Control of Robots with Hybrid Locomotion Capabilities

von

Babu Ajish

Dissertation zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

Vorgelegt im Fachbereich 3 (Mathematik & Informatik)
der Universität Bremen

1. September 2025

Universität Bremen

dfki ai

Datum des Promotionskolloquiums: 26.08.2025

Gutachter: Prof. Dr. Dr. h.c. Frank Kirchner
Prof. Dr.-Ing. Udo Frese

Abstract

Hybrid locomotion robots—systems combining multiple modes of motion—are well-suited for challenging terrain and have broad practical applications. However, developing effective control strategies remains challenging due to nonlinear dynamics, multimodal locomotion, computational constraints, and multiple objectives. Existing solutions are often not portable to unconventional morphologies, requiring substantial redesign. This thesis proposes multiple control solutions for three morphologically distinct robots: Asguard, SherpaTT, and ARTER.

Asguard's five-spike wheel design presents challenges in forward motion and point turning. A cascaded position–velocity–torque controller improves wheel positioning accuracy by up to 56 % while maintaining defined offsets. The controller based on a novel torque estimator using mechanical coupling deflection enables real-time torque control. Adjusting inter-wheel offsets reduces resistance to motion by up to 90 %. For point turns on rough terrain, an algorithm that utilizes external load torques enables reliable rotation, outperforming baseline controllers.

Dedicated control frameworks, denoted as Motion Control System (MCS), are developed for the wheel-legged robots SherpaTT and ARTER. SherpaTT-MCS supports teleoperation, assistance functions, and autonomy. Its terrain adaptation module improves force distribution by 80 % and reduces attitude error by up to 95 % in laboratory tests. It has been successfully deployed in over ten research projects and has proven its efficacy in three Mars-analogous field trials. ARTER-MCS incorporates kinematic modeling of parallel linkages and nonlinear model predictive control, and is currently in active use in multiple projects.

For ARTER, a Deep Reinforcement Learning (DRL)-based terrain adaptation controller is introduced, leveraging compressed height-maps via autoencoders. Ten variants of the controller were trained using different combinations of observations, including contact distances and latent-space representations. The controller that demonstrated the strongest performance utilized contact-detection and a 4-dimensional terrain latent space as observation, offering a favorable combination of both performance and complexity. All controllers achieved baseline objectives and has the potential to be ported to other platforms with active suspension.

ARTER also demonstrates stepping locomotion via a controller that combines the movement of the manipulator arm, the legs and the wheels. This controller applies hierarchical reinforcement learning and action masking, integrating domain knowledge to simplify training. A three-level hierarchy is employed: the lowest level manages diverse simpler motions (manipulator, longitudinal motion, end-effector height adjustments, etc.); the middle level sequences these for stepping in and out of obstacles; the top level manages task transitions. The architecture generalizes across three different types of stepping terrain and generated motion comparable to that of an expert operator.

In summary, this thesis presents a series of control solutions designed to enhance the locomotion performance, efficiency, and adaptability of hybrid robotic platforms. The learning-based methods offer strong morphological generalization and address long-sequence tasks with reduced engineering effort. These contributions represent quantitative and qualitative advancements in the control of diverse robots with hybrid locomotion capabilities.

Zusammenfassung

Roboter mit hybriden Lokomotionsfähigkeiten kombinieren mehrere Bewegungsmodi, wodurch sie sich gut für schwieriges Gelände eignen und umfassende praktische Anwendungsmöglichkeiten bieten. Die Entwicklung effektiver Steuerungsstrategien bleibt jedoch aufgrund nichtlinearer Dynamik, multimodaler Lokomotion, rechnerischer Einschränkungen und vielfältiger Ziele eine Herausforderung. Bestehende Lösungen lassen sich oft nicht auf unkonventionelle Morphologien übertragen und erfordern eine umfassende Neukonzeption. Diese Arbeit schlägt mehrere Steuerungslösungen für drei morphologisch unterschiedliche Roboter vor: Asguard, SherpaTT und ARTER.

Das Fünf-Speichen-Rad-Design von Asguard stellt Herausforderungen bei der Vorwärtsbewegung und beim Punktdrehen dar. Ein kaskadierter verbessert die Radpositionierungsgenauigkeit um bis zu 56 %, während definierte Offsets beibehalten werden. Der Regler basiert auf einem neuartigen Drehmoment-Schätzer, der die mechanische Kupplungsauslenkung nutzt und eine ermöglicht. Durch die Anpassung der Radversätze wird der Bewegungswiderstand um bis zu 90 % reduziert. Für Punktdrehungen auf unebenem Gelände ermöglicht ein Algorithmus, der externe Lastmomente nutzt, eine zuverlässige Drehung und übertrifft damit die Basisregler.

