.

The design process (Fig. D.6) involves three steps: acquiring state-action data from the AI policy, reverse-engineering the policy, and deploying the controller in a whole-body control framework. First, various state-action pairs are acquired through simulation for different external disturbances. In the reverse-engineering process, based on the criterion of least square error fit, both a suitable system dynamics representation and a controller using these system dynamics will be determined. Lastly, in the deployment phase, the engineered controller will be used to generate step locations and the CoM trajectory which will then be tracked by a whole-body controller.

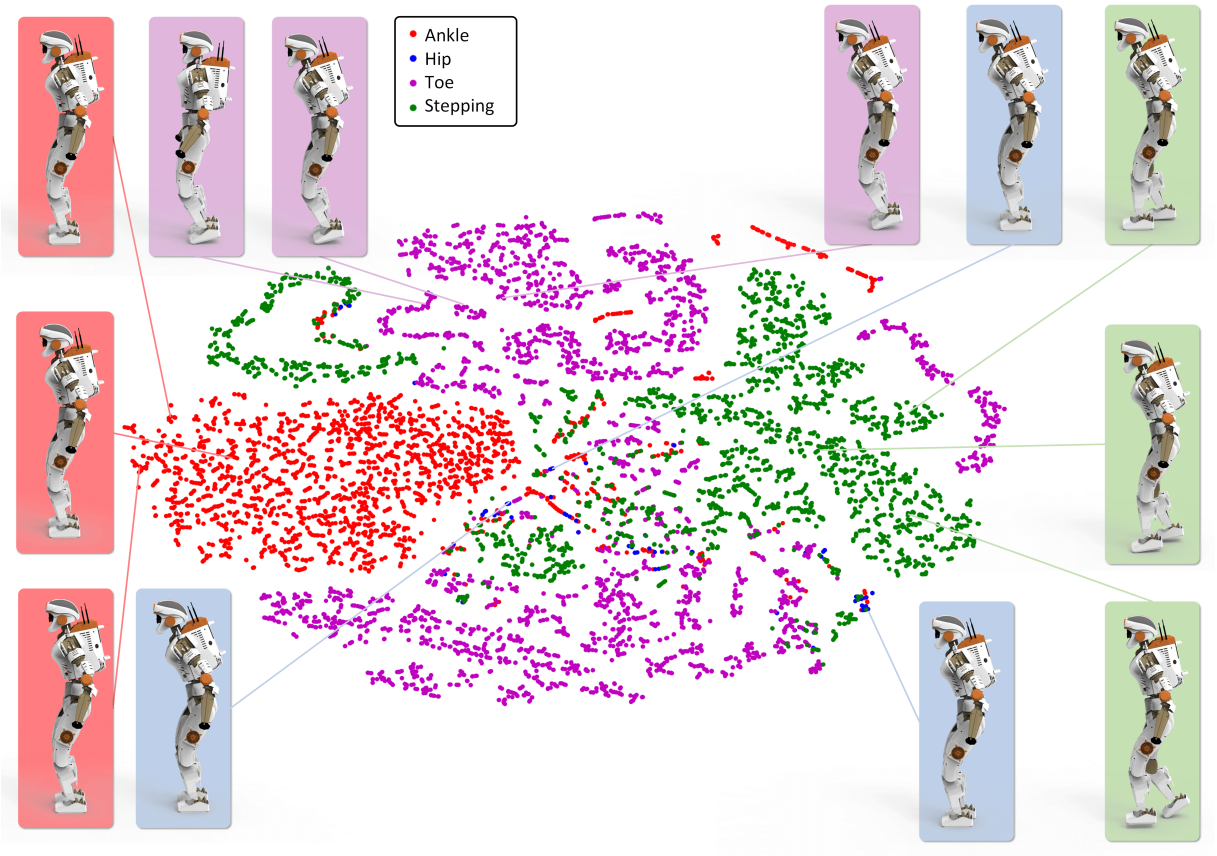


Figure D.7: T-SNE with every dot indicating the activation of all 175 neurons during one time step. The classification of every dot is determined by whether they are non-zero (exceed a threshold) for foot velocity (step), angular momentum (hip), vertical CoM velocity (toe), and is Ankle Strategy otherwise.

Analysing the AI-policy

The Actor Neural Network can be analysed thoroughly with the goal of finding guidelines and quantities that enable general push recovery. The insights gathered from analysing the actor can be used to improve controller design. Methods for analysing and interpreting the NN are mainly of visual nature and originate from the field of Computer Vision.

In this work, we visualise the NN's activation by using t-distributed Stochastic Neighbor Embedding (t-SNE) to project the activation of NN's onto a 2D plane while preserving neighbourhoods and clusters in projections [180]. T-SNE is performed on the neuron activations to project the high-dimensional NN activation on a 2-dimensional space in order to investigate the generalisation behaviour of the policy. In the final projection, data points that represent similar NN activation are grouped nearby with high probability, whereas non-similar data points are distant from each other. We treat all neuron activations (c.f. Fig. D.4) during one time-step as one high-dimensional data point for the t-SNE analysis. For every time step of a trial we collect all neuron activations across disturbances ranging from $0\text{Ns} - 210\text{Ns}$. During these disturbances, all four push recovery strategies (Fig. D.1) occurred and are labelled by colour in Fig. D.7.

As can be seen in Figure D.7, the distinct strategies are cover separate

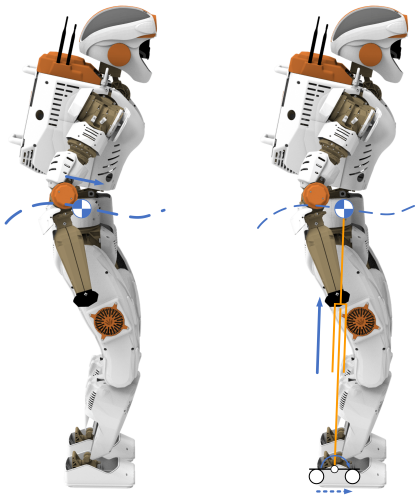


Figure D.8: System models used for control design. Left: Point-mass model with free moving CoM independent on actuation. This model is used for our proposed Minimum Jerk Model-Predictive Controller. Right: Inverted-Pendulum model with actuated ankle torque and kick-force and free moving CoM according to the Inverted Pendulum motions. This model is used to validate and verify that the controller generated a feasible and implementable trajectory.

regions and thus partly explain the ability of the AI policy to generalise across different disturbances, as the activation is similar for the same strategy. NN representations that are labelled as the same strategy are perceptually similar and are projected close to each other. The points that represent ankle strategy cluster well with few outliers and is visually distinct from other strategies. The toe strategy and stepping strategy are also quite distinct from each other. Furthermore, the lack of activations corresponding to the hip strategy (blue points in Fig. D.7) indicate that the AI policy does not utilise the Angular Momentum as much as the other strategies and could suggest that the hip strategy should be used less than other strategies, such as the toe strategy. The hip strategy does not cluster as clearly as the other strategies, thus we hypothesise that it is an artefact of other motions rather than an intentional motion. The effectiveness of the hip strategy is further questioned in the literature [233], and should be kept in mind when reverse-engineering the AI policy.

Incorporating AI-policy inspired principles in the control design

The close resemblance between the DRL (dashed lines in Fig. D.10) and human data (dashed lines in Fig. D.13), for which strong evidence of being minimum jerk exists [231], [234], along with the typical smoothness characteristics of minimum jerk trajectories motivates the design of a minimum jerk control scheme. Inspired by this, reverse-engineering the AI policy involved investigating whether it is also minimum jerk[1]. We designed an MPC controller that is using point-mass dynamics to minimise the jerk of the CoM state for motion generation and feedback control. In order to prevent that arbitrarily large motions are generated that violate the actuation constraints of the robot, the CoM state is further constrained in the MPC scheme.

The idea of MPC lies in solving an optimisation problem at every time-step. By using the feedback of the current state, a closed-loop control behaviour can be achieved. The objective function $J = \frac{1}{2}\int_0^{t_f}\left(\frac{\mathrm{d}^3 x(t)}{\mathrm{d}t^3}\right)^2 \mathrm{d}t =$

1: For the control design process of reverse-engineering the AI policy, we followed the processes as in Fig. 5. Our study eventually found that applying the controller proposed in our previous work [234] leads to the best fitting results, and strong resemblance between the AI policy and the reconstructed controller. In contrast, the further tested different models (point-mass, Inverted Pendulum, Whole-Body Model), and controllers (PD control, Linear Quadratic Regulators, and MPC) have worse fitting to the data.

$\frac{1}{2}\int_0^{t_f} u(t)^2 \mathrm{d}t$ is designed to minimise jerk $\dddot{x}$ (the input u of the system) with final time t_f. The MPC solves the following constrained optimisation problem:

$$\begin{aligned} \min_{u(t)} \quad & \frac{1}{2}\int_0^{t_f} u(t)^2 \mathrm{d}t \\ \text{subject to} \quad & \frac{\mathrm{d}^3 x(t)}{\mathrm{d}t^3} = u \\ & [x(0), \dot{x}(0), \ddot{x}(0)] = [x_0, \dot{x}_0, \ddot{x}_0] \\ & [x(t_f), \dot{x}(t_f), \ddot{x}(t_f)] = [x_f, \dot{x}_f, \ddot{x}_f] \\ & [x_{\min}, \dot{x}_{\min}, \ddot{x}_{\min}] \leq [x, \dot{x}, \ddot{x}] \leq [x_{\max}, \dot{x}_{\max}, \ddot{x}_{\max}], \end{aligned} \tag{D.5}$$

with initial condition $[x_0, \dot{x}_0, \ddot{x}_0]$, and terminal condition $[x_f, \dot{x}_f, \ddot{x}_f]$. The upper and lower inequality constraints of the optimisation problem prevent the Minimum Jerk Model-Predictive Control (MJMPC) scheme outputting trajectories that the subsequent whole-body controller cannot track. We constrain the maximum vertical displacement, velocity, and acceleration of the CoM, as well as the maximum jerk the point mass can be exposed to. For the final state two quantities need to be determined: the final time t_f at which the system should reach state x_f, and the final CoM state itself which is provided by non-linear step optimiser [235] and thus evokes stepping behaviour if the push gets too large.

Determining final time via data fitting

While the final time t_f, at which the robot should come to a rest, could also be optimised, this would introduce non-linearity to the optimisation problem. Thus, we treat the final time t_f as an open parameter: too short and it violates the physical capabilities, too long and it may get unstable or use too much energy. We determine an appropriate value for t_f via least square error fitting between collected DRL data and the MJMPC. In Figure D.9 it can be seen that both a piece-wise linear and a quadratic approximation are able to fit the data.

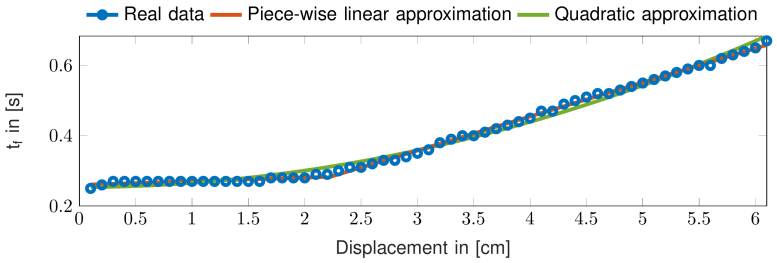


Figure D.9: CoM displacement from nominal pose over t_f for the robot.

Non-linear Step Optimisation

In an outer loop, a non-linear step optimiser [235] provides the MJMPC scheme with a reference point. The position x_f in the final state $x_f = [x_f, 0, 0]$ is determined by the non-linear step location optimiser, and the velocity $\dot{x}_f$ and acceleration $\ddot{x}_f$ are set to zero. The step optimiser is able find a step location and timing within 3ms due to the specific

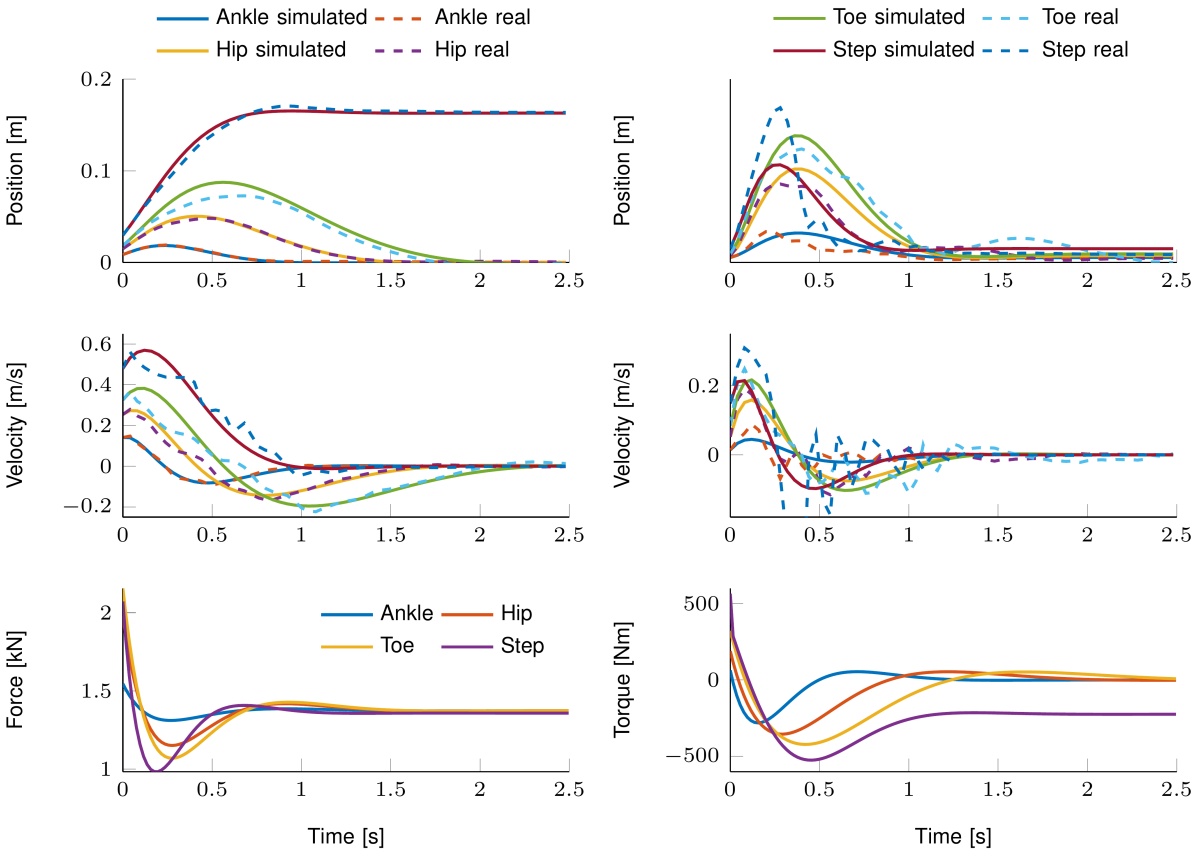


Figure D.10: Fit between robot data and reverse-engineered controller. Left column: x position, velocity, and required force. Right column: z position, velocity, and required torque

formulation of the optimisation problem. Consequently, we are able to run the optimiser at real time at 100Hz. Furthermore, the non-linear optimiser considers the kinematic constraints of the robot to determine the maximal step length, while also constraining the minimal and maximal step velocity of the robot.

Emerging Strategies and Quality of Fit

From the x and z CoM position trajectories in Figure D.10 it can be seen that MJMPC generates trajectories from which the push recovery strategies (Fig. D.1) emerge naturally. From the data of the DRL policy we found that very little Angular Momentum was generated (Section D.3), and consequently do not regulate the angular momentum of the humanoid. Typical for the ankle strategy (blue solid line), the Centre of Pressure (CoP) is moved within the Support Polygon to move the CoM position. The CoM height modulation emerges from regulating the CoM height to its nominal position. For large pushes, the step optimiser sets a new reference position, and thus stepping behaviour emerges.

To show the quality of fit, we identify the open parameters by optimising these via least square error between the controller and the DRL policy. We apply k-fold cross validation across 2000 trials of the robot, and show the mean and standard deviation in Table D.1 with an average

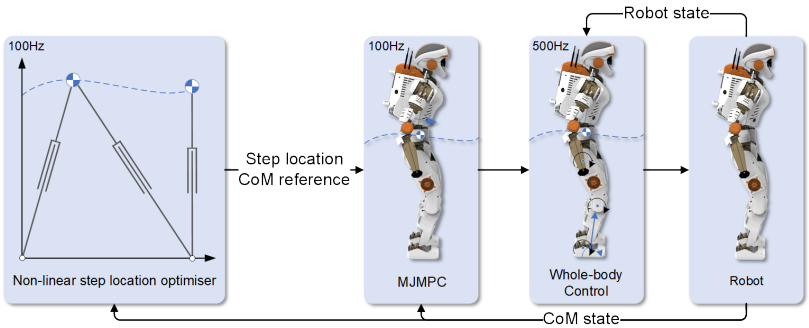


Figure D.11: Control diagram for the robot.

coefficient of determination of 0.95 and 0.91 for x and z direction respectively.

Table D.1: Average Coefficient of Determination (R^2) over 2000 trials between DRL and robot trajectories for Ankle, Hip, Toe, and Step Strategy.

	Mean R^2				
Axis	Ankle	Hip	Toe	Step	Total
X	0.97	0.94	0.93	0.96	0.95
Z	0.90	0.91	0.96	0.88	0.91

As can be seen in Figure D.10, by using the methodology described in Section 1, a final time t_f can be chosen such that the engineered policy fits the DRL, indicating that MJMPC is a suitable controller for resembling the AI policy. The fit for the vertical component is slightly worse due to the fact, that the approximation of t_f for the CoM height was not as well as for the sagittal component.

Realisability of MJMPC on Real Systems

In the following, we propose a framework for real world deployment of the generated push recovery motions and demonstrate the feasibility of the motions generated from MJMPC.

In order to deploy the push recovery motions on a real robot while guaranteeing stability and implementability, a whole-body controller is included in the control framework (Fig. D.11). To this end, Quadratic Programming (QP) based whole-body controllers can be leveraged to track reference motions while providing stability and were successfully deployed on humanoids such as Valkyrie [10], Atlas [204], and HRP-2 [236]. Beside the ability to incorporate stability and feasibility guaranteeing constraints in the optimisation problem formulation, QP problems can be solved extremely fast with off the shelf QP solvers enabling loop closure at over $500Hz$. Furthermore, in practise, whole-body QP controllers exhibit robustness to model uncertainties due to fast frequent loop closure, and have been shown to reliably work on the real Valkyrie platform.

After the MJMPC generates a reference CoM trajectory y_{ref}, a whole-body controller is minimising the tracking error $J_{\text{task}} = \frac{1}{2}\|y - y_{\text{ref}}\|$ while guaranteeing physically feasible torques realising the push recovery motion. Reformulation of the tracking tasks of $J_{\text{task}} = \frac{1}{2}\|y - y_{\text{ref}}\|^2 = \frac{1}{2}\|AX - b\|^2$ with state $X = [\ddot{q}, \tau, \lambda]^T$ consisting of torque commands

τ, joint accelerations $\ddot{q}$ and contact wrench λ yields the matrices $H = A^T A, f = -A^T b$. The QP problem is formulated as:

$$\begin{aligned} \min_X \; & X^T H X + f^T X \\ s.t. \quad & A_{eq} X + B_{eq} = 0 \\ & A_{ineq} X + B_{ineq} \geq 0. \end{aligned} \tag{D.6}$$

Further tasks, e.g., tracking feet trajectories from the non-linear step location optimiser, regularisation terms, or other tasks benefiting the stability of the controller, can be stacked in $A = [w_0 A_0, w_1 A_1, ..., w_n A_n]^T, b = [w_0 b_0, w_1 b_1, ..., w_n b_n]^T$ and task priorities are represented by the weights w_i, where $i = 0, ..., n$ is the i-th task (for details, c.f., [204]).

The equations of motion guaranteeing physical coherence between the states form the equality constraints of the QP problem (D.6):

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \end{bmatrix} + h(q, \dot{q}) = 0, \tag{D.7}$$

with inertia matrix $M(q)$, selection matrix S, Jacobian matrices $J^T(q)$ of the contact links, and nonlinear effects $h(q, \dot{q})$.

In addition to physical coherence of the solution, locomotion specific constraints are formulated guaranteeing that the robot will not fall over. This is achieved by imposing inequality constraints in the QP problem on the contact wrench $\lambda = [f_x, f_y, f_z, \tau_x, \tau_y, \tau_z]$ between feet and the ground. Slippage in x, y direction is prevented by constraining the force in the respective direction $|f_x| \leq \mu f_z, |f_y| \leq \mu f_z$ for a given friction coefficient μ. Unilateral forces (no suction of the feet to the ground) are guaranteed by $f_z > 0$. A stability constraint is achieved by constraining the CoP within the Support Polygon $|\tau_x| \leq \mathcal{Y} f_z, |\tau_y| \leq \mathcal{X} f_z$ with dimensions $\mathcal{X}, \mathcal{Y}$ of the Support Polygon. Lastly yaw slippage is prevented by $\tau_{min} \leq \tau_z \leq \tau_{max}$ with $\tau_{min} = -\mu(X+Y) f_z + |\mathcal{Y} f_x - \mu\tau_x| + |\mathcal{X} f_y - \mu\tau_y|, \tau_{max} = +\mu(X+Y) f_z - |\mathcal{Y} f_x + \mu\tau_x| - |\mathcal{X} f_y + \mu\tau_y|$.

The physical feasibility of the motions on the actuators during push recovery motions can be further shown using an Inverted Pendulum Model. The Inverted Pendulum (IP) Model [31] simplifies the dynamics of a robot into an IP whose length can be extended via a "kick-force" and a torque applied to the pivot point corresponding to the ground reaction force and torque of the real robot. Using the torque limits of the actuators, the robot can produce a ground contact force of $f_{max} = 2500$N and torque of $\tau_{max} = 1000$Nm. Using the IP model, the required ground contact forces and torques for tracking the CoM reference can be calculated and are shown in Figure D.10. For all four push recovery scenarios, the required force and torques are well below the maximal generatable ground contact force and torque of $f_{max} = 2500$N and $\tau_{max} = 1000$Nm respectively.

Summarised, feasible and stable motions are enabled using this proposed control framework of first generating push recovery motions via MJMPC and then tracking the motions with a whole-body controller. In the inequality constraint (D.5) of the MJMPC optimisation problem it is explicitly considered at which speed the whole-body controller can

track a reference motion. Therefore, trackability of the reference motion is guaranteed. Furthermore, for the whole-body controller, the CoM Height Modulation does not necessarily need to come from a toe lift, but can come from any leg length extension increasing the CoM height. This is possible if the gait is not performed with stretched knees which is mostly the case in robot control tasks that aim to prevent singularities in a stretched knee pose.

D.4 Benchmarking Between Human- and AI-policies

To this point, we have shown that methods of policy extraction can be used to estimate and reconstruct control policies from DRL. Interestingly, when these methods are applied to human movement similar strategies are revealed [234]. In this section we compare the findings from our previous work with those based on the AI policy to highlight the similarities between policies extracted from humans and DRL. This will show that even though training in DRL can take just hours, the emergent behaviours are similar to human behaviours which are highly successful only after being refined over months. These similarities show that DRL can be a high quality source of inspiration for control design since it can closely match human level behaviour.

Data Collection

To compare human and DRL push recovery policies, human data was recorded in 60 trials while short impulses were applied close to their CoM (Fig. D.12 left), analogous to those applied to robots during DRL testing. After the push ends (Fig. D.12 middle) the subject takes a recovery action (e.g., stepping in Fig. D.12 right). Different impulse magnitudes were applied to obtain a wide range of push recovery behaviour. Impulses were measured by a Force/Torque sensor. Movement was measured via a VICON motion tracking system and the data was post-processed using OpenSim and gait analysis tools as presented in [237].

Push Recovery Strategies in Humans and AI policies

Analysis of the human data [234] shows that various strategies are used for push recovery as shown in Fig. D.1. Each strategy is associated with a control action such as the modulation of (CoP), Angular Momentum, CoM Height, or Support Polygon. Interestingly, a very similar structure also emerges in the AI policy, which demonstrates the similarities between DRL and human policies.

Control Strategy

We find that a single controller implementing the MJMPC scheme in Section D.3 can explain human data across all strategies (Fig. D.13) as

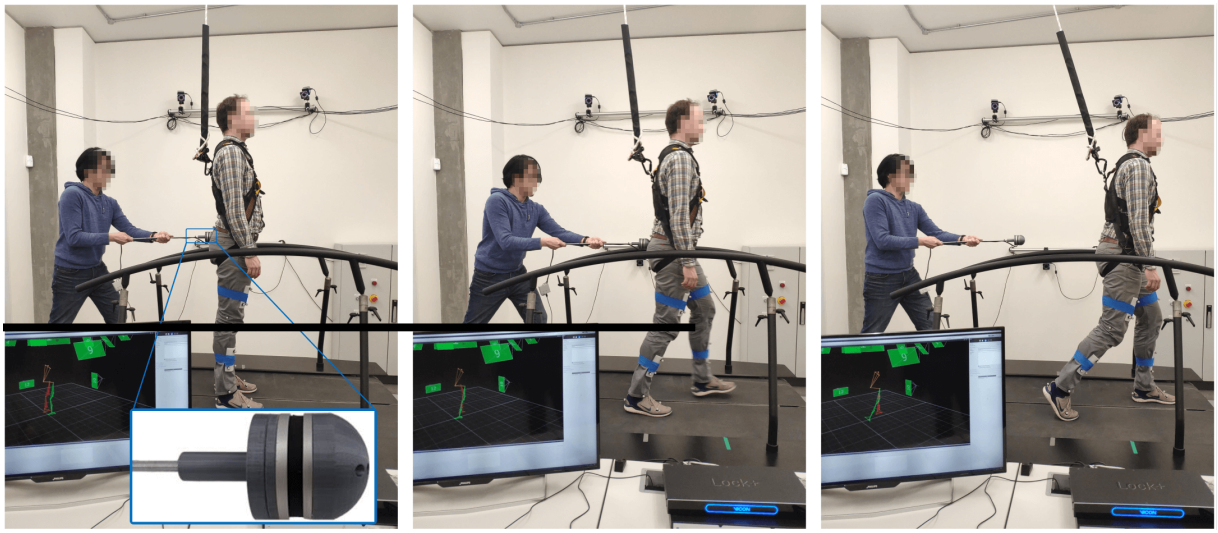


Figure D.12: Example of one experiment trial. Left: subject stands upright during the beginning of the push. Middle: subject transits into stepping strategy after being pushed. Right: subject comes to a halt after stepping action.

was the case for DRL. Comparing these to the DRL results (Fig. D.10, Table D.1) and it becomes apparent that the trajectories are similar in each strategy. In conclusion, applying policy extraction methods to humans and DRL revealed that they are generalisable to different sources and shows that the respective policies are very similar, as evidenced by the MJMPC controller.

Table D.2: Coefficient of Determination (R^2) between human and robot trajectories for Ankle, Hip, Toe, and Step Strategy.

	Mean R^2				
Axis	Ankle	Hip	Toe-Lift	Step	Total
X	0.76	0.84	0.95	0.89	0.86
Z	0.95	0.92	0.86	0.59	0.83

Why Learn From DRL Policies Instead of From Humans?

This section showed that the policies of DRL and humans and their derived, reversed-engineered controllers are similar, which raises the question: why use an AI policy instead of learning from a human policy? The answer lies in the logistics and applicability of both methods for control design. While humans could be used as template to gain insights on how to improve the target system (humanoid robots), learning from an AI policy yields three main advantages:

1. Access to all internal states: by deploying DRL on the same system as the target system, the state space is identical. Thus, data can be collected from DRL learned policies as soon as the learning process is complete and is immediately available for analysis.
   In the case of human policy analysis, data post-processing is highly labour intensive and requires expensive data collection equipment in order to infer accurate joint angle, joint torques and CoM positions [237]. Data collection required around two weeks

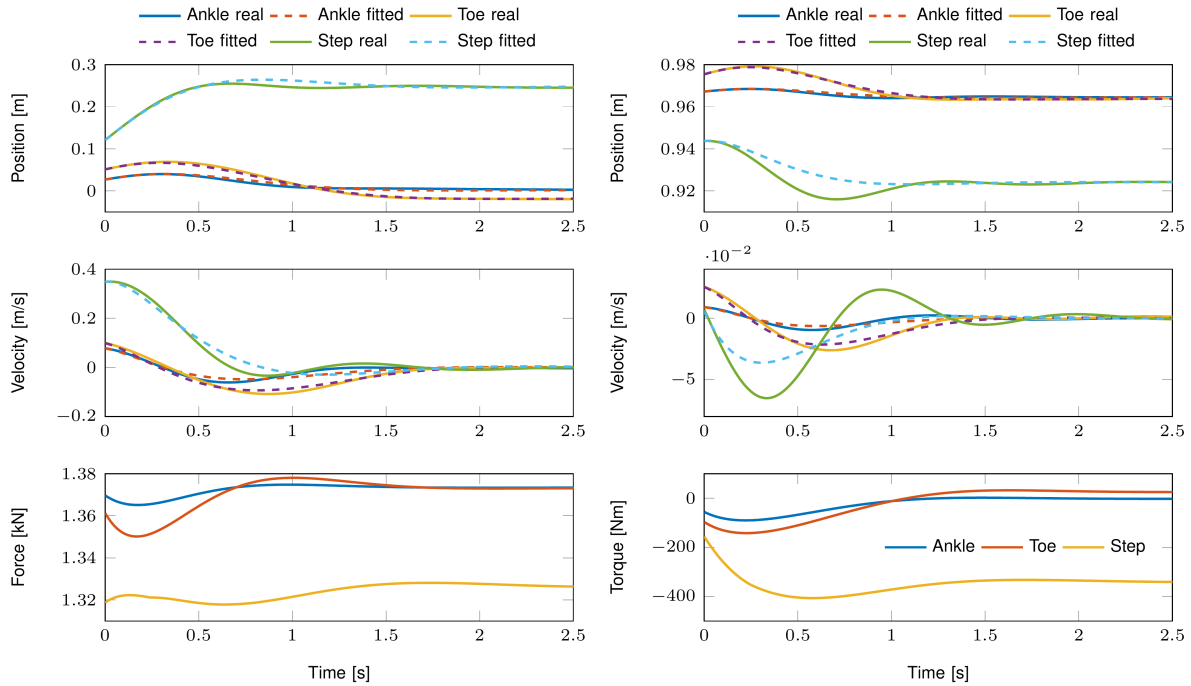


Figure D.13: Fit between human data and reverse-engineered controller. Left column: x position, velocity, and required force. Right column: z position, velocity, and required torque.

of work, including setting up experiments, collecting subjects, carrying out experiments and processing data.
These labour intensive logistics further motivate the use of DRL to transfer policies over the collection of human data since a lot more data can be extracted in a shorter time from DRL trials.

2. Policy transfer to non-humanoid robots: using humans as template policy only allows a policy transfer to humanoids. In contrast, DRL policies can be applied to system which are non-humanoid, such as quadrupeds or multi-legged robots, or where the template is not available, such as extinct vertebrates [238].
3. Analysis of the internal mechanisms of the policy: For analysis of the policy beyond its input-output relations, e.g., t-SNE analysis (Section D.3), the AI policy is more accessible than the human policy. The AI policy is represented as NN and analysis tools outlined in Section D.3 can be leveraged to gain insights in the mechanisms of the AI policy. Analysing the mechanisms of the human policy on the other hand require a neuroscientific understanding of the brain and other involved components of the humans.

The summarised strengths and limitations of DRL in a number of different areas with respect to classical control and human control can be found in Table D.3.

Table D.3: Comparison between various control paradigms. Note that the limitations of DRL in optimality, robustness, and safety are cancelled out by the strengths of classical control in these areas, and vice versa. The "Control + DRL" paradigm can overcome the difficulties which are encountered in human-inspired control (last column).

Attribute	Control	DRL	Humans	Control + DRL	Control + Human
Optimality	Optimal	Sub-Optimal	Near-Optimal	Optimal	Optimal
Robustness	High	Low	High	High	High
Behaviour Emergence Time	Weeks	Weeks	Months	Weeks	Weeks
Generalisability	High	High	Low	High	Low
Data Collection Time	N/A	Low	High	Low	High
Prior Knowledge Required	High	Low	High	Low	High
Safety/Accountability	High	Low	High	High	High

D.5 Discussion and Conclusion

In this work, we presented an alternative application of DRL: instead of directly deploying the AI policy, we aim to formulate control guidelines from the AI policy. As a result, we bypass major drawbacks of learned policies - unsafety, and stability issues - and obtain a certifiable, optimal controller that demonstrates human-like behaviours (Fig. D.1). These results were obtained for the challenging task of Push Recovery.

Results

We showed that DRL is powerful enough to learn complex motions that resemble those of humans. The learned policy demonstrated the same push recovery strategies that can be observed in humans, and exhibit similar robustness as state-of-the-art control algorithms. After analysing the learned AI policy we use it as a guideline and template for control design.

The engineered controller is able to reproduce the same strategies as human and AI policies with close quantitative fit to the collected data. As observed in humans [234], the policy minimises jerk and implements feedback control via Model-Predictive Control. Furthermore, analysis of the required torques and forces on the system show that the trajectories provided by the engineered policy are realisable on the real system.

We further compared the decoded DRL and human push recovery policies, and surprisingly found that the AI policy has strong similarity with human policies. We hypothesise that both the AI and humans are able to identify the key features in the problem, as required for high quality performance. This finding is interesting due to the time for the DRL agents to acquire human-comparable push recovery abilities: learning a good AI policy requires 6-8 hours; human infants require 10-18 months to learn the ability of locomotion [239].

While MJMPC was able to reproduce both policies, and both human and AI policy can be used as template model for control design, we found the usage of AI policies for template models more advantageous. This

was due to the data of DRL being immediately available - in contrast to the human data which required time-intensive post-processing -, and having direct access to the policy in form of a Neural Network allowing analysis of the Neural Network in addition to merely the input-output behaviour. We performed a t-SNE analysis on the AI policy (Fig. D.7), and found that the DRL agent - through interaction with the environment - decided to not use the Angular Momentum modulation due to its little effectiveness.

Challenges and Outlook

In this study, we showed that through analysis of template models from human and AI policies for push recovery a certifiable safe controller for humanoid robots can be engineered. Currently, all analysis and implementations were conducted in a high-fidelity physics-simulator which, despite our best efforts, differs from reality. To mitigate this problem, we proposed a control framework for real-world deployment that combined the proposed control law with a whole-body controllers in a cascaded manner. Due to the shown feasibility of the motions and the reliable stability of whole-body controllers in real-world applications, we anticipate a seamless deployment of the controller onto a real system.

Our future work will investigate approaches in directly bridging this reality gap and use interactions with the real environment to close this reality gap in the DRL process. Furthermore, we aim to apply this principled approach of designing controllers from template models to other systems (e.g., quadrupeds) and tasks (locomotion and manipulation).

High-Dimensional Bayesian Optimisation

E

In this chapter, an automatic parameter tuning process is presented that allows the tuning of high-Degree of Freedom (DoF) robots. To this end, we propose Alternating Bayesian Optimisation (ABO) [7] that is used to tune 36 parameters of the humanoid Valkyrie enabling robust locomotion over uneven terrain. The media to this chapter can be found in https://ieeexplore.ieee.org/abstract/document/8651385/media#media.

E.1 Introduction

For robot locomotion, control parameters are essential for the stabilization [240], Inverse Dynamics and whole-body control [241] of legged robots, e.g. humanoids [204, 242] and quadrupeds [243]. However, the high-dimensional and often sensitive parameters need to be correctly chosen to guarantee stability and good performance, which could be manually tuned or automatically found by search algorithms. The former is time consuming and suboptimal due to the correlation of high-dimensional parameters, while the latter requires sophisticated search subject to the high-dimensionality of the problem that may be sample-inefficient or often impossible.

The influence of parameters on the performance of a task cannot be directly computed, because the evaluation needs to be quantified from the interaction between the robot and the environment. Therefore, the objective function for evaluating the performance can be treated as a black-box, and derivative-free searching algorithms can be useful to determine the optimal parameters. With increasing dimensionality of the parameters, random or grid search approaches are inefficient as the parametric space increases exponentially, and thus the amount of evaluations. Derivative free searching algorithms, such as Sequential Model-based Algorithm Configuration [244], evolutionary algorithm [245], and particle swarm methods [246], are able to find a suitable

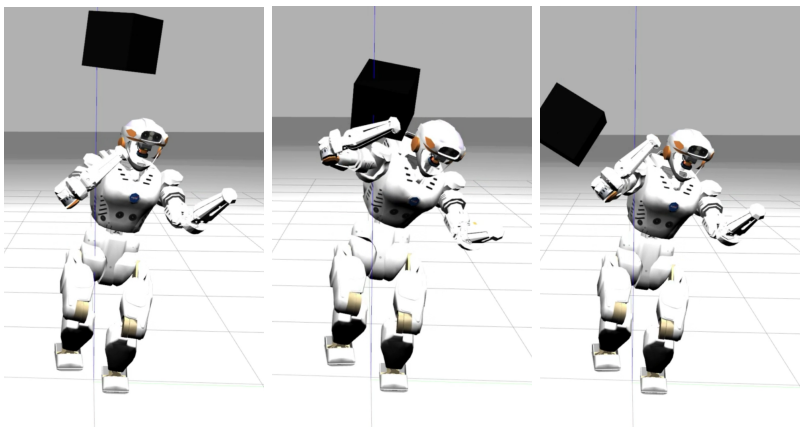


Figure E.1: Robust balancing against perturbations of NASA's Valkyrie using 36 *automatically* tuned control parameters.

set of parameters. However, they are not well suited for expensive evaluations due to their sample-inefficient nature.

Furthermore, automatic tuning of control parameters can be equivalently achieved by machine learning. Reinforcement learning can be used in robot control to tune optimal gait parameters [247], or to directly learn control policies for humanoid balancing control [171]. Alternatively, mathematical optimization can be effective in low dimensional problems such as optimizing the gait parameters considering the kinematics and dynamics constraints [248, 242].

Bayesian Optimization (BO) [19], a derivative-free, sample-efficient optimization algorithm, is suitable to find global optima for black-box optimization functions [42], and is widely used for hyper-parameter tuning [43]. In robotics, BO has been used to find gait parameters on real, physical hardware in [44] (5 dim./9 dim.), [45] (7 dim.), [46] (8 dim.), [47] (15 dim.). Additionally, domain knowledge has been applied to find suitable kernels for improving sample efficiency on hardware experiments [44, 48]. However, the dimensionality in these works was relatively low, and hence a direct implementation of BO performed well.