Für SherpaTT und ARTER wurden spezielle Steuerungsframeworks entwickelt, die als Motion Control System (MCS) bezeichnet werden. SherpaTT-MCS unterstützt Fernsteuerung, Assistenzfunktionen und Autonomie. Sein Geländeanpassungsmodul verbessert die Kraftverteilung um 80 % und reduziert den Lagefehler in Labortests um bis zu 95 %. Der Roboter wurde bereits in über zehn Forschungsprojekten erfolgreich eingesetzt und hat seine Wirksamkeit in drei Mars-analogen Feldversuchen unter Beweis gestellt. ARTER-MCS umfasst die kinematische Modellierung von Parallelverbindungen sowie eine nichtlineare modellprädiktive Steuerung. Das System wird derzeit in mehreren Projekten aktiv eingesetzt.

Für ARTER wird ein auf Deep Reinforcement Learning (DRL) basierender eingeführt, der komprimierte Höhenkarten über Autoencoder nutzt. Zehn Varianten des Controllers wurden unter Verwendung verschiedener Kombinationen von Beobachtungen trainiert, darunter Kontaktabstände und Latent-Space-Darstellungen. Der Controller, der die stärkste Leistung zeigte, nutzte die Kontakterkennung und einen 4-dimensionalen Terrain-Latent-Space als Beobachtung und bot eine günstige Kombination aus Leistung und Komplexität. Alle Controller erreichten die grundlegenden Ziele und können potenziell auf andere Plattformen mit aktiver Federung übertragen werden.

ARTER demonstriert auch die Schrittbewegung, die die Bewegung seines Manipulatorarms, seiner Beine und seiner Räder kombiniert. Dieser Controller wendet hierarchisches Reinforcement Learning und Action Masking an und integriert Domänenwissen, um das Training zu vereinfachen. Es wird eine dreistufige Hierarchie verwendet: Die unterste Ebene verwaltet verschiedene einfachere Bewegungen (Manipulator, longitudinale Bewegung, Höhenverstellungen des Endeffektors); die mittlere Ebene sequenziert diese für das Ein- und Aussteigen aus Hindernissen; die oberste Ebene verwaltet Aufgabenübergänge. Die Architektur lässt sich auf drei verschiedene Schrittarten verallgemeinern.

Zusammenfassend präsentiert diese Arbeit eine Reihe von Regelungsansätzen zur Verbesserung der Lokomotion hybrider robotischer Systeme hinsichtlich ihrer Effizienz und Anpassungsfähigkeit. Die lernbasierten Methoden bieten eine starke morphologische Generalisierung und bewältigen Aufgaben mit langen Sequenzen mit reduziertem technischen Aufwand. Diese Beiträge stellen quantitative und qualitative Fortschritte bei der Steuerung verschiedener Roboter mit hybriden Lokomotionsfähigkeiten dar.

Acknowledgments

I want to take this opportunity to extend my heartfelt thanks to the many individuals whose unwavering support, guidance and encouragement made this doctoral thesis possible.

First and foremost, I would like to express my most profound appreciation to my doctoral supervisor, Professor Frank Kirchner. His wisdom, insightful guidance and boundless inspiration were instrumental in shaping the core ideas and direction of this research. Through countless discussions and academic retreats, Professor Kirchner helped refine the thesis objectives, clarify key concepts, and contributed significantly to developing innovative solutions. His mentorship provided me not only with academic guidance but also with the opportunity to participate in several exciting research projects at DFKI (German Research Center for Artificial Intelligence), particularly in the area of locomotion in robotics. For all of this, I am deeply grateful.

I would also like to extend my sincere thanks to Dr. Daniel Kühn, Dr. Malte Wirkus, Dr. Florian Cordes, Dr. Dennis Mronga and Dr. Sebastian Bartsch whose administrative and logistical support was crucial throughout the entire process. Their assistance helped me navigate through the complex challenges that arise in completing such a demanding academic endeavor.