Despite its benefits, the capability of BO in finding global optima is limited by the high dimensionality, as the search space grows exponentially. Dimensionality reduction approaches, such as Principal-Component Analysis [49] or Partial Least Squares [50], would fail when the evaluation of the objective function is expensive, or the data is insufficient to find correlations in parameters. An approach to combine BO with LQR was suggested in [51], in which, rather than tuning the full state-feedback controller, only the lower dimensional LQR weights were tuned via BO. For humanoid Whole-Body Control, a Trial-and-Error learning algorithm in [52] dealt with model inaccuracies by learning repulsors to alter reference motions that prevents an unstable configuration of state space.

Recently, high-dimensional BO methods have been proposed: in [53] only a subset is optimized over by randomly dropping out parts of the parameters space; in [54] a local search is performed near a priorly given location. However, the work in [53] does not consider the correlation between the parameters and randomly drops out parameters, while the solution in [54] relies on prior knowledge of the optimum which is mostly unavailable without pre-tuning.

To overcome these limitations, we propose a novel approach that applies domain knowledge for partitioning the parameter space, and is able to find an optimum without assuming the location of the optimum. We adopt the idea of Alternating Optimization [55] and iteratively optimize one partitioned parameter sub-space at a time while keeping the rest of the parameter space fixed.

Our proposed algorithm has similarities with Coordinate Descent approaches [56]. However, instead of a sequential search along the coordinates until convergence, our proposed method searches in sub-hyper-boxes of the parameter space and is independent on the convergence of previous iterations. In combination with BO, a global optimization algorithm, the alternating nature reduces the risks of local minima and

allows parallelization, which are the two characteristics that are not provided by Coordinate Descent methods.

In this chapter, we show in simulation that the proposed algorithm is able to find an optimal hyper-parameter set for 36 parameters of a Whole-Body Quadratic Programming controller. As a further validation, our proposed algorithm is shown to be capable of finding the optima of 24 challenging objective functions from the COCO benchmarks [249]. Our contributions are summarised as follows:

- A novel Alternating Bayesian Optimization (ABO) algorithm that is able to optimize black-box objective functions with high-dimensional parameters.
- An automated parameter tuning framework for Whole-Body Control that can find the high-dimensional optimal parametric set from scratch within a few iterations.
- Evaluation of the versatility of the proposed ABO on the COCO benchmarking platform that shows consistent convergence of finding near global optima for challenging high-dimensional objective functions.

This chapter is structured as follows. First, the BO problem for whole-body control is formulated in Section E.2. Second, our novel ABO algorithm and the automated tuning framework are presented in Section E.3. The results of the whole-body control and benchmarking on COCO are analyzed in Section E.4, and the potential uses and possible limitations of the proposed algorithm are discussed in Section E.5. Finally, we conclude in Section E.6.

E.2 Bayesian Optimization

To find the maximum of an expensive-to-evaluate objective function, BO performs three steps. First, a cheap-to-evaluate, surrogate objective function is built as a Gaussian Process (GP). Second, an acquisition function maximises over the GP in order to find a local maximum. The point found by maximising the acquisition function is a candidate for the global optimum. Third, this point is sampled on the actual expensive-to-evaluate objective function. Observing a point at the suspected optimal points reduces (eliminates if noiseless) the variance of the GP at that point, and therefore refines the GP. By continuously iterating so, a global optimum of the actual objective function can be found [43]. The pseudo code is given in **Algorithm 7**. The sampled objective value y_i consists of the true objective function $J(x_i)$ and the noise ϵ_i. The notation $y_{1:T}$ indicates the samples gathered for y_i at time step $i = 1, ..., T$.

Gaussian Processes

A Gaussian Process is defined by a mean function $m(x)$ and a covariance function $k(x, x')$:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{E.1}$$

Algorithm 7: Pseudo code for Bayesian Optimization

$y_{1:T} \leftarrow J(x_{1:T}) + \epsilon_{1:T}$, sample T points
Initialise GP with $\mathcal{D}_{1:T} \leftarrow \{x_{1:T}, y_{1:T}\}$
for $i = 1, 2, ..., N$ **do**
  $x_t \leftarrow \text{argmax}_x a(x)$, get next query point
  $y_t \leftarrow J(x_t) + \epsilon_t$, sample query point x_t
  Update GP with $\mathcal{D}_{1:t} \leftarrow \{\mathcal{D}_{1:t-1}, (x_t, y_t)\}$

The prior mean $m(x)$ (not conditioned on data) is chosen to be a zero function $m(x) = 0$ [250]. For the choice of a covariance function $k(x, x')$, several kernels have been proposed [250]. Throughout this work, we used a Matérn kernel with the parameters* $\nu = 1.5$, $l = 1.0$:

$$k(x_i, x_j) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}d}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}d}{l} \right), \tag{E.2}$$

where $d = \|x_i - x_j\|$, gamma function $\Gamma(\cdot)$, modified Bessel function K_ν, and non-negative parameters ν, l.

Optimization Problem Formulation

The BO aims to find a set of parameters X that maximises the reward function $J(X)$, which keeps the objectives to be within a certain user-defined range. The reward function $J_{track}(X)$ for tracking performance is:

$$J_{track}(X) = r_C + r_F + r_T + r_P + r_H, \tag{E.3}$$

where the tracking objectives are Centre of Mass (COM) (r_C), foot (r_F), torso (r_T) and pelvis (r_P), and hand (r_H). Every tracking reward $r_X = \sum_{i=0}^{N} r_{x,i}$ consists of the sum of N rewards $r_{x,i}$. A reward $0 < r_{x,i} \leq 1$ at time step i for tracking a desired value x_{d_i} is given if the objective is in a certain range (determined by width κ):

$$r_{x,i} = \exp(-\kappa \|x_{d_i} - x_i\|^2). \tag{E.4}$$

The width $\kappa = -\ln(C)/\delta_{max}^2$ is calculated by the range δ_{max} and associated reward $C \doteq 0.001$ ($C \to 0$, because $C = 0$ and $\ln(0)$ are infeasible). Critical links, such as feet and COM have an error range $\delta_{max} = 0.05$ for both orientation in $[rad]$ and position in $[m]$ ($\kappa = 2760$, red curve, Fig. E.2). An error range $\delta_{max} = 0.1$ for torso, pelvis, and hands yields a width parameter $\kappa = 690$ (blue curve, Fig. E.2). The admissible error range is motivated by the physical shape of the support polygon, where $0.1m$ and $0.05m$ correspond to half of the foot length and width respectively, and the rotational errors approximately match the smallest torso joint limits.

A fall penalty is added if either the orientation of the torso θ_{rpy} exceeds

* While BO requires user-choices, e.g., kernel and acquisition function and related hyper-parameters, our ABO in this study is robust towards these choices. The same results were achieved by different acquisition functions (Expected Improvement, Upper Confidence Bound, Probability of Improvement), and kernels (Radial Basis Function, Matérn, Rational Quadratic) using the default parameters of scikit-learn.

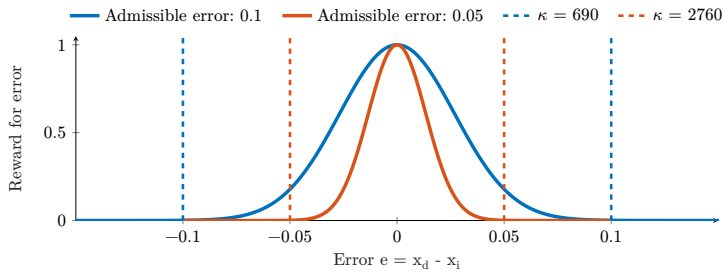


Figure E.2: Reward of error values for $\kappa = 690$ and $\kappa = 2760$.

a threshold $\delta_{rpy_{max}}$, or the pelvis height z_c is below a threshold δ_z:

$$J_{fall} = \begin{cases} 0, & \text{if } \theta_{rpy} \geq \delta_{rpy_{max}} \text{ or } z_c < \delta_z, \\ 1, & \text{else.} \end{cases} \tag{E.5}$$

The overall reward function is:

$$J(X) = J_{track}(X) \cdot J_{fall}. \tag{E.6}$$

Acquisition Function

The goal of an acquisition function $a(x)$ is to have a fast-to-evaluate function at any given point x in order to decide where to sample next for the observation (y_i, x_i). The next sample point is chosen as the point $x = \text{argmax}_x a(x)$ that maximises the acquisition function. The mean $\mu(x) = \mu(x; \mathcal{D}, \theta)$ and variance $\sigma^2(x) = \sigma^2(x; \mathcal{D}, \theta)$ are calculated from previous observations $\mathcal{D} = \{x_{1:n}, y_{1:n}\}$ and hyper-parameters θ of the GP. In this work, the upper confidence bound (UCB) [251] is used. It automatically trades off exploration and exploitation by weighing mean against variance:

$$a_{UCB}(x) = \mu(x) + \alpha\sigma(x), \tag{E.7}$$

where the trade-off parameter $\alpha \geq 0$. The UCB for maximisation can be seen as an algorithm that minimises the accumulated regret $R_T = \Sigma_{t=1}^{T} f(x^*) - f(x_t)$ with the unknown-optimal point x^* to a point of no-regret [252]:

$$\lim_{T \to \infty} R_T / T = 0. \tag{E.8}$$

E.3 Dimensionality Reduction Techniques

This section presents the principles for reducing the dimensionality of the optimization variables. First, we elaborate our proposed Alternating Bayesian Optimization (ABO) approach for finding high-dimensional parameters using a pseudo algorithm. Next, we introduce domain knowledge such as symmetry to reduce the dimensionality of parameters from more than 100 down to 36, and then group these 36 parameters by their correlation.

Core Concept and Formulation of the Algorithm

We propose a novel method (**Algorithm 8**) of finding the global optimum for the whole set of optimization variables X. We partition $X = [X_1, ..., X_K]$ into K groups, and successively optimize the objective function (E.6) on each partition X_j $(j = 1, ..., K)$ while keeping the other $K - 1$ partitions fixed. We define the crossed-out notation $\cancel{X_i}$ to indicate that the parameter group X_j $(j = 1, ..., K)$ is fixed.

First, an initial set of parameters X_0 will be used for all optimization variables with maximum value $y_{max} = J(X_0)$. Next the objective function $J(X)$ will be optimized via BO with all parameter groups being fixed except X_j. If using X_j results in a better value than the current maximum value y_{max}, then X_j will be used and y_{max} will be updated.

After iterating through all K groups for N times or when a termination criterion is met, a final BO step with N_{final} iterations is performed to locally optimize around the near optimal parameter set. This step aims to either fine-tune the parameter X_{sub} (if $J(X_{final}) - J(X_{sub}) < \delta$), or unstuck the optimization (if $J(X_{final}) - J(X_{sub}) > \delta$).

In **Algorithm 8**, ABO is used to find near-optimal parameters, and the final holistic BO will either globally fine-tune or unstuck the local optimum. For large variations between ABO (**Algorithm 8**, line 2) and holistic BO (**Algorithm 8**, line 9), which indicates a local maximum, the algorithm will be restarted with new initialisation $X_0 = X_{final}$.

Algorithm 8: Pseudo code for Alternating Bayesian Optimization

```
1  X ← X_0, y_max ← J(X_0)
2  for i = 1, 2, ..., N and not terminate do
3      for j = 1, ..., K do
4          X_j^+ ← argmax_{X_j} J(X_1, ..., X_j, ..., X_K)
5          if J(X_1, ..., X_j^+, ..., X_K) > y_max then
6              X ← [X_1, ..., X_j^+, ...X_K]
7              y_max ← J([X_1, ..., X_j^+, ..., X_K])
8  X_sub ← X
9  for i = 1, 2, ..., N_final do
10     X_final ← argmax_X J(X)
11 if J(X_final) − J(X_sub) > δ then
12     Restart ABO with X_0 ← X_final
```

Reduction of Dimensionality for Whole-Body Control

The Quadratic Programming (QP) based Whole-Body controller optimizes physically feasible torques for tracking task-space references. The task priorities are represented by the weights as $w = [w_0, ..., w_n]$ in the

objective function of the whole-body optimization problem[†] (E.9). The whole-body QP in [204] is adopted and the tasks J_{tasks} are:

$$J_{task} = \frac{1}{2}\|AX - b\|^2, \tag{E.9}$$

where $A = [w_0 A_0, \cdots, w_n A_n]^T$, $b = [w_0 b_0, \cdots, w_n b_n]^T$, and the optimization variable $X = [\ddot{q}, \tau, \lambda]^T$ consisting of torque commands τ, joint accelerations $\ddot{q}$ and ground reaction forces λ. Rearranging (E.9) leads to the QP form:

$$\min_X X^T H X + f^T X \tag{E.10}$$

$$s.t. \quad A_{eq} X + B_{eq} = 0 \tag{E.11}$$

$$A_{ineq} X + B_{ineq} \geq 0. \tag{E.12}$$

The cost function (E.10) is a weighted sum over tracking objectives and regularization (cf. Table H.1). The Cartesian reference acceleration for tracking COM and body link trajectories is calculated from the desired position x_d, velocity $\dot{x}_d$, and acceleration $\ddot{x}_d$ via a PD law as:

$$\ddot{x} = \ddot{x}_d + K_P(x_d - x) + K_D(\dot{x}_d - \dot{x}). \tag{E.13}$$

The equations of motion form the equality constraints (E.11):

$$\begin{bmatrix} M(q) & -S & -J^T(q) \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \tau \\ \lambda \end{bmatrix} + h(q, \dot{q}) = 0, \tag{E.14}$$

with inertia matrix $M(q)$, selection matrix S, stacked Jacobian matrices $J^T(q)$ of the contact links, and nonlinear effects $h(q, \dot{q})$. Torque limits, friction constraints, and COP constraints are considered in the inequality constraints (E.12) and are implemented as proposed in [253].

Optimization Variables

By assuming symmetry between the left and right and grouping symmetric links, the optimization variables can be reduced to 36 (Table H.1). As shown in Table H.2, H.3, holistically optimizing 36 parameters altogether for the whole-body control yields suboptimal results, and is thus impractical.

We will show that grouping the optimization variables into three physically meaningful groups will result in faster convergence and better performance. The parameters $X = [X_W, X_{PD}, X_M]$ are categorised into objective function weights X_W (E.9), PD gains X_{PD} (E.13), and miscellaneous parameters X_M. Here, the miscellaneous set X_M describes the parameters required for the wrench constraints as in [253], including the foot geometry x_{front}, x_{back}, y_{left}, y_{right} with $y_{side} = y_{left} = y_{right}$, and friction constraints μ. Instead of using a constant foot size in X_M, the foot geometry can be treated as a tunable parameter for finding

[†] The priority is determined via ABO. The reward function (E.6) is designed such that the weights will be optimized to keep the tracking error within the user-specified error range and as close to zero as possible.

a suitable stability margin, where boundary limits are the actual foot dimensions of the robot. We found that leaving a stability margin for the actual foot geometry and X_M leads to higher robustness. Averaged over 10 trials, a stability margin of $2cm$ was found.

E.4 Results

This section presents three key results: 1) the ability of ABO to find optimal parameters for whole-body control of the Valkyrie robot; 2) comparison of ABO with three other parameter search algorithms; 3) a further evaluation study of ABO on the COCO benchmarking suite [249].

Comparison Methodology

ABO is compared with three other parameter search algorithms: holistic BO, alternating random search, and BO using dropout [53]. The holistic BO approach optimizes over the whole parametric space (**Algorithm 7**), while ABO iteratively optimizes over its sub-spaces (**Algorithm 8**). The alternating random search method is similar to our ABO approach, but the next sample point is uniformly and randomly sampled from the search space instead of being found by an optimized acquisition function.

BO using dropout achieves dimensionality reduction by randomly dropping out dimensions and optimizing over d parameters instead of the full, high-dimensional parameter space. The work in [53] proposed two fill-in strategies for the dropped dimensions: dropout-random and dropout-copy. The third method, dropout-mix, performed similar to dropout-copy and is not shown in our comparison (Fig. H.1b) for clarity purposes. We implemented both fill-in strategies with a sampled dimension of $d = 8$ and $d = 16$.

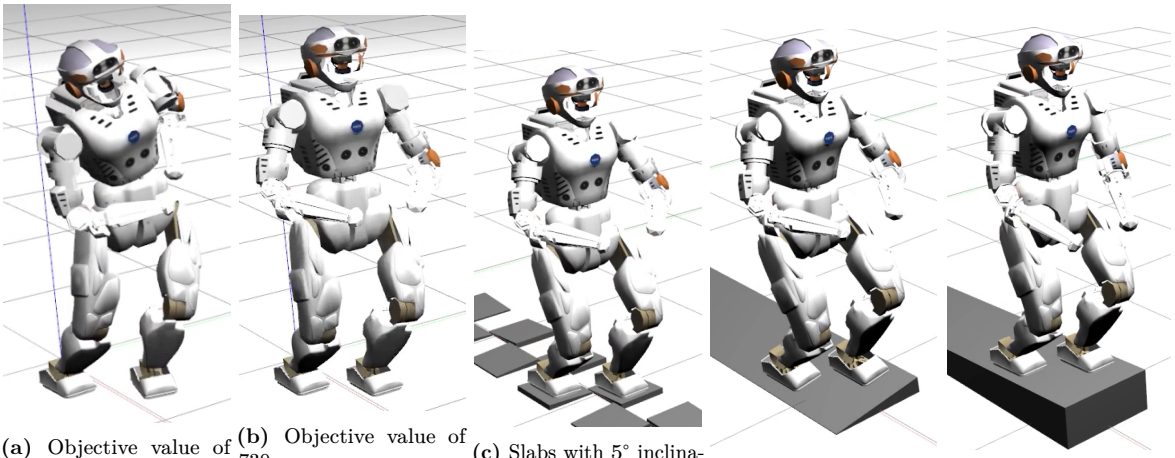
For ABO, every BO iteration consists of 30 BO evaluations with UCB as acquisition function $a_{UCB}(x)$ using the trade-off parameter $\alpha = 3$. For the low-dimensional parameters X_M, only 10 BO evaluations are conducted. Thus, for the whole-body control, 70 objective function evaluations are conducted per iteration (30 for X_W, X_{PD}, 10 for X_M). The holistic, random search, and dropout algorithms use the same evaluation budget (70 per iteration) as ABO.

Optimizing Hyper-parameters for Whole-Body Control

This section presents the results obtained from learning optimal parameters by interacting with the environment. All simulations were conducted in Gazebo using an accurate model provided by NASA for the humanoid Valkyrie [10] - a $1.80m$ tall, $139kg$ heavy humanoid robot with 44 actuated Degrees of Freedom (DOF).

An episode consists of a start and an end phase of $1s$ each and 5 straight steps of $0.25m$ with a step duration of $3s$. The walking motion

is generated by using Model-Predictive Control as in [4] to track these pre-planned footsteps for both Gait Planning and Feedback Control. At a frequency of $50Hz$, the sum of 850 rewards (E.4) are gathered per episode as the scalar output of the reward function. To the best of our skills, the hand-tuned parameter set yielded a value of 726. At a theoretical maximum of 850 for absolutely perfect tracking, a value of over 700 is able to robustly perform all locomotion tasks (Fig. E.3b). As a baseline, walking all 5 steps using bad tracking is possible for values over 550, and standing is possible for values over 250 (Fig. E.3a). The objective weight and PD gain parameters for the alternating approaches are initialised (alg. 8, line 1) with zeros $X_W = X_{PD} = 0$, and the miscellaneous parameters are initialised using their mean values $X_M = [0.09, 0.04, 0.03, 0.5]$.



(a) Objective value of 250 (poorly tuned, 1st iteration).