I am especially grateful to my dear friends and colleagues, Sankar Natarajan and Vinzenz Bargsten, for their thought-provoking discussions and shared ideas. Their insights contributed to the development of several solutions presented in this thesis. Their friendship and intellectual stimulation were invaluable during this journey. Special thanks go to Yi-Ling Liu and Siddhant Shete for reviewing and proofreading the thesis.

Many other colleagues who collaborated with me on various projects have also supported my work, both directly and indirectly. I had the privilege of working with many current and former colleagues at the Robotics Innovation Center (RIC), and I am profoundly grateful for their encouragement, support, and guidance over the years. Although it is impossible to mention every name here, I extend my thanks to all of them and sincerely apologize if I have overlooked anyone.

On a personal level, words are not enough to express my gratitude to my parents and my brother, Bijish, who have been my pillars of strength, always providing encouragement and inspiration when it was needed most. Their belief in me has fueled my determination to achieve this goal.

Most importantly, I owe a deep debt of gratitude to my wife, Thabitha, whose love, patience, and understanding made this journey possible. She has been my constant source of motivation, providing unwavering support while making immense personal sacrifices. To our wonderful children, Theo and Antonio, who bring endless joy into our lives, I hope you will grow up to share in the wonders and excitement that science brings. Without the unwavering love and encouragement of my family, this accomplishment would not have been possible; I dedicate this dissertation to you.

Babu Ajish 01.09.2025

CONTENTS

List of Figures

List of Tables

Introduction

Autonomous robots are being developed and deployed in challenging environments for applications, including search and rescue, surveillance, environmental monitoring, inspection, maintenance and logistical operations. To fulfill their intended objectives, the robots must perform a variety of complex tasks, such as navigation, manipulation, mobility, state estimation, sensing, perception and decision-making. A particularly crucial capability for mobile robots is the ability to traverse an environment from one location to another. This capability is referred to as *locomotion*, a term derived from the Latin roots *locus* and *motion*, signifying "place" and "movement" respectively. The specific mode of locomotion adopted by a robot is predominantly influenced by the characteristics of its operating environment, and its own mobility capabilities.

This chapter provides a concise synopsis of the locomotion mechanisms observed in the biological realm and the various forms of locomotion exhibited by robots. Subsequently, the concept of hybrid locomotion, which constitutes the primary focus of this thesis, is examined. An extensive review of the pertinent literature on the control of locomotion with hybrid locomotion capabilities is also conducted. The objectives of the thesis are subsequently formulated, and the chapter culminates with a brief account of the thesis's structure. The following section introduces the locomotion mechanisms in animals.

1.1 Locomotion Mechanisms in Animals

In the biological world, animals employ a variety of locomotion types to traverse diverse environments and terrains. The primary motivations for this adaptability include the acquisition of food, the assurance of safety, the facilitation of mating and the establishment of suitable habitats. An analysis of animal locomotion reveals two broad categories: self-propelled and passive. Running, jumping, flying, hopping and swimming are examples of self-propelled locomotion which involves the expenditure of energy by the animal to move. In contrast, passive locomotion, such as sailing, kiting and rolling, involves the exploitation of external forces to achieve mobility.

Evolution plays a predominant role in the development of animal locomotion, with the fitness of the animal serving as the determining factor. According to Alexander 2002, the fitness for natural selection could be decided by the following factors: (i) speed, (ii) acceleration, (iii) maneuverability, (iv) endurance, (v) economy of energy and (vi) stability.

The development of locomotion is influenced by a compromise between the aforementioned factors and various other properties shaped by the animal's environment. Additionally, numerous environmental constraints exert their influence on the developmental process. In essence, this problem can be conceptualized as a design optimization problem, wherein the objective is to enhance a locomotion

function while considering the constraints imposed. The outcomes manifest in a variety of forms, driven by the constraints encountered, resulting in diverse forms of locomotion observed across different animal species.

Table 1.1: An overview of the locomotion mechanisms utilized in biological systems is outlined, inclusive of the nature of motion, resistance to motion and kinematic processes. This overview is inspired by Figure 2.1 in Siegwart and Nourbakhsh 2004.

Type of Motion	**Resistance to Motion**	**Basic Kinematics of Motion**
Flow in a channel	Hydrodynamic forces	Eddies
Crawl	Friction forces	Longitudinal vibration
Sliding	Friction forces	Transverse vibration
Running	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum
Jumping	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum
Walking	Gravitational forces	Rolling of polygon

Animals have evolved to utilize the environmental characteristics to facilitate efficient locomotion. For instance, human walking (Inman 1966) employs the conversion between kinematic and potential energy, as well as passive motion of the limbs, to ensure optimal efficiency. Beal et al. 2006 elaborates on how a dead fish can propel upstream by leveraging the flexibility of its body and the oncoming vortices. Similarly, migratory birds employ wind to traverse vast distances, often thousands of kilometers, without pause or sustenance. This efficiency can be attributed to the strategic use of horizontal tailwinds (Liechti 2006) and vertical winds (Duriez et al. 2018). Siegwart and Nourbakhsh 2004 provides a brief overview (Table 1.1) of the mechanisms employed by animals for locomotion, along with the primary factors that resist movement and the fundamental kinematic principles underlying locomotion. A significant proportion of robot locomotion research draws inspiration from animal locomotion studies.

1.2 Bio-Inspired Locomotion in Robots

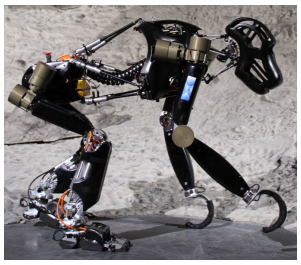
Robots have been developed with a wide variety of locomotion capabilities, the majority of which are inspired from the biological world. The locomotion mechanism of the robot depends on the application and the environment it is interacting with, as is the case with biological systems. A variety of locomotion types have been developed and used, some of them only as research platforms. These include rolling, walking, hopping, brachiating, slithering, flying, swimming and metachronal locomotion, among others.

Rolling with wheels is the most prevalent locomotion type due to its ease of control, efficiency on even terrains and high stability. Wheeled motion is rarely observed in biological systems due to the difficulty of developing and actuating a wheel-like mechanism in biological systems and also since most animals need to traverse uneven terrain. As per the findings of Siegwart and Nourbakhsh 2004, human walking bears similarity to a rolling polygon.

Legged walking robots have also garnered significant interest in research communities, particularly those that possess bipedal or quadrupedal locomotion. Robots that utilize legged walking locomo-

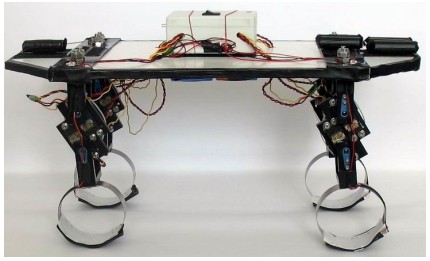
tion are well-suited for environments designed for humans and can adapt to diverse terrains. This adaptability is inspired by the prevalence of bipedal and quadrupedal walking in the biological world. Examples of robots with 2, 4 and 6 legs, including RH5, Charlie and CREX are depicted in Figures 1.1a to 1.1c respectively.

(a) Bipedal walking humanoid RH5.

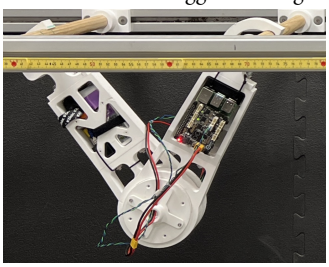


(b) Charlie is a primate-inspired robot capable of bipedal and quadrupedal locomotion.

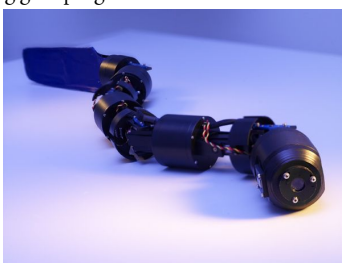
(c) The CREX is a six-legged walking robot.



(d) The four legged robot Pithekos II moves using galloping locomotion.



(e) The Acromonk robot can utilize the hook to brachiate.



(f) The Mehen robot traverses in water by using slithering locomotion.

Figure 1.1: Robots developed at the DFKI are equipped with diverse locomotion mechanisms that draw inspiration from biological systems. (Image sources: DFKI GmbH).