(b) Objective value of 730 (well tuned, 6th iteration).

(c) Slabs with 5° inclination (well tuned).

(d) Roll inclination of 10° (well tuned).

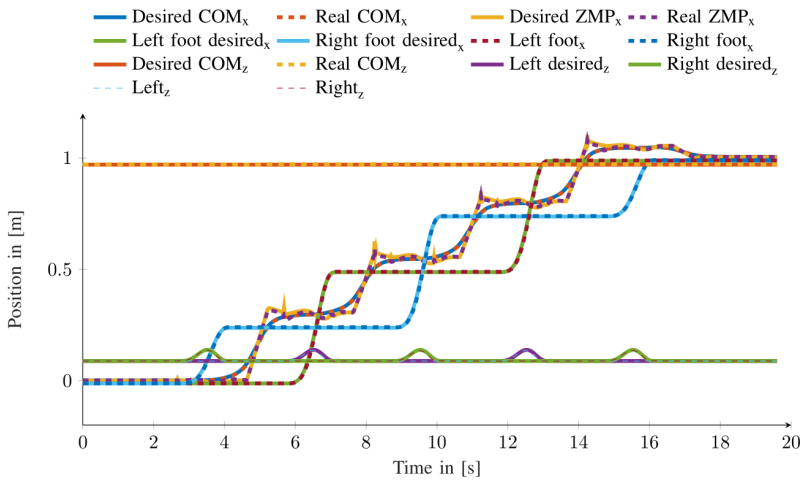
(e) Pitch inclination of 5° (well tuned).

Figure E.3: Snapshots of different walking trails using automatically tuned parameters. The detailed motions and scenarios can be found in the accompanying video.

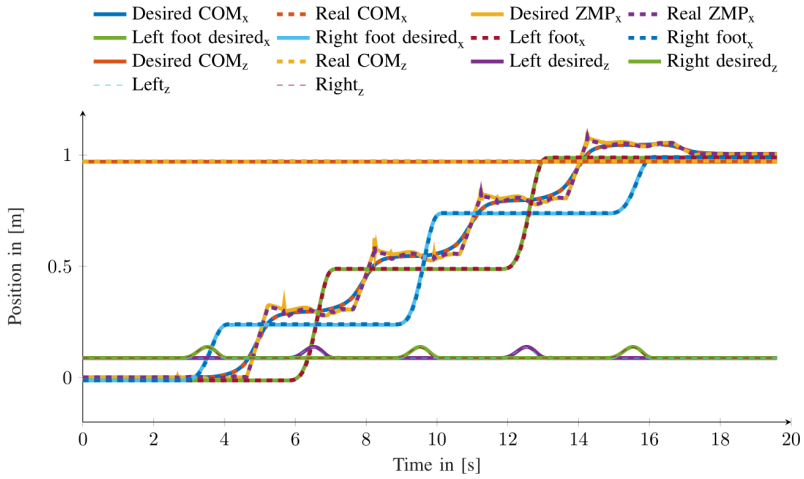
On average, optimal parameters were found within 6 iterations via ABO (alg. 8). These parameters were generalised to multiple locomotion and balancing tasks and exhibit good impedance behaviour of walking over unmodelled, uneven terrain (cf. supporting video, Fig. E.1, Fig. E.3c-e). The tracking performance of task space references is shown in Fig. E.4.

The learning curves of the automatic parameter tuning are in Figure H.1. The best values indicate the maximum value at the respective time step and therefore only shows improving values. The mean of this curve shows that the robot starts to walk after roughly 2 ABO iterations (140 evaluations), achieves a manual-tuned level after 4 ABO iterations (280 evaluations), and performs the final mean value of 732 after 8 ABO iterations (560 evaluations).

Table H.2 shows the number of iterations and the final value at convergence from four different methods. Over 10 trials, ABO always found a parameter set for task completion. The alternating random approach has one trial finding such a parameter set, but the holistic approach has no trial of finding a working parameter set. BO using dropout-copy was able to find a task-completing parameter set for all 10 trials with both $d = 8, d = 16$, while BO using dropout-random was not able to



(a) Tracking task space references in x coordinate.



(b) Tracking task space references in y coordinate.

Figure E.4: Tracking of COM, COP and feet trajectories, where the yellow solid line is the online re-generated desired ZMP consists of the nominal ZMP and the feedback corrections.

find a working parameter set. The results in Table H.2 suggest that the nature of alternating search of parameters for high-dimensional problems greatly improves the success of finding good parameters even for non-sophisticated search algorithms such as random search.

Both ABO and dropout-copy for $d = 8$ and $d = 16$ can achieve an objective value over 550 indicating that the robot is able to walk stably (Figure H.1b). However, ABO converges faster and achieves a higher value than the dropout-copy, and thus tracks the reference trajectories better. Dropout-random achieves similar performance as the alternating random search, which is not able to succeed a stable 5-step walking task.

Validating Versatility of ABO by COCO Benchmarks

To further understand the versatility of solving other problems, ABO was validated on the COCO benchmarking suite that has 24 functions (f1-f24) in an explicit form for benchmarking global optimizers in a black-box setting. Notably, in addition to standard objective functions

(f1, f5, f13), it also contains ill-conditioned (f2, f6, f10, f11, f12, f18), local minimum trapping (f3, f4, f7, f8, f9, f14, f15, f20, f23), irregular (f16, f21, f22), and multi-modal (f17, f19, f24) objective functions to thoroughly test the optimizer.

For normalization, every objective function is off-setted such that the smallest value is larger than zero. The smallest possible value varies for every objective function $f1 - f24$ and increases with the dimension of the problem. In addition to a maximal number of iterations, tolerances $t_1 = 1\%$ and $t_2 = 10\%$, representing the tolerance of being 1% and 10% away from the maximal possible value, are used as termination criterion calculated individually for each function $f1 - f24$. The average maxima for the well and poorly conditioned objective functions, and the success rate r_{t_1}, r_{t_2} of using tolerance rate t_1 and t_2 are described respectively in Table H.3. All parameters are uniformly randomly initialised, and the optimization variables are partitioned randomly.

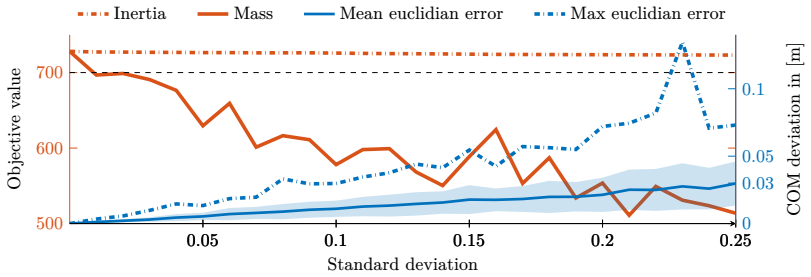
The lower success rates from the alternating random search suggest that the higher success rates of ABO is attributed to our proposed method, rather than randomly finding the optimum.Furthermore, similar to the whole-body control ABO case, using an alternating approach increases the success rate over a holistic one, especially for high-dimensional problems. In contrast to [249], an evaluation budget is used. This is due to the fact that a limitless evaluation budget would lead to lengthy computations, as well as an exhaustive search that eventually leads to the optimum. The COCO benchmark, in which budget is considered as secondary, is designed for being challenging to "defeat" the optimization problem. We aim to preserve the sample efficient nature of BO, by trading off an absolute success rate of the ABO approach that can arguably be improved. In fact, the success rate of ABO being 2-3 times higher than that of holistic BO has already shows an advantage. In summary, a holistic BO approach yields good results for dimensions up to 10D but low success rates for the dimensions higher than 10D, whereas the ABO approach yields good results of success rates over 75%.

E.5 Discussion

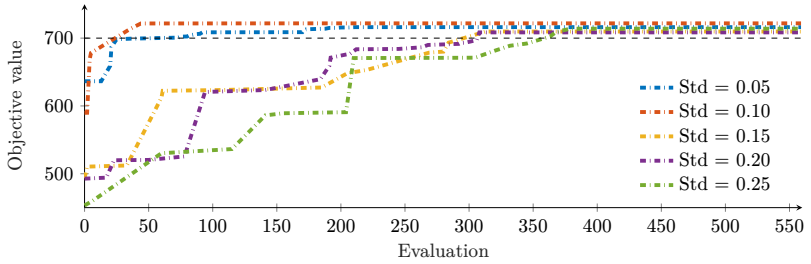
This section discusses the potential use of our proposed ABO algorithm and the consideration of deploying ABO on real systems regarding the simulation-to-reality gap.

Reality Gap between Simulation and Real World

Model uncertainties hinder a direct transfer of the tuned parameters from simulation to reality, and are mainly caused by the inaccuracy of the masses, inertias, link dimensions, communication latency, and noises. The following study shows the robustness of our automatically tuned parameters against model uncertainties, and suggests the need of a small number of fine-tuning iterations on the real system given an initial set of parameters auto-tuned from a perfect model.



(a) Resulted reward due to model imperfection on mass and inertia.



(b) Learning curve for fine-tuning on realistic model.

Figure E.5: Reward from different model discrepancies and the resulted learning curves for fine-tuning the parameters.

Model Imperfections

The imperfection was modelled by the differences between model and real robot in mass and inertia. We first ran ABO on a perfect model in simulation, and then for the test, we altered the simulation model by adding uncertainties of masses and inertia from a normal distribution with varying standard deviations.

In Figure E.5a, the results averaged over 5 trials suggest that it is important to accurately identify mass as it contributes greatly to the reward and causes COM deviations (blue line, Fig E.5a) of up to 3cm on average. In contrast, wrong inertia identification contributes much less to the reality gap. Inaccuracies of up to 5% can be tolerated without losing tracking quality, and even inaccuracies of up to 25% still yield stable gait despite bad tracking.

Fine-tuning of Parameters on Real Hardware

After obtaining a parameter set that works well in simulation, but exhibits bad tracking performance on the real model, ABO has the potential to be further used to fine-tune parameters on the real robot, which is studied in simulated cases here (not on the real hardware) by using a different model with intentional changes mimicking discrepancies between simulation and hardware. From the results in Fig. E.5b, it can be seen that for small model errors, ABO converges after one iteration and 50 function evaluations; for a 25% discrepancy as the reality gap, the algorithm requires 6 ABO iterations (360 evaluations) to reach optimal behaviour.

Potential and Possible Applications of ABO

In addition to tuning whole-body control, ABO could also identify and tune additional parameters by considering model asymmetries between the left and right, using independent gains in x, y, z coordinates, and identifying dynamics and off-diagonal elements in the weight matrices (E.10). In the following we depict potential applications of ABO.

Consideration of Asymmetries

In Table H.1 symmetry between left and right side of the robot is assumed, which may not be true in reality. In Section E.5, we showed that the optimized parameters were robust to uncertainties including model asymmetry, and additional fine-tuning can further improve the performance. Alternatively, the model asymmetry could be directly included in the optimization setting by adding more optimization variables. As a proof of concept, we were able to achieve the same performance as before by adding 4 additional tuning parameters for the left and right foot separately.

Separate Parameters for Postural Control

In addition to physical asymmetries, the body pose can also be decomposed into 6 components with separate weights requiring longer training time due to higher dimensionality. We tested separate horizontal (x, y) and vertical (z) gains for the feet and COM, and obtained similar results as in the original non-separating case by an average increase of training time of 100 evaluations.

Adaptation to Variation of Dynamics Properties

In Section E.4, we exemplarily showed the possibility of identifying friction parameters. Identification of more parameters could be conducted on the dynamics parameters. To this end, ABO would optimize over masses and inertia matrices. We used ABO to successfully identify the 8 heaviest links and the inertia matrix of torso with a deviation of up to 10% from the nominal values specified in the URDF.

Identification of Correlated Off-diagonal Elements

Perceiving correlations of the off-diagonals in manual tuning is difficult for humans, and thus ABO could potentially be a leverage to identify those elements in the cost function (E.10). Future work is needed to study ABO's ability to find suitable parameters given a much larger number of additional optimization variables.

E.6 Conclusion

In this work, we proposed an Alternating Bayesian Optimization (ABO) algorithm capable of tuning optimal parameters for high-dimensional optimization problems. This was achieved by evaluating data from interactive trials of between the robot and the environment (simulated scenarios) and the search of appropriate hyper-parameters, which produced optimal performance of robust locomotion under model uncertainties. We tackle the arising problems in high-dimensionality, such as sample-inefficiency and exhaustive searches, by partitioning the whole parameter space into low-dimensional sub-spaces, and iteratively optimizing over each sub-space while fixing the rest sub-spaces.

We first applied dimensionality reduction to reduce 100 more parameters to 36, and then used the proposed ABO algorithm to automatically tune this 36-dimensional parameter set for whole-body control of NASA's Valkyrie robot to locomote over uneven terrains. The robot stood stably after 1 iteration, performed dynamic walking after 3 iterations, and converged to the best performance within 6 iterations.

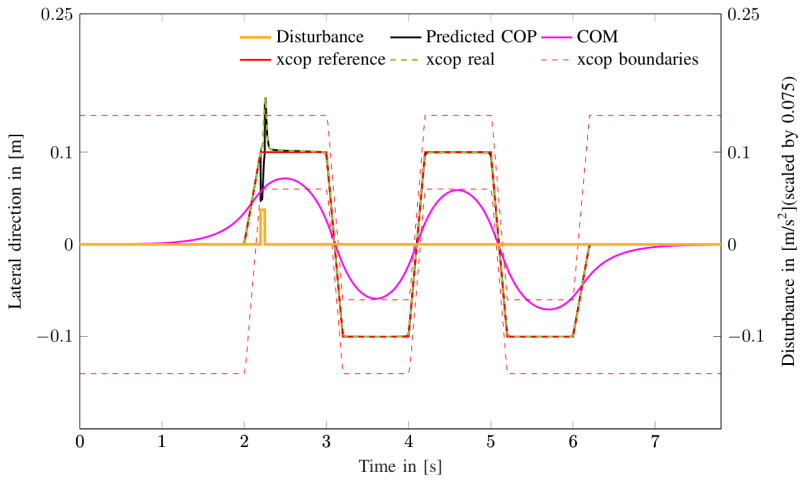
The proposed method found better parameters within fewer iterations than the manual tuning from the experienced researchers. This is mainly due to human limitation in spotting correlations between parameters in high dimensions, whereas ABO utilises Gaussian Processes which are precisely designed to capture these correlations. Hence, our proposed algorithm can be applied to other systems that require automatic tuning of high-dimensional, correlated hyper-parameters. In the COCO benchmarking suite, the proposed ABO algorithm was further validated by finding global optima of the challenging objective functions. Lastly, we discussed potential applications and limitations of the algorithm that may arise particularly during the transfer from simulation to reality.

Our future work will focus on implementing ABO on real hardware for automatic gait tuning by using the parameters obtained from simulation as an initialisation. Furthermore, constrained Bayesian Optimization methods, such as SafeOpt [254], will be considered to protect the robot from damage.

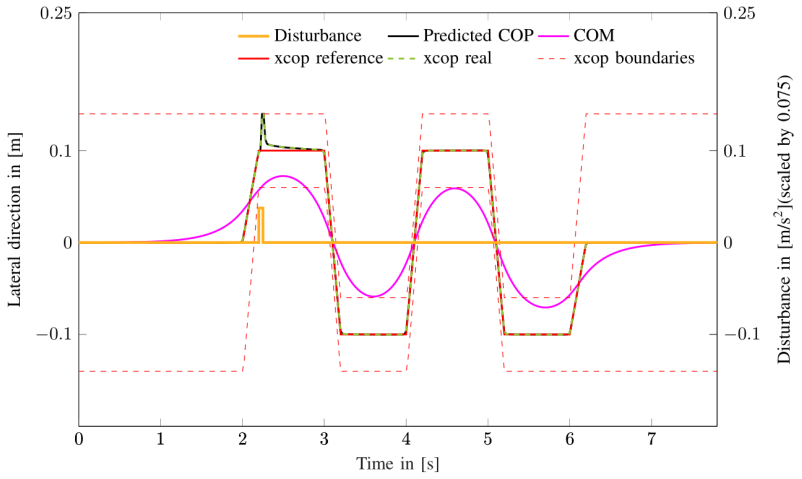
Stability Control

F

Following are the comparison figures for Section A.5.

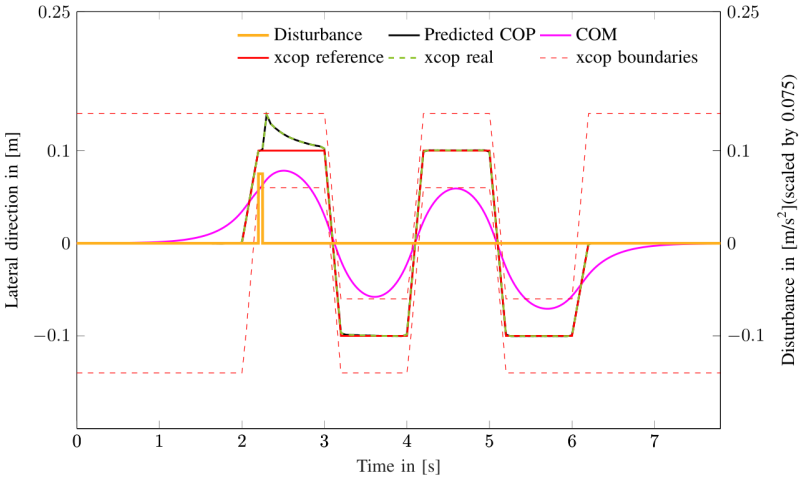


(a) Impulse disturbance while COP tracking with formulation E.

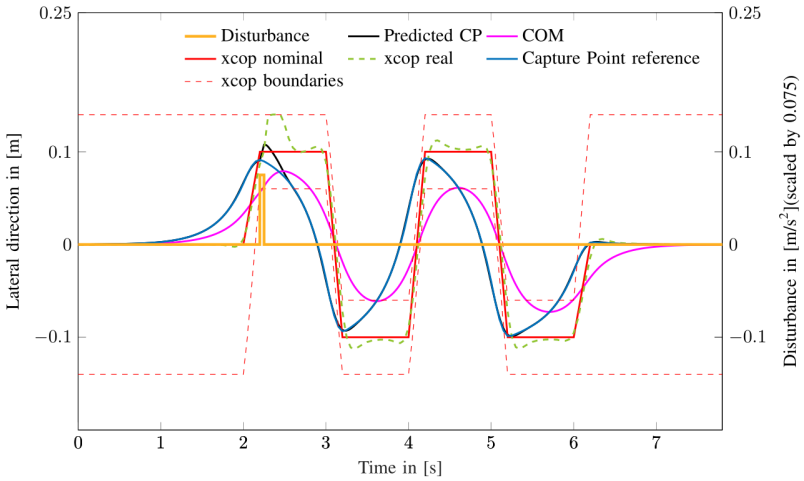


(b) Impulse disturbance while COP tracking with proposed formulation D.

Figure F.1: Comparison between formulation E (a), and proposed D (b) for an impulse push of $\ddot{x}_{ext} = 0.5m/s^2$. Due to wrongly capturing the COP dynamics the controller in (a) is wrongly reacting to the disturbance, moving the COP outside of the SP. Formulation (b) is correctly complying with the ZMP constraint.