As shown in Figure 1.1, a spectrum of locomotion patterns is exhibited by robots developed at German Research Center for Artificial Intelligence (DFKI). The RH5 (Eßer et al. 2021) robot utilizes a bipedal walking mechanism, resembling human locomotion. In the case of the Charlie (Kuehn et al. 2017) robot, the utilization of four limbs during locomotion is predominant; however, it possesses the capacity to employ solely the hind limbs for bipedal walking. By contrast, the CREX (Machowinski et al. 2017) robot relies on multiple limbs, a trait that confers upon it inherent stability, which is absent in bipedal walking mechanisms. The Pithekos II robot is equipped with four legs and employs a galloping motion to facilitate movement. Additionally, the Acromonk (Javadi et al. 2023) robot utilizes a brachiating motion to traverse hanging bars, while the Mehen robot, an aquatic robot, employs a slithering movement underwater.

Bipedal walking can also be achieved using passive dynamics (McGeer 1990), a method that has been demonstrated to be highly energy efficient. This mechanism is capable of walking on small slopes using only gravity. Additionally, active energy can be utilized to actuate this design. Passive walking systems exploit the inherent dynamics of their structure and the environment to achieve longitudinal motion. These mechanisms conserve energy by converting it between potential energy and kinematic energy in a cyclic manner, thereby generating periodic motion that, by design, produces natural and stable locomotion.

It is not strictly necessary for a robot to rely on a single locomotion mechanism; in fact, it is often advantageous for a robot to possess multiple locomotion modes, as will be discussed in the following section.

1.3 Hybrid Locomotion

A particular mode of locomotion is optimized for a specific environment, exhibiting both suitability and efficiency. For instance, wheeled locomotion is well-suited for flat terrain and slightly uneven terrain. However, in highly uneven terrain or when navigating steps or a soft substrate, wheeled locomotion can become inefficient and potentially unsuccessful. In such instances, legged locomotion may prove to be a more viable option. Conversely, legged locomotion on a flat, even terrain is remarkably inefficient.

To address this challenge, a combination of different locomotion types can be employed, leveraging the strengths of each mode. This approach is referred to as "hybrid locomotion", involving the integration of multiple distinct modes of locomotion. This approach enables robots to adapt to diverse environments by switching or using multiple modes of locomotion concurrently. By employing hybrid locomotion, the robot can select the most suitable locomotion mode for the prevailing environment, thereby enhancing its adaptability and efficiency. The versatility of hybrid robots renders them well-suited for applications such as search and rescue, planetary exploration, and other missions where challenging and diverse terrains are commonplace.

A taxonomy of hybrid locomotion robots is predicated on the combination of distinct modalities. According to the survey conducted by Russo and Ceccarelli 2020, hybrid robots can be classified into two categories: reconfigurable and non-reconfigurable. *Reconfigurable* hybrid robots deploy the same components to generate distinct modes of locomotion. In contrast, *non-reconfigurable* hybrid robots possess distinct mechanisms that generate varied locomotion modes. The mobility of these hybrid robots can be exploited to traverse diverse environments, including land (terrestrial), air (aerial) and water (aquatic or sub-aquatic).

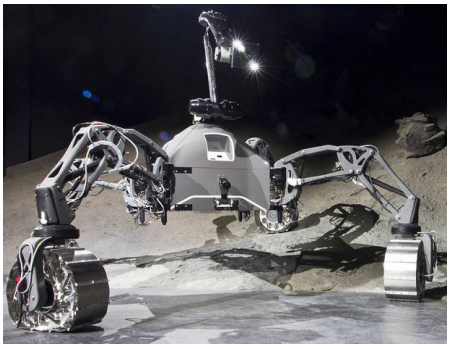
The focus of this work is hybrid terrestrial robots, with a particular emphasis on wheeled-legged robots, which are equipped with wheels affixed to the articulation points of their legs. Prominent examples within this category include SherpaTT (Cordes and Babu 2016), Handle, ANYmal with wheels (Bjelonic et al. 2019), Skatebots (Geilinger et al. 2018), MAMMOTH (Reid et al. 2014) and ATHLETE (Wilcox et al. 2007). The functionality of these robots is characterized by a duality of purpose; their legs can either be utilized for locomotion by maintaining wheel-ground contact or can be employed to support the wheels in maintaining contact with the ground. Another possibility is to use pure legs in combination with wheeled-legs to overcome obstacles, as seen in walking excavator robots such as Autonomous Rough Terrain Excavator Robot (ARTER) (Babu et al. 2022) and HEAP (Jud et al. 2021), which use their manipulator to serve the purpose of a leg, to support its locomotion. Figure 1.2 shows some of

(a) Handle (b) ANYmal (c) Skaterbots (d) MAMMOTH

(e) RHex (f) ATHLETE (g) TeleMAX

Figure 1.2: Examples of robots with hybrid locomotion capabilities (Image sources: Bjelonic et al. 2019; Boston-Dynamics 2017; Chiou et al. 2022; Geilinger et al. 2018; NASA 2009; Reid et al. 2014; Saranli et al. 2004a respectively).

(a) The Coyote3 robot (R. U. Sonsalla et al. 2015) is a passive hybrid robot with a hybrid legged wheel.



(b) SherpaTT robot (Cordes and Babu 2016) with adaptive legged wheels is designed for planetary exploration.

(c) ARTER (Babu et al. 2024) is a walking excavator robot with adaptive legged wheels and a manipulator which can support locomotion.

Figure 1.3: Robots developed at DFKI with hybrid locomotion capabilities (Image sources: DFKI GmbH).

the most popular hybrid locomotion robots that are used in the industry or by research community. DFKI GmbH (Robotics Innovation Center (RIC)) has developed a number of hybrid locomotion robots, some of which are shown in Figure 1.3. The subsequent subsections provide a detailed exposition on the specific category of passive hybrid robots and a selection of instances that demonstrate the efficacy of hybrid locomotion.

1.3.1 Passive Hybrid Locomotion

Passive hybrid locomotion robots are characterized by the absence of active separation or switching between different locomotion modes, a distinction that is typically achieved by the inherent properties of the robot design. These robots are often distinguished by their simplicity in design, ease of control and energy efficiency. However, it should be noted that the performance and efficiency achievable through simple control of such robots may not always be optimal. Examples of passive hybrid locomotion robots include Coyote3 (R. U. Sonsalla et al. 2015) and RHex (Saranli et al. 2004a).

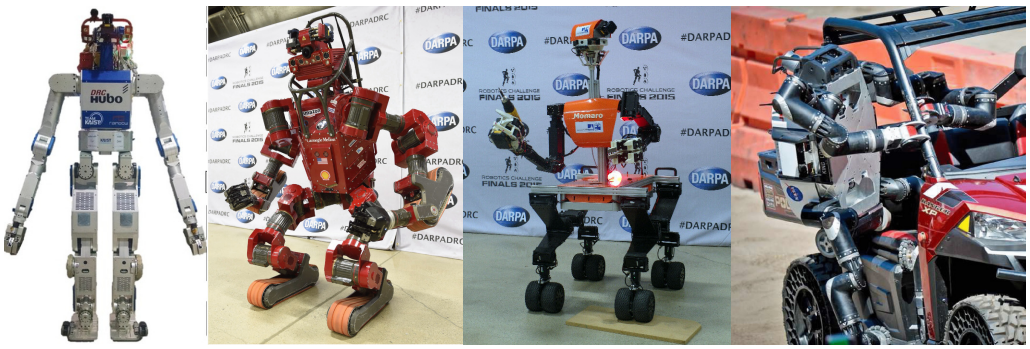
The robot Coyote3 is derived from the robots in Asguard series, but with very similar locomotion mechanisms. Asguard is one of the robots for which controllers are developed in the thesis. The primary design emphasis is on the unique wheel configuration, which integrates a combination of wheel and leg functionality. The control input is limited to a single Degree of Freedom (DoF) per wheel, totaling four DoFs, augmented by a passive joint within the body to ensure consistent contact with the ground. The robot's locomotion capabilities are noteworthy for their versatility, enabling navigation on flat surfaces, climbing stairs, and traversing rough terrain with relative ease, despite the simplicity of the controls. However, the system is not without its limitations, including a loss of efficiency, the potential for increased shocks and vibrations and challenges in turning. These limitations are further elaborated upon in the following chapter (Chapter 2).

RHex, developed by Boston Dynamics, is a passive hybrid robot that has been the subject of several studies (Komsuoglu et al. 2001; Saranli et al. 2001). It possesses six wheels, each of which is shaped like a "C", and during movement, at least three of these wheels are in contact with the ground, thereby providing stability. Despite its relatively simple controls, RHex is capable of traversing rough terrain. Numerous controls were implemented and assessed on this platform. These controls include those outlined by Moore et al. 2002, Campbell 2004, Weingarten et al. 2004 and Saranli et al. 2004a. RHex is shown in Figure 1.2.