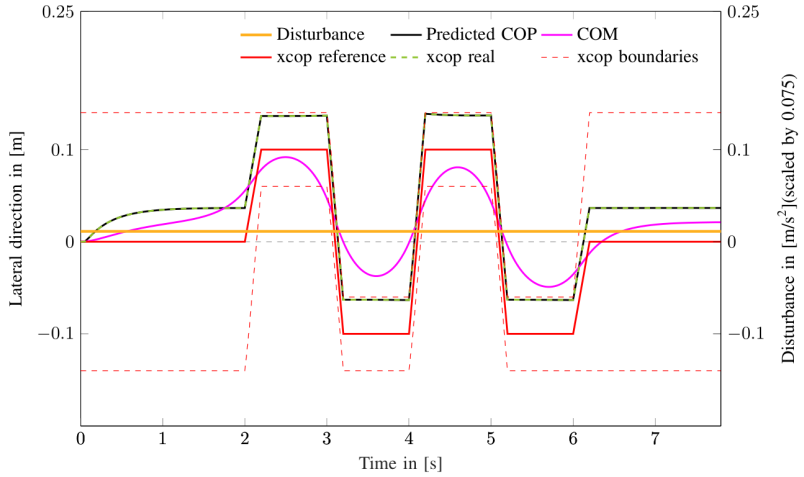


(a) COP tracking with proposed formulation D.

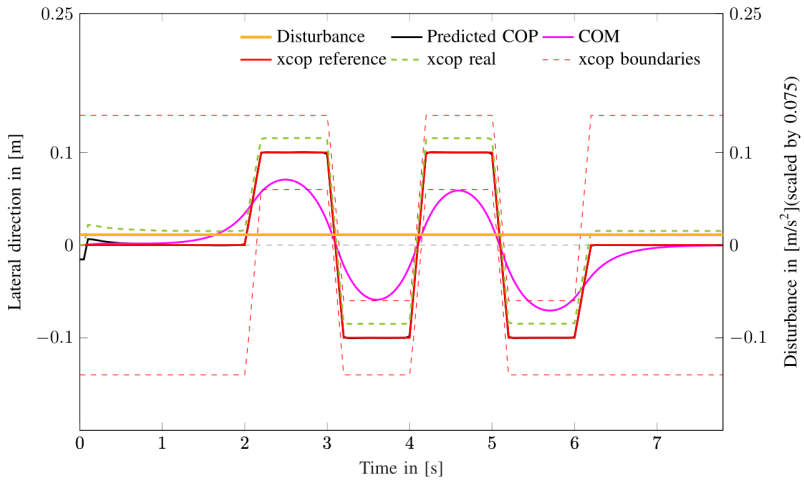


(b) CP tracking with proposed formulation D.

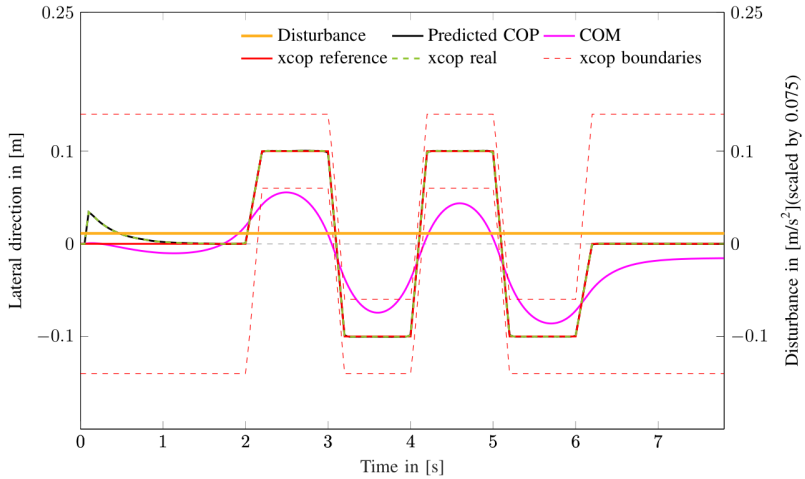
Figure F.2: Comparison between COP (a) and CP (b) tracking under an impulse disturbance of $\ddot{x}_{ext} = 1m/s^2$.



(a) Constant disturbance while COP tracking with formulation C.



(b) Constant disturbance while COP tracking with formulation E.



(c) Constant disturbance while COP tracking with proposed formulation D.

Figure F.3: Comparison between formulation C (a), E (b), and proposed D (c) for constant disturbance $\ddot{x}_{ext} = 0.15m/s^2$. The proposed formulation D updates the COP trajectory correctly according to (A.8). Formulation E calculates the COP trajectory wrongly by (A.2), attributing the external acceleration wrongly to the COP.

Accelerated Deep Reinforcement Learning

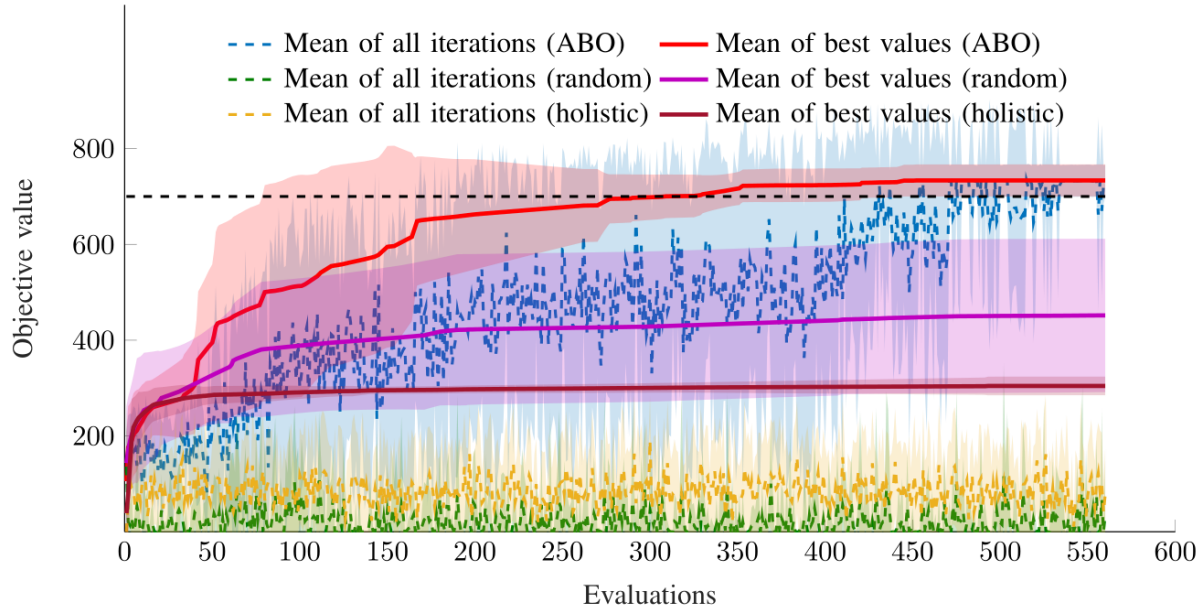
G

Table G.1: Wall clock times for various robotic environments trained with PPO and SAC

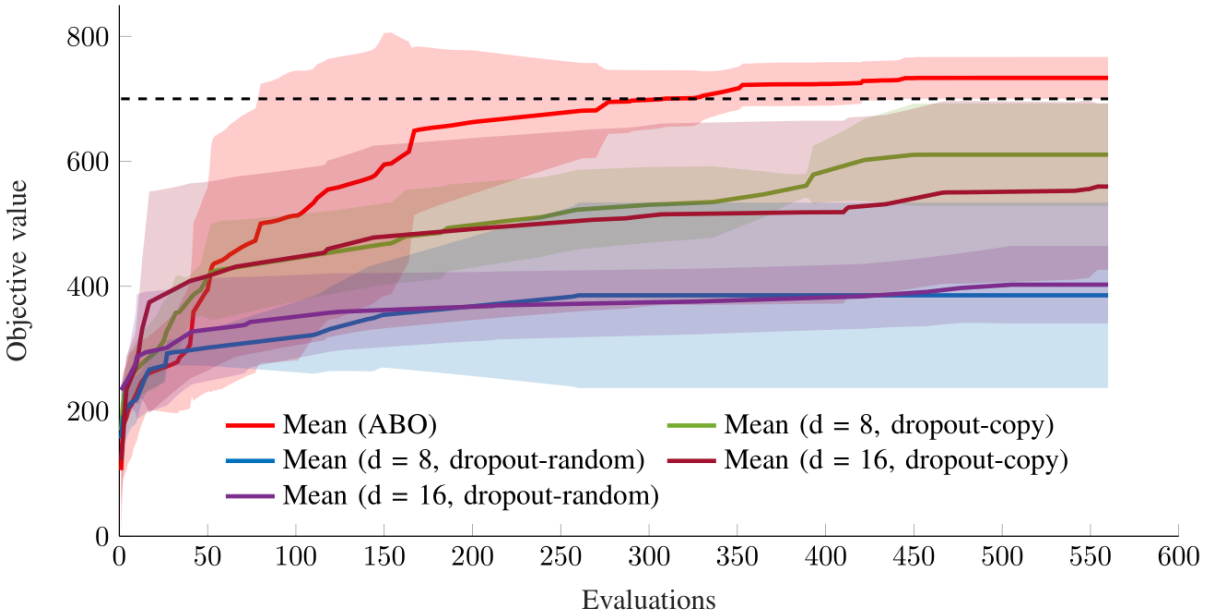
	Real-Time Factor (RTF)			PPO for 200K samples			SAC for 200K samples		
	SAC RTF	PPO RTF	Max RTF	Total time [s]	Train [s]	Roll-out [s]	Total time [s]	Train [s]	Roll-out [s]
Ant	323	41	600	**214**	19	195	1255	1230	25
HalfCheetah	313	52	1200	**172**	17	155	1250	1224	26
Hopper	333	54	700	**178**	17	161	1256	1232	24
Walker	278	50	800	**168**	19	149	1229	1200	29
ANYmal	152	5	202	1666	47	1619	**1276**	1224	53
Valkyrie	91	4	120	2309	49	2260	**1328**	1240	88

High-Dimensional Bayesian Optimisation

H



(a) ABO, alternating random search, and holistic BO comparison.



(b) Comparison between ABO, BO using dropout-random ($d = 8, 16$), and BO using dropout-copy ($d = 8, 16$).

Figure H.1: Comparison between ABO, alternating random search, holistic BO, and BO using dropout for Whole-Body Control (shaded area: standard deviations).

Table H.1: Range for the optimization variables. For pose tracking, separate weights for position and orientation are used.

Objective weight	Objective	Range x in $[10^x]$	PD gain	Controlled link	Range x in $[10^x]$	Misc. param.	Range
O0	Track COM	[0]	C1-C2	PD COM position	$[-1, 3]$	x_{front}	$[0, 0.18]$
O1-O2	Track torso pose	$[-3, 3]$	C3-C4	PD COM orientation	$[-1, 3]$	x_{back}	$[0, 0.08]$
O3-O4	Track pelvis pose	$[-3, 3]$	C5-C6	PD foot position	$[-1, 3]$	y_{side}	$[0, 0.06]$
O5-O6	Track Hand pose	$[-3, 3]$	C7-C8	PD foot orientation	$[-1, 3]$	μ	$[0.2, 0.8]$
O7-O8	Track Foot pose	$[-3, 3]$	C9-10	PD torso/pelvis position	$[-1, 3]$		
O9-O10	Track COP & weight dist.	$[-3, 3]$	C11-C12	PD torso/pelvis orientation	$[-1, 3]$		
O11	Joint angle q	$[-3, 3]$	C13-C14	PD hand position	$[-1, 3]$		
O12-O16	Reg. $\tau, \ddot{q}, \dot{\tau}, \dot{\lambda}$	$[-3, 3]$	C15-C16	PD hand orientation	$[-1, 3]$		

Table H.2: Average number of iterations and final values over 10 trials for Whole-Body Control of NASA's Valkyrie. The values in parenthesis indicate the number of evaluation.

Prob. dim.	Opt. dim.	Mean iter.	Mean max.	Median iter.	Median max.	Best iter.	Best max.	Worst iter.	Worst max.	Method
36	36	8 (560)	304	8 (560)	299	8 (560)	360	8 (560)	289	Holistic BO
36	16	**5 (350)**	**734**	**6 (418)**	**737**	**2 (84)**	**779**	**8 (560)**	**665**	Alternating BO
36	16	8 (560)	452	8 (560)	448	8 (560)	662	8 (560)	259	Alternating Rand.
36	16	8 (560)	560	8 (560)	605	8 (560)	676	8 (560)	330	Dropout-copy, $d = 16$
36	16	8 (560)	403	8 (560)	384	8 (560)	470	8 (560)	330	Dropout-random, $d = 16$
36	8	8 (560)	611	8 (560)	584	8 (560)	742	8 (560)	543	Dropout-copy, $d = 8$
36	8	8 (560)	386	8 (560)	300	8 (560)	440	8 (560)	294	Dropout-random, $d = 8$

Table H.3: Average number of iterations and final values over 10 trials for well and poorly (f2, f7, f10, f11, f12, f18, f22) conditioned optimization problems. The random search is averaged over 1000 trials.

Prob. dim.	Opt. dim.	Max. value (good cond.)	Max. value (bad cond.)	Success rate r_{t_1} (opt.)	Success rate r_{t_2} (opt.)	Success rate r_{t_1} (rand.)	Success rate r_{t_2} (rand.)	Method
40	40	$6 \cdot 10^5$	$7 \cdot 10^{10}$	0.12	0.13	0.00	0.12	Holistic
20	20	$3 \cdot 10^5$	$8 \cdot 10^9$	0.13	0.24	0.09	0.16	Holistic
10	10	$3 \cdot 10^5$	$7 \cdot 10^9$	0.54	0.80	0.15	0.40	Holistic
5	5	$2 \cdot 10^5$	$3 \cdot 10^8$	0.64	0.88	0.26	0.46	Holistic
40	16	$6 \cdot 10^5$	$7 \cdot 10^{10}$	0.13	0.70	0.13	0.32	Alternating
40	10	$6 \cdot 10^5$	$7 \cdot 10^{10}$	**0.21**	**0.75**	0.13	0.38	Alternating
20	10	$3 \cdot 10^5$	$8 \cdot 10^9$	**0.27**	**0.74**	0.12	0.39	Alternating
10	5	$3 \cdot 10^5$	$7 \cdot 10^9$	**0.59**	**0.88**	0.31	0.48	Alternating

Decoding Motor Skills of AI and Human Policies

I

Table I.1: Mechanical specifications of Valkyrie.

	Arm				Torso			Leg					
	Shoulder roll	Shoulder pitch	Shoulder yaw	Elbow pitch	Torso roll	Torso pitch	Torso yaw	Hip Roll	Hip pitch	Hip yaw	Knee pitch	Ankle Pitch	Ankle Roll
Lower joint position $[rad]$	-1.3	-2.9	-3.1	-2.2	-0.2	-0.1	-1.3	-0.6	-2.4	-0.4	-0.1	-0.9	-0.4
Upper joint position $[rad]$	1.5	2.0	2.2	0.1	0.3	0.7	1.2	0.5	1.6	1.1	2.1	0.7	0.4
Joint velocity $[rad/s]$	5.9	5.9	11.6	11.5	9.0	9.0	5.9	7.0	6.1	5.9	6.1	11.0	11.0
Joint torque $[Nm]$	190	190	65	65	150	150	190	350	350	190	350	205	205

Multi-Expert Learning of Adaptive Locomotion Behaviours

J

Following the Supplementary Notes to Chapter 4 of our publication [2] are attached.

Science Robotics
AAAS

Supplementary Materials for

Multi-expert learning of adaptive legged locomotion

Chuanyu Yang*, Kai Yuan*, Qiuguo Zhu, Wanming Yu, Zhibin Li†

†Corresponding author: zhibin.li@ed.ac.uk

This PDF file includes:

Other supplementary files for this manuscript include:

Notes

Note S1. Data analysis of MELA learning results
From fig. S12A, the changing weights around the boundaries of locomotion modes indicate that MELA produced smooth and quick transitions across successive modes. The data in fig. S12C-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 S12C 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. S12D, where the sum of expert 5, 6 and 8 has particularly high peaks during fall recoveries but constantly low in other modes. Figure S12E 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.

In conclusion, the data in fig. S12 reveals the correlations and patterns between the weights of collaborating experts and the corresponding locomotion mode, suggesting that trained experts in the MELA were activated in a coherent manner.

Note S2. Additional Materials and Methods
In this section, we provide additional details of the training procedures and the control framework that are needed to reproduce our presented policies.

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.

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 (note S2).

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 (*59*, *60*). 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 (Fig. 6A). A policy with an update frequency of around 25 Hz is a common setting as a high-level motion planning layer in robot control (*27*, *28*, *31*, *61*). 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 S6. 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 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. The supplementary video (movie S2) shows that the robot interacted in a compliant manner, instead of being stiff as it would be in a pure position control mode with the same PD gains.

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) (*48*). 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 (*62*, *63*). 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.

Nomenclature
This note describes the definitions of mathematical notations to help explain the equations of the reward terms in the following section.

ϕ^{base}	Orientation vector: the projection of the normalised gravity vector in the robot base frame to represent the orientation
h^{world}	The robot base height (z) in the world frame
v^{world}_{base}	The linear velocity of the robot base in the world frame
v^{local}_{base}	The linear velocity of the robot base in the robot's local heading frame: $R^T(\theta^{world}_{yaw}) \times v^{world}_{base}$
θ^{world}_{yaw}	The yaw orientation of the robot body in the world frame
ω^{world}_{yaw}	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 position
$\dot{q}$	The vector of all joint velocity
$\widehat{(\cdot)}$	The desired quantity of selected property $(\cdot)$, where $(\cdot)$ is a placeholder
$h^{world}_{foot,n}$	The n-th foot height (z) in the world frame
$v^{world}_{foot,n}$	The horizontal linear velocity $(\dot{x}, \dot{y})$ of the n-th foot in the world frame
p^{world}_{goal}	The horizontal component of the goal position (x, y) in the world frame
p^{world}_{robot}	The horizontal component of the robot base position in the world frame
$u^{base}_{goal,\,base}$	The unit vector pointing from the robot base to the goal at the base frame
$p^{world}_{foot,n}$	The horizontal placement of the n-th foot in the world frame

Soft Actor Critic

Compared to the classic reinforcement learning methods that obtain a policy $\pi(s_t)$ by maximising the expected sum of rewards as:

$$J(\pi) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}(r(s_t, a_t)), \tag{1}$$

we used the Soft Actor Critic (SAC) with the objective $J(\pi)$ optimised over an additional maximum entropy objective $H(\pi(\cdot|s_t))$ as:

$$J_{SAC}(\pi) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}(r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))), \tag{2}$$

where $H(\pi(\cdot|s_t))$ is the expected entropy of policy π over the sample distribution ρ_π. The temperature parameter α affects the stochasticity and the exploration capability of the optimal policy by changing the influence of the entropy term, where a higher α 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 $\mathcal{N}(\mu_\psi(s_t), \sigma_\psi(s_t)^2)$ with mean $\mu_\psi(s_t)$ and covariance $\sigma_\psi(s_t)$ generated by a neural network. A *tanh* function is applied to the Gaussian samples to project the action distribution within $\hat{a}(s_t) \in (-1, 1)$. The squashed action $\hat{a}(s_t)$ is then scaled by the joint range $\bar{q}$ to formulate the target action $a(s_t) = \bar{q}\hat{a}(s_t)$ as the joint position references for the robot. The hyperparameters of the SAC algorithm can be found in table S7.

Initialisation and early termination

We designed a set of initial reference states to initialise episodes for each task. As shown in fig. S26, 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 (fig. S27).

We specified three termination criteria for locomotion related tasks: (i) any body part

other than feet are in contact with the ground, (ii) the body orientation exceeding a threshold of 90°, and (iii) reaching the time limit of the episode. For the fall recovery task, only the 3rd criterion was used because the robot had to undergo the states specified in criteria (i) and (ii) to learn how to recover from failures autonomously.

Reward design
We used a radial basis function (RBF) to design a bounded reward function:

$$\varphi(x, \widehat{x}, \alpha) = exp(\alpha(\widehat{x} - x)^2) \ , \tag{3}$$

where x is the physical quantity for the evaluation, $\widehat{x}$ is the desired value, and α is the parameter that controls the width of the RBF. We utilised the RBF for all reward terms that involve continuous physical quantities. Reward terms with discrete properties, i.e., foot-ground contact, body-ground contact, and ground contact imitation, are designed separately. Details of the individual reward terms using the aforementioned nomenclature are presented in table S3 and table S4.

In table S3, 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 S3 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 $\widehat{h}$. 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.

Note S3. Expert imbalance phenomenon
In this section, we provide additional details of the implementation of MoE and its qualitative comparison with MELA. The MoE approach synthesises multiple experts using the weighted sum of the output of each expert network as:

$$o = \sum_{n=1}^{8} \alpha_n o_n, \tag{4}$$

where $n = 1, \ldots, 8$ is the index of experts, o_n are the output generated by the nth expert network, $\alpha_n \in [0, 1]$ are the variable weights generated by the gating network, and o is the final synthesised output of MoE.

The MoE policy was obtained using the same pre-trained experts and training

procedure as MELA. The comparison of the learning curves of MELA and MoE are presented in fig. S25B, where MELA shows a faster learning curve than MoE, e.g., a higher reward than MoE during 120-300 episodes. The training continued until 600 episodes to ensure that each policy had fully converged in order to be used for the comparison study.

The MoE and MELA policies were benchmarked in the same target-following test involving fall recovery, simultaneous trotting and steering. The target ball within the test scenario was designed to move forward in a straight line at a constant speed during 0-7s and then with a sinusoidal lateral motion superimposed on the forward motion during 7-40s.

We found that for fall recovery and forward trotting tasks, MoE and MELA policies show similar performance in these trained scenarios, because of the use of the same pre-trained experts and training settings. However, the performance becomes different when it comes to the adaptive behaviours that are represented by the newly emerged behaviours and transitional skills, which were not pre-trained: MELA steered left and right actively and followed the moving target very well, whereas MoE could not walk straight towards the target and its body was not oriented towards the direction of travel.

As shown in fig. S22A-C, the phenomenon of an asymmetric gait was observed from the MoE policy, which exhibits unbalanced left and right steering skills in contrast to the symmetric steering behaviour of the MELA policy, indicating that MoE's left and right steering experts are not trained well, with some experts favored more than others. During the target-following test, the robot controlled by MELA was able to turn left and right equally well for changing the heading direction progressively with the moving target, as shown in fig. S22A. However, the robot controlled by MoE could not follow the heading to the ball and was side stepping forward with unresponsive right steering, resulting in a constant offset of the heading and body yaw angle, as shown in the 3rd-5th snapshots in fig. S22B.

Figure S22C illustrates the global position of the target and the robot as well as the yaw angle using MELA and MoE, respectively. From 0-7s, the MELA-controlled robot trotted forward after getting up from a prone pose and tracked the zero yaw angle, while the MoE-controlled robot moved forward with an offset yaw angle because of degenerated steering experts. Moreover, during 7-40s, MELA showed equally active experts for left and right steering with a symmetric profile of yaw angle around zero, whereas MoE exhibits a drifting yaw angle and offset body pose due to unbalanced steering skills.

The t-SNE analysis in fig. S22D-E further reveals the underlying differences between MoE and MELA in fall recovery and trotting tasks respectively. In contrast to the distinct clusters from MELA (Fig. 4D-E), the lack of learning adaptive skills was observed in MoE that only some favored experts were well trained, leading to limited diversification of skills. In fig. S22D, only five experts are distinctly clustered (expert 1, 2, 5, 7, 8), while others are sparsely overlapped with each other. In fig. S22E, only four experts have

individual clusters (expert 2, 5, 7, 8), while others are scattered. This t-SNE analysis shows that approximately half of the experts are imbalanced in MoE leading to under-trained skills, and such limited diversification is disadvantageous in learning adaptive motor skills.

Figures

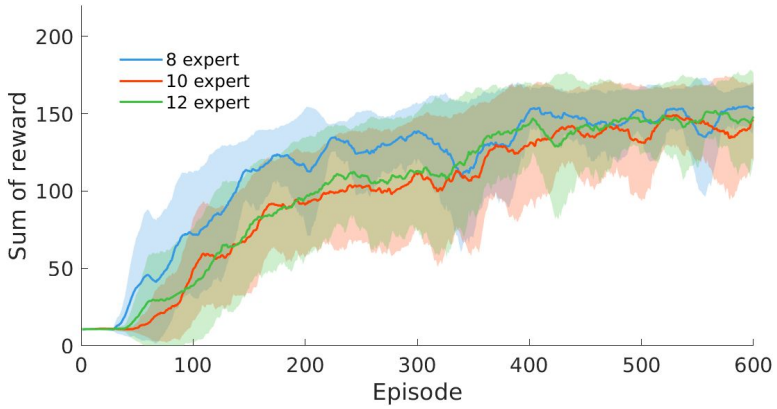


Fig. S1. Comparison of MELA's learning curves using different numbers of expert networks. It can be seen that using more than 8 experts does not improve the task performance, and has a slower convergence instead.

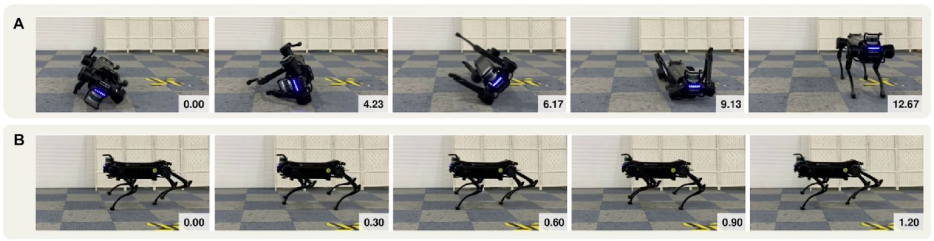


Fig. S2. Baseline experiments of fall recovery and trotting from engineered controllers. (**A**) The fall recovery from the engineered controller has a fixed sequence of motions and takes more than 12 s to stand up. (**B**) The trotting gait from the robot's control suite provided by the manufacturer. See movie S8 for more details.

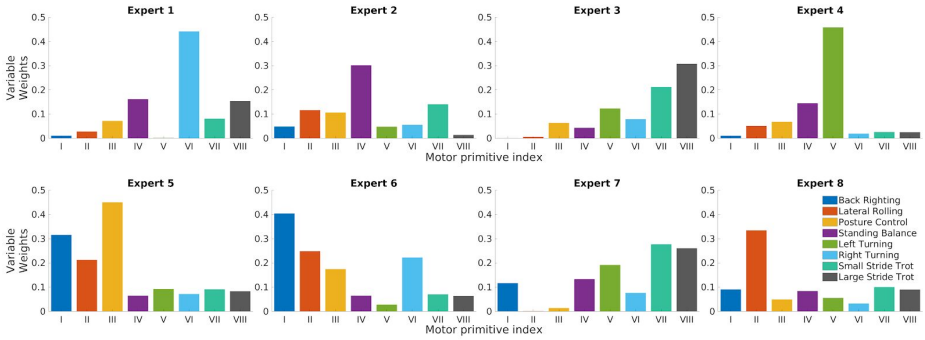


Fig. S3. Activation patterns of experts across all motor skills. The unique activation pattern of each expert, in which the specialisation is indicated by the highest activation of a motor skill numerated by roman numbers. The specialised motor skills of expert 1-8 are: (i) right turning, (ii) balance 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.

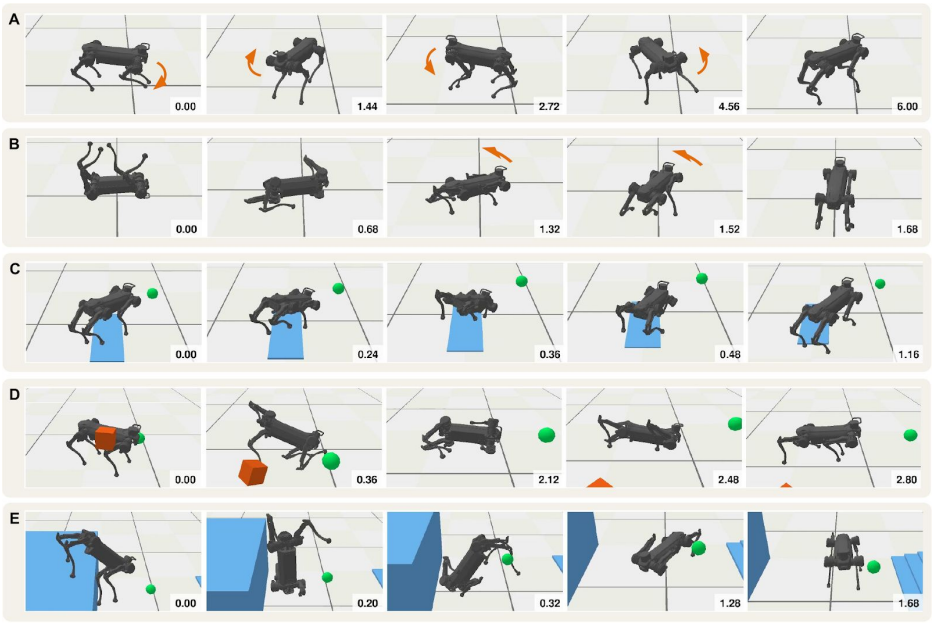


Fig. S4. Five representative cases showing adaptive behaviours of the MELA expert under unseen situations in simulation. (A) An emerged behaviour of left and right steering on the spot. (**B**) An emerged behaviour of simultaneous steering and standing up while recovering from fall to trotting. (**C**) 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. 5D). (**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).

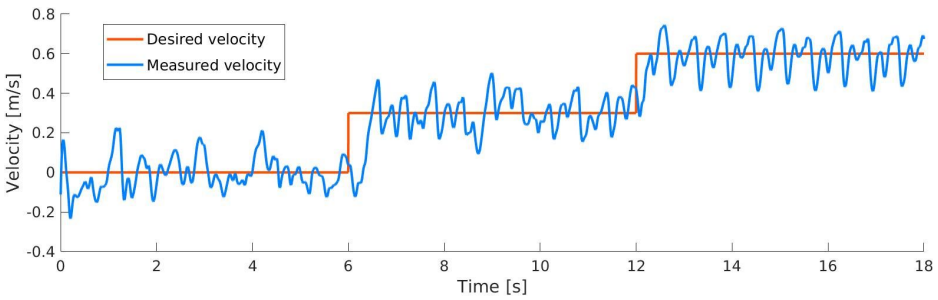


Fig. S5. Forward trotting velocity during the variable-speed trotting simulation. The robot adapted its trotting speed and followed the moving target (see movie S6).

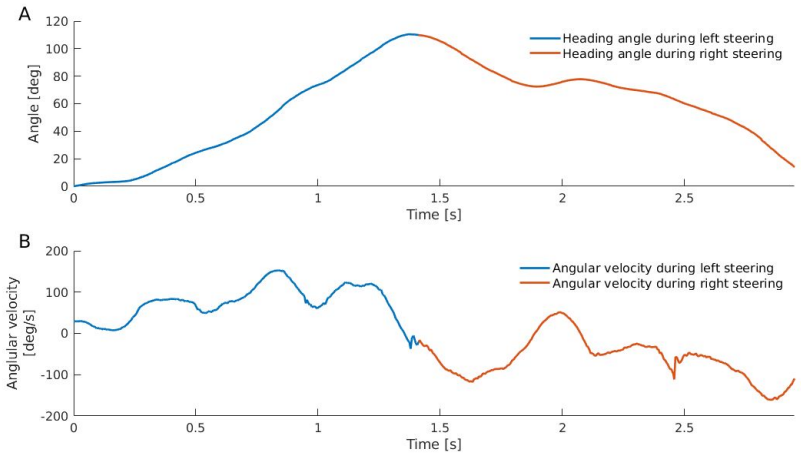


Fig. S6. Heading angle and angular velocity during the steering experiment on the real robot. The heading data here corresponds to the experiment in Fig. 5B. (**A**) The robot first steered counter-clockwise towards the left and then clockwise towards the right. (**B**) The average yawing velocities 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*).

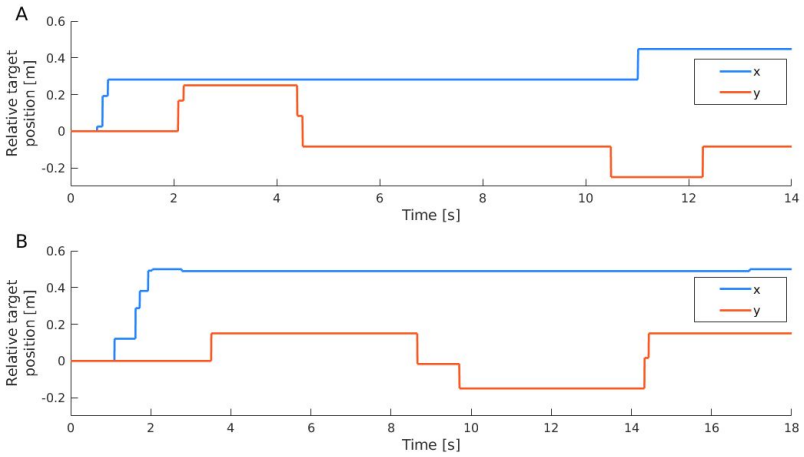


Fig. S7. Relative target positions with respect to the robot from the user command as the input to the MELA networks during the real locomotion experiment. (**A**) The changing target position (x, y) during the real target-following experiment (Fig. 5C), and runtime was 14 seconds. (**B**) The changing target position (x, y) during the fall-resilient experiment (Fig. 5D), and runtime was 18 seconds.

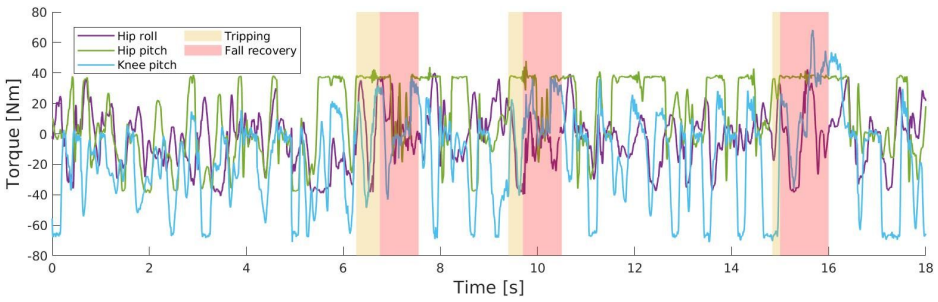


Fig. S8. Measured torques of the front left leg during the real locomotion experiment (Fig. 5D). During this experiment, the robot trotted at large steps (see the larger commanded (x, y) target positions in fig. S7B) and saturated the motor torques at times, e.g., the hip pitch joint. In this case, the torque-saturated leg was not able to move as intended and the robot stumbled and tripped (yellow regions), leading to fall recoveries (red regions).

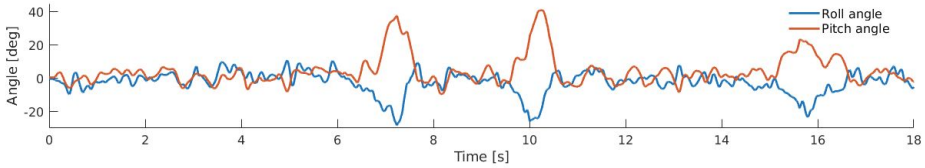


Fig. S9. Roll and pitch angles during the real locomotion experiment. The measurements correspond to the experiment presented in Fig. 5D. Significant changes were observed in the body orientation by the roll and pitch angles during the tripping moments. The peaks in roll and pitch angles were up to 0.47 *rad* (26.7 *deg*) and 0.7 *rad* (39.8 *deg*), respectively. The short duration of the large deviations of the body posture shows that the recovery was accomplished within 1 second time.

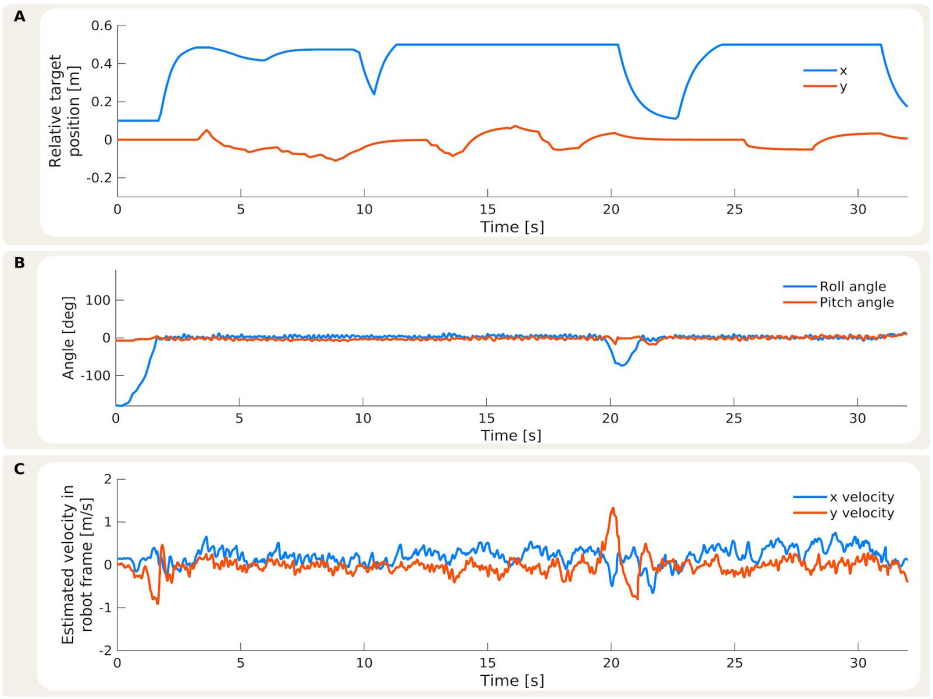


Fig. S10. Target position, body orientation and velocity during the real locomotion experiment on grass (Fig. 5E). (A) Relative target positions with respect to the robot from the user command as the input to the MELA networks. **(B)** The measured roll and pitch angles of the robot. The robot is able to maintain a steady posture while trotting on uneven grass. **(C)** The estimated velocity of the robot represented in the local robot frame. The robot trots with a forward velocity around 0.25 *m/s*. The robot experiences large lateral velocity at around the 20s moment when it is being knocked over.

Fig. S11. Adaptive and agile locomotion behaviours on pebbles and grass demonstrated by MELA policy. (**A-E**) Self-righting and trotting on pebbles. (**F-J**) Self-righting, trotting, and adaptive fall recovery in presence of various pushes on grass.

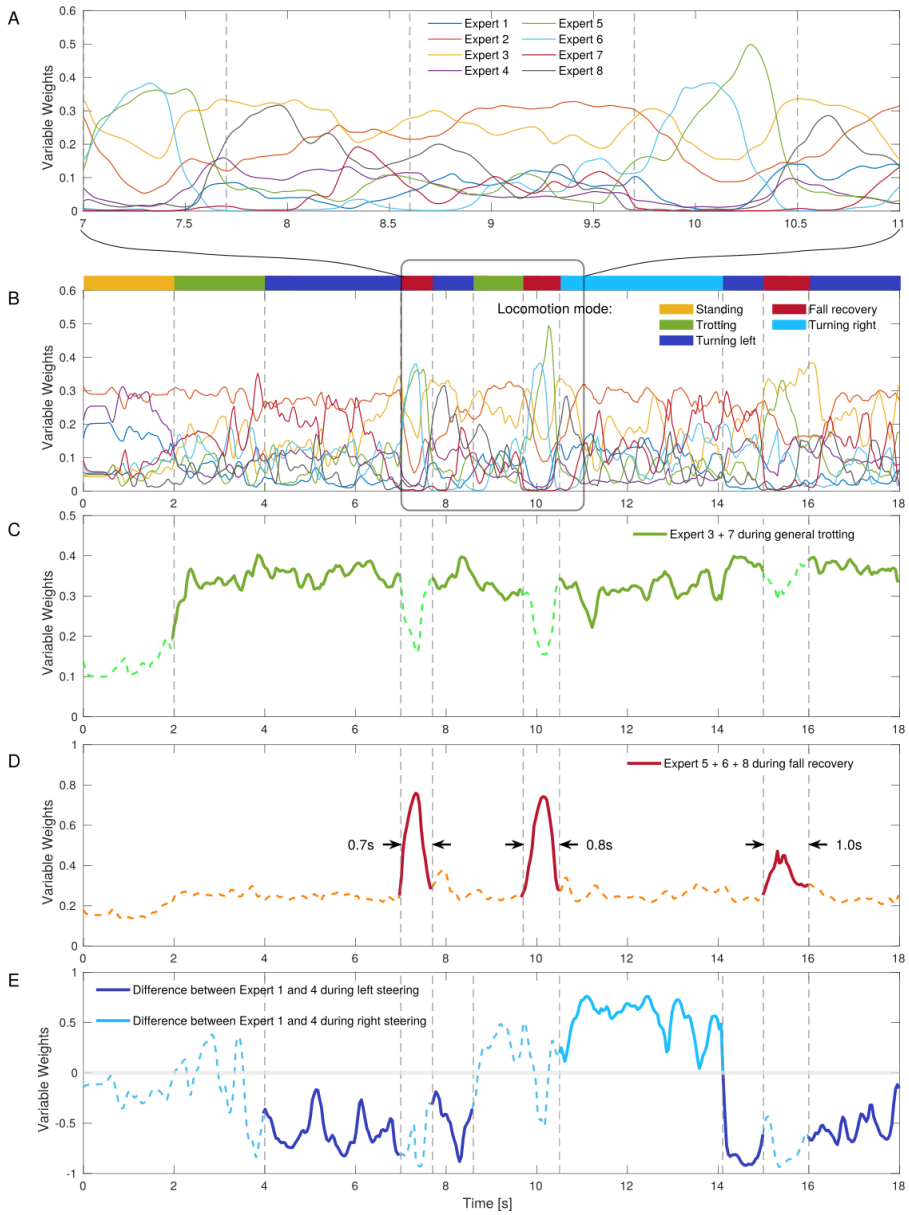


Fig. S12. Continuous and variable weights of all experts during the real MELA

experiment (Fig. 5D). (**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.

Fig. S13. Continuous and variable weights of all experts during the real MELA experiment on grass (Fig. 5E). (**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 and fall recovery during a target-following task. (**C-D**) The activation levels of paired weights from collaborating experts, where the expert groups (3, 7) and (5, 6, 8) cooperated together in trotting and fall recovery, respectively.

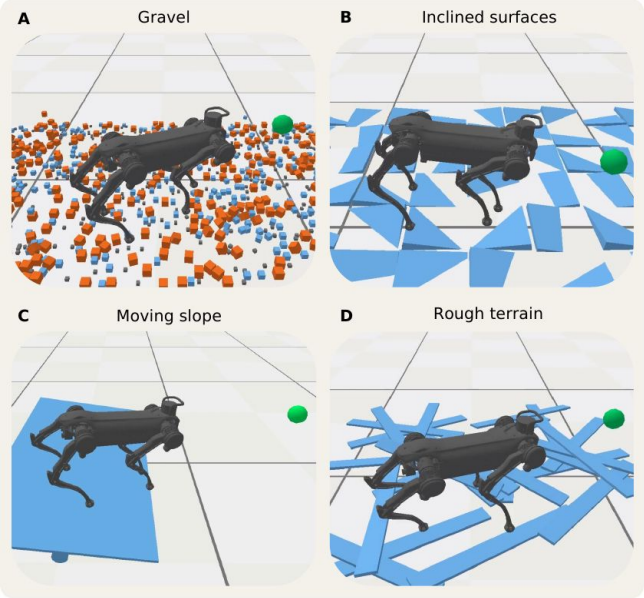


Fig. S14. Four types of unseen terrains for testing the multi-skill MELA policy in the simulation. (**A**) The gravel is constructed by a variety of freely moving **cubes** with dimensions of 0.02m, 0.035m, and 0.05m. (**B**) The inclined surfaces consist of rectangular slabs (0.4 m x 0.4 m x 0.2 m), 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.

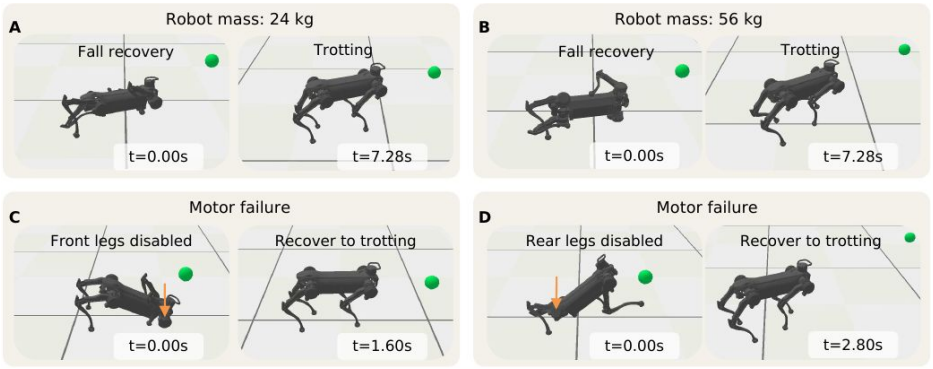


Fig. S15. Simulated test scenarios for evaluating the robustness of the MELA policy. (**A-B**) Uncertainties in dynamic properties are simulated by modifying the robot model, i.e., robot mass with variations of $\pm 25\%$, $\pm 30\%$ and $\pm 40\%$ of the original value (40kg). We show snapshots with the mass variation of $\pm 40\%$ as an extreme example, and the rest of the tests can be seen in movie S7. (**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.

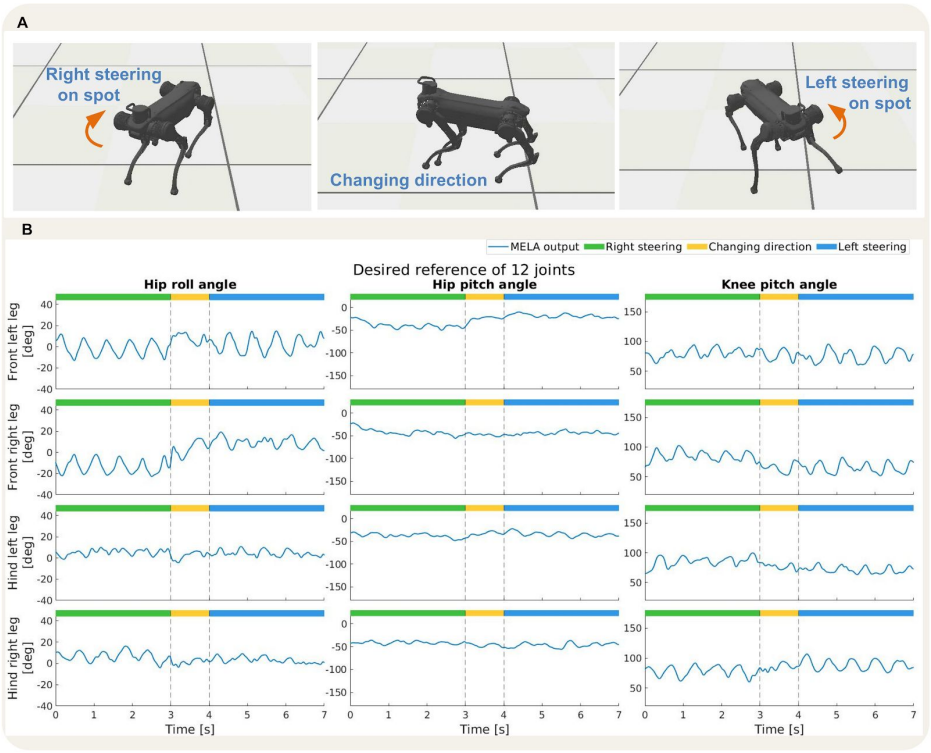


Fig. S16. Representative adaptive behaviour from the simulated scenario of steering on spot (fig. S4A, movie S6). (**A**) Snapshots depicting the behaviours during left and right steering. (**B**) Position references of all the joints during left and right steering phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

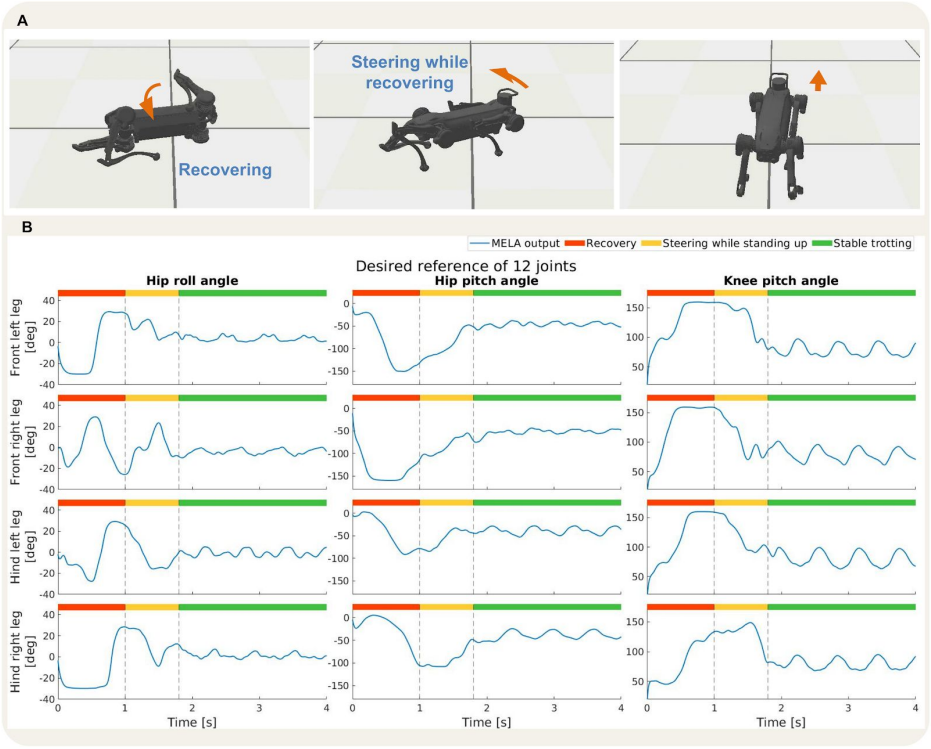


Fig. S17. Representative adaptive behaviour from the simulated scenario of steering while recovering to trotting (fig. S4B, movie S6). (**A**) Snapshots depicting the behaviours during recovery and steering. (**B**) Position references of all the joints during recovery, steering, recovery, and trotting phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

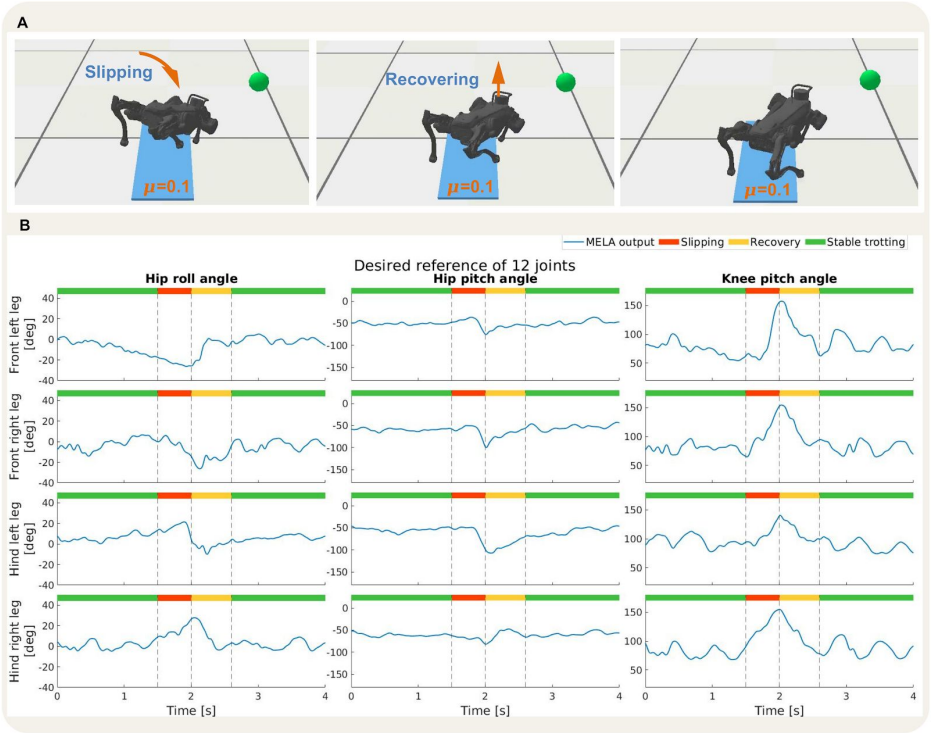


Fig. S18. Representative adaptive behaviour from the simulated scenario of tripping (fig. S4C, movie S6). (**A**) Snapshots depicting the behaviours during slipping and recovery. (**B**) Position references of all joints during slipping, recovery, and trotting phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

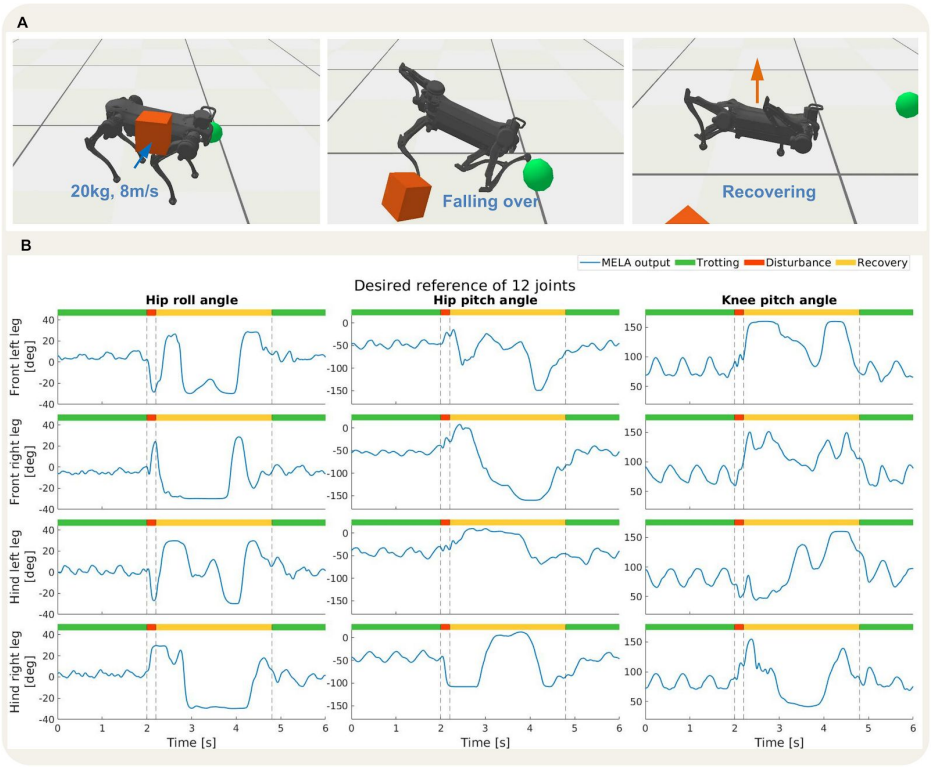


Fig. S19. Representative adaptive behaviour from the simulated scenario of a large impact (fig. S4D, movie S6). (**A**) Snapshots depicting the behaviours during the moment of disturbance and recovery (**B**) Position references of all the joints during trotting, disturbance, and recovery phases. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

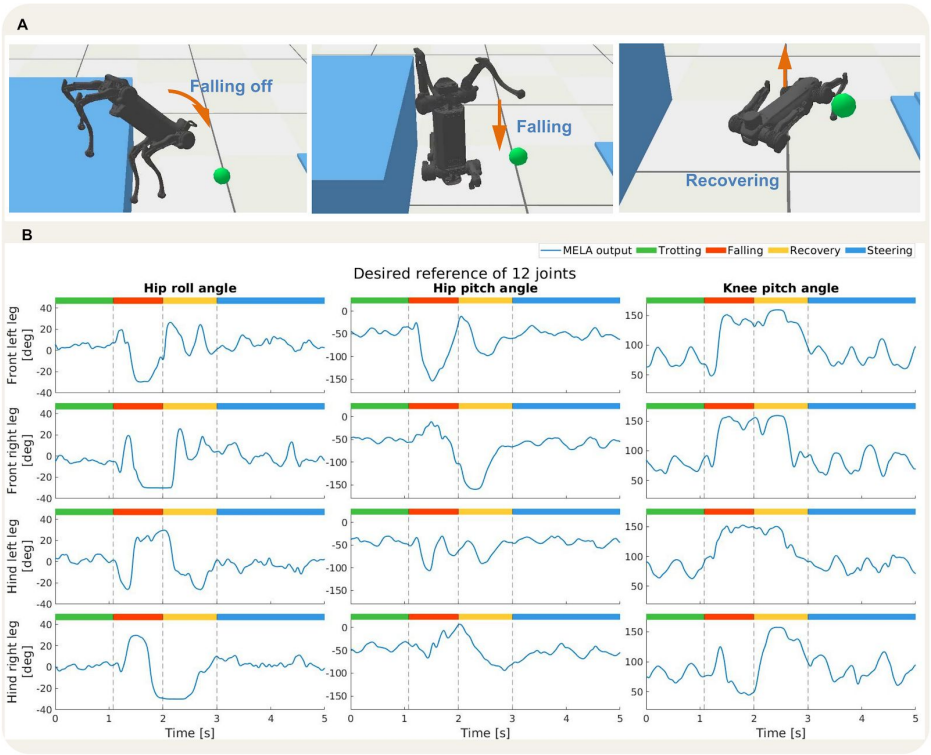


Fig. S20. Representative adaptive behaviour from the simulated scenario of falling off a cliff (fig. S4E, movie S6). (**A**) Snapshots depicting the behaviours during falling and recovery. (**B**) Position references of all the joints during falling, recovery, and steering. The smooth change in desired joint positions indicates that the MELA framework has learned how to synthesise expert skills during various transitions seamlessly.

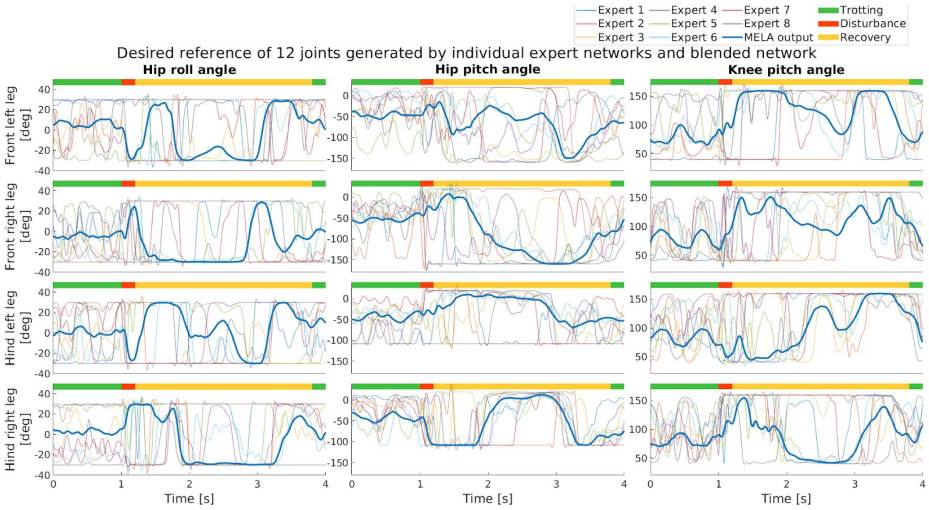


Fig. S21. Analysis of responses from the MELA policy during the simulated scenario of a large external perturbation (fig. S4D, movie S6). We use the case of a flying box impact as a representative example to show how the policy actively reacts to perturbations with a smooth and seamless transition. The 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.

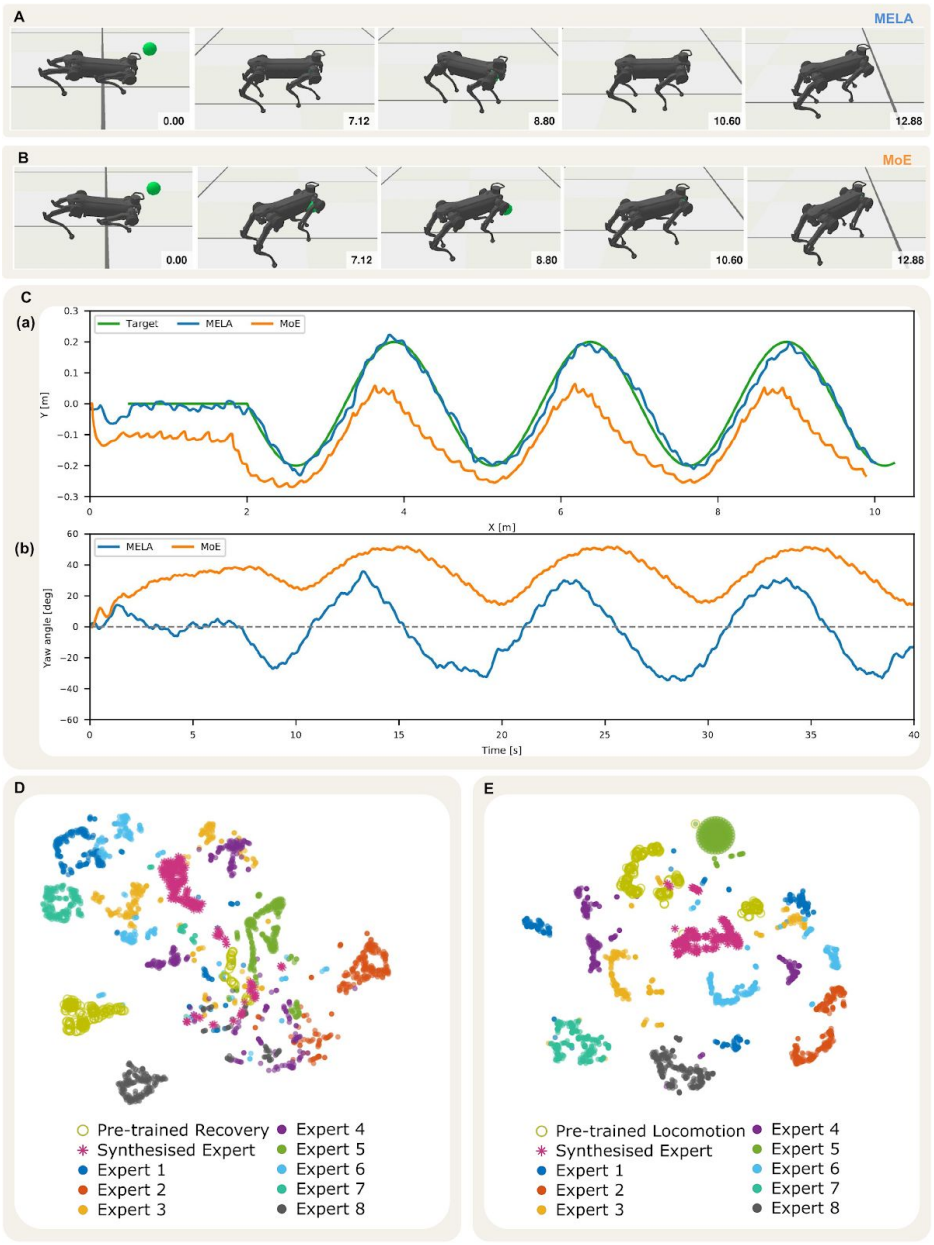


Fig. S22. Phenomenon of asymmetric gait and imbalanced experts from MoE. (**A**)

MELA policy performing a target-following test with fall recovery, steering and trotting. The robot was able to steer left and right to follow the moving target. (**B**) MoE policy performing a target-following test, where the yaw angle drifted due to the degenerated expert in steering skills. (**C**) Global position and yaw angle of the robot during the target-following test using MELA and MoE, respectively. (**D**) and (**E**) are the t-SNE analysis of MoE during fall recovery and trotting respectively, using target actions from pre-trained, co-trained, and synthesised experts. In contrast to the eight unique clusters from MELA (see Fig. 4D-E), the expert imbalance was observed in MoE that only half favoured experts were well trained with discernible clusters, while others are scattered and overlapped.

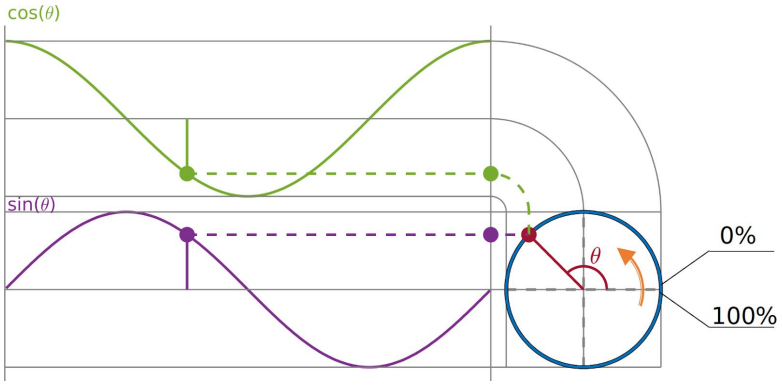


Fig. S23. Illustration of the 2D phase vector for training the locomotion policy. The sine and cosine functions are used to represent the time-varying phase variable in a continuous manner, and the resulting phase vector contains temporal information to describe the phase (0-100%) of a periodic gait.

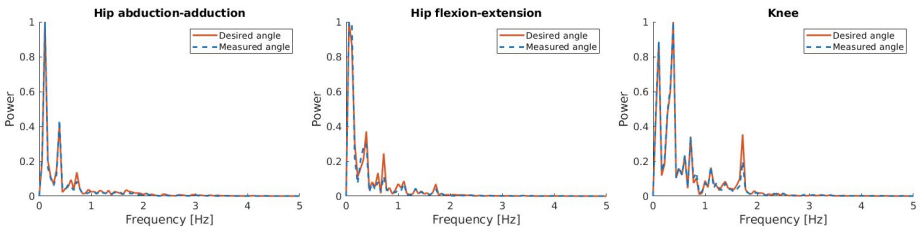


Fig. S24. Normalised power spectrum analysis of motions during the real locomotion experiment (without the DC component). The data were collected from the experiment shown in Fig. 5D. The majority of the frequency components were below 1Hz, and some small components were around 1.67Hz corresponding to the trotting motions, which indicated that all useful motion components were unaffected by the action filters.

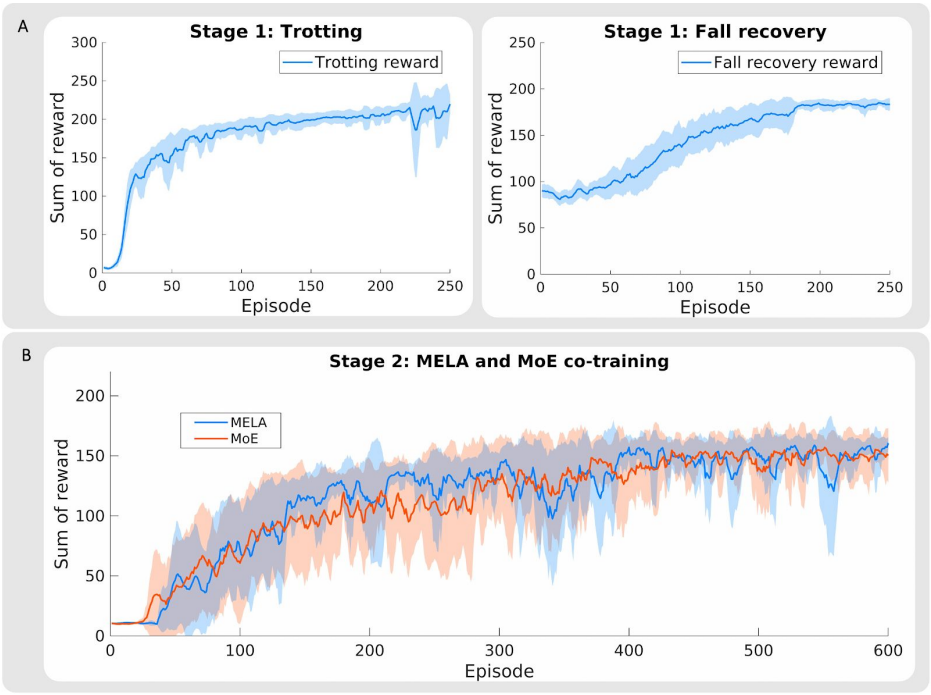


Fig. S25. Learning curves during the 2-stage training of MELA and MoE. (**A**) Learning curves of fall recovery and trotting experts. For training experts in the first stage, 250 episodes were required for both the fall recovery and trotting tasks. The same fall recovery and trotting experts can be used for the second stage of contraining for both MELA and MoE. (**B**) Learning curves of MELA and MoE during the second stage of co-training. The training continued until 600 episodes to ensure that each policy had fully converged to its maximum reward for a comparison study. Each episode consists of 5000 samples which were collected at 25 Hz.

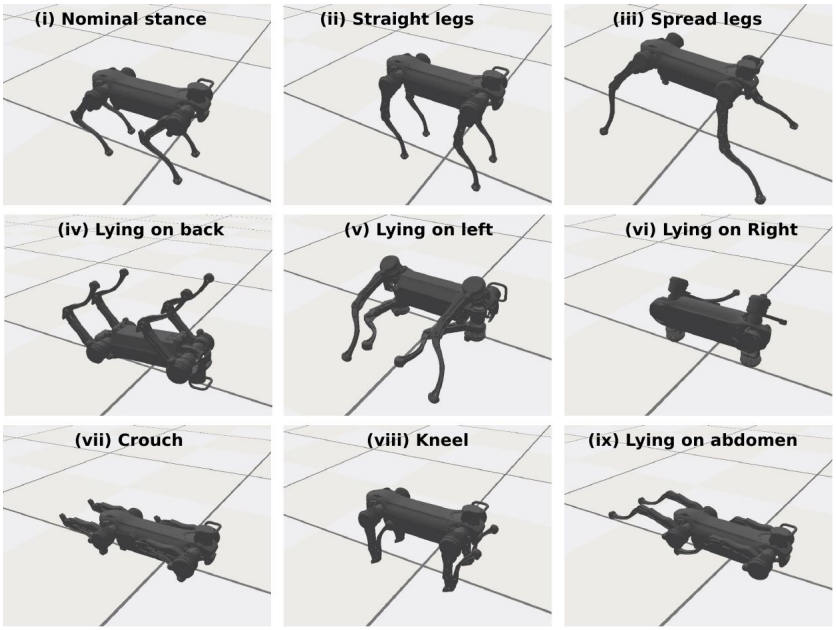


Fig. S26. Nine distinct configurations used as the initialisation for training fall recovery policies in simulation. Snapshots are taken from the physics-based simulator using the PyBullet engine (*64*).

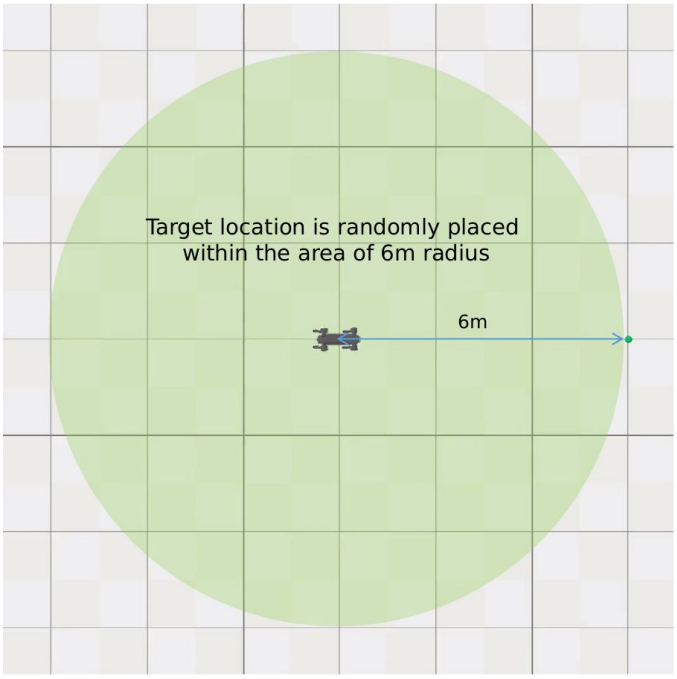


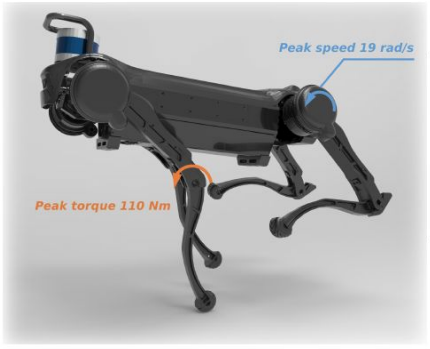
Fig. S27. Setting of the target location for training MELA policies in simulation. During the initialisation of each sample collection episode, the target location (the green ball) is randomly placed within the area of 6 m radius around the robot, and remains fixed within the same episode.

Tables

Table S1. Distribution matrix of expert specialisations over motor skills. As shown in Fig. 4A, more than one expert is activated for a motor skill, and hence we highlight the top two dominant experts that are activated in each motor skill in this table.

Motor Skill index	Expert index							
	1	2	3	4	5	6	7	8
I					●	●		
II						●		●
III					●	●		
IV	●	●						
V				●			●	
VI	●					●		
VII			●				●	
VIII			●				●	

Table S2. Specification of the *Jueying* quadruped robot.



	Joint range (°)	$\tau\ (Nm)$ [*Peak*]	$\omega\ (rad/s)$ [*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	

Table S3. Detailed descriptions of the individual reward terms.

Physical quantities	Equations
Base pose	$\varphi(\phi, [0, 0, -1], \alpha)$, $\alpha = -2.35$
Base height	$\varphi(h, \widehat{h}, \alpha)$, $\alpha = -51.16$
Base velocity	$\varphi(v_{base}^{world}, \widehat{v}_{base}^{world}, \alpha)$, $\alpha = -18.42$
Joint torque regularisation	$\varphi(\tau, 0, \alpha)$, $\alpha = -0.003$
Joint velocity regularisation	$\varphi(\dot{q}, 0, \alpha)$, $\alpha = -0.026$
Foot ground contact	$\begin{cases} 1, & \text{foot in contact with ground} \\ 0 \end{cases}$
Body ground contact	$\begin{cases} 0, & \text{main body in contact with ground} \\ 1 \end{cases}$
Yaw velocity	$\varphi(\omega, 0, \alpha)$, $\alpha = -7.47$
Joint position reference	$\varphi(q, \hat{q}, \alpha)$, $\hat{q}$ is the joint reference, $\alpha = -29.88$
Foot contact reference	$\begin{cases} 1, & \text{match desired foot contact in imitation data} \\ 0 \end{cases}$
Robot heading	$\varphi(u_{goal,base}^{base}, [1, 0, 0], \alpha)$, $\alpha = -2.35$
Goal position	$\varphi(p_{goal}^{world}, p_{base}^{world}, \alpha)$, $\alpha = -0.74$
Swing and stance	$\varphi(1/4 \cdot \sum_{n=1}^{4}((h_{foot,n}^{world} - \widehat{h}_{foot,n}^{world}) \cdot v_{foot,n}^{world}), 0, \alpha)$, $\alpha = -460.50$
Foot placement	$\varphi(1/4 \cdot \sum_{n=1}^{4} p_{foot,n}^{world}, p_{base}^{world}, \alpha)$, $\alpha = -18.42$

Table S4. 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.

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 regularisation	**0.018**	**0.067**	**0.020**
Joint velocity regularisation	**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**	0.00	**0.020**
Foot clearance	**0.036**	0.00	**0.036**
Joint position reference	**0.416**	0.00	**0.167**
Foot contact reference	**0.083**	0.00	**0.033**
Average foot placement	**0.036**	0.00	**0.036**
Robot heading	0.00	0.00	**0.143**
Goal position	0.00	0.00	**0.214**

Table S5. Selection of state inputs for different tasks and neural networks. The MELA gating network receives task-oriented state inputs (i.e., the normalised gravity vector, the angular and linear velocities, and the goal position) for the target-following locomotion; and the synthesised MELA network receives all the feedback except the goal position.

Physical quantities	Trotting	Fall Recovery	MELA Gating Network	Synthesised MELA Network
Joint position	✓	✓		✓
Gravity vector	✓	✓	✓	✓
Angular velocity of the robot	✓	✓	✓	✓
Linear velocity of the robot in its local heading frame	✓		✓	✓
Phase vector	✓			✓
Goal position			✓	

Table S6. Proportional-Derivative parameters for the joint-level PD controller.

	Hip roll	Hip Pitch	Knee pitch
Kp (Nm/rad)	700	700	700
Kd (Nms/rad)	10	10	10

Table S7. Hyperparameters for SAC.

Smoothing loss coefficient	2.0
Learning rate	3e-4
Weight decay	1e-6
Discount factor	0.987
Soft target update	0.001
Replay buffer size	1e6
Steps per epoch	5e3

Multi-Expert Synthesis | K

ights for quadruped experts.

	w_{task}:	x_{vel}	y_{vel}	z_{vel}	z_{pos}	g^L	reg	w_{imit}:	q	$\dot{q}$	contact	end-effector pos
Gait Recovery r_{gr}	1.0	1	1	1	5	10	1	0.0				
Standing r_{stand}	1.0	2	2	2	4	4	1	0.0				
Locomotion r_{loco}	0.3	6	1	0	1	3	1	0.7	0.5	0.2	0.05	0.25