1.3.2 Success of Hybrid Locomotion

Terrestrial hybrid locomotion offers numerous advantages in challenging terrains by virtue of its capacity to adapt or switch locomotion to suit the terrain type being traversed. This has led to extensive use of hybrid locomotion robots in outdoor mobile robots in rough terrain. The efficacy of hybrid locomotion designs has been demonstrated in a variety of scenarios and events.

One such event is the DARPA Robotics Challenge (Krotkov et al. 2018), which was initiated in 2012 and culminated in June 2015. The competition involved numerous teams, with a designated focus on disaster and emergency response scenarios. The robots were required to perform a series of timed tasks in semi-autonomous mode and the teams were ultimately ranked based on the number of successful tasks completed and the time taken to complete all tasks. Of the twenty-five teams that participated in the finals, four of the top five teams utilized hybrid locomotion. The robots HUBO (Team KAIST) (Lim et al. 2017), CHIMP (Team TARTAN RESCUE), Momaro (Team NIMBRO RESCUE) and



(a) HUBO (b) CHIMP (c) Momaro (d) RoboSimian

Figure 1.4: Robots with hybrid locomotion capabilities at DARPA robotics challenge 2015. (Image sources: Haynes et al. 2017; Karumanchi et al. 2017; Lim et al. 2017; Schwarz et al. 2016 respectively)

RoboSimian (Team ROBOSIMIAN) were positioned in the top four, and are equipped with multiple modes of locomotion. With the exception of Momaro, all of these robots are classified as bipedal, with the capability of utilizing wheels to traverse extended distances on flat terrain. Momaro, a wheeled-legged robot, utilizes wheels for locomotion to a greater extent than its counterparts. Figure 1.4 shows the top finalists at DARPA Robotics Challenge with hybrid locomotion capabilities.

Figure 1.5: Walking excavator in extreme terrains where the wheeled-legged design is used to adapt to terrain (left[1]) as well as using the manipulator on slopes (right[2]).

Another such use case in which hybrid locomotion has been effectively utilized is the design of walking excavators, also known as “spider excavators”. These machines are engineered for use in mountainous terrain. This is accomplished through the integration of a legged-wheeled hybrid design. Furthermore, the manipulator can be utilized to traverse obstacles or ditches. The manipulator’s functionality extends to slippery slopes, where the wheels experience insufficient traction. As illustrated in Section 1.3.2, the walking excavator’s efficacy is evident in scenarios where traversing is challenging due to extreme terrain conditions.

[1]Image source: GT1976, 2018-10-30 (851) Walking excavator Menzi Muck M545 at Mariazellerbahn in Rabenstein an der Pielach, Austria, CC BY-SA 4.0 LEGAL CODE

[2]Image source: HH58, Schreitbagger Kaiser SX - 1, CC BY-SA 4.0 LEGAL CODE

This section explored the utilization of hybrid locomotion in robots. The subsequent section will offer a comprehensive review of the existing literature concerning controllers and control solutions that have been developed for utilization with such robots.

1.4 Related Works

A wide array of solutions has been devised to address the domain of hybrid locomotion in the context of terrestrial mobile robotics. These solutions encompass both customized solutions tailored to the specific characteristics of the robot and automated solutions that learn the behaviors necessary for traversal. The subsequent sections delve into the various types of these solutions and offer illustrative examples. It is evident that these classifications are not mutually exclusive, as some controllers may employ a combination of multiple classes of solutions. The ensuing subsections furnish a comprehensive survey of the diverse control solutions that have been devised for hybrid locomotion robots, which are classified into the following categories: robot-specific solutions, model-based controllers, sampling-based controllers, optimal-control controllers and intelligent controllers. Thereafter, a qualitative comparison of several state-of-the-art solutions is presented, with the evaluation conducted on the basis of several criteria.

1.4.1 Robot-Specific Solutions

Hybrid locomotion robots are known to occasionally possess distinctive designs, precluding the possibility of generalization. This characteristic gives rise to solutions that are tailored to the specific system and demonstrate significant efficacy within the intended framework. In many instances, robot design is unique or capable of undergoing transformations that result in alterations to the system's morphology during deployment.

Examples of hybrid locomotion mobile robots with unique design are Asguard (Eich et al. 2008a), RHex (Saranli et al. 2001) and WheTLHLoc (Bruzzone et al. 2021) whose designs do not adhere to common morphologies. The stair climbing controller for Asguard in Eich et al. 2008b uses predefined motion patterns which are adapted specifically for the scenario by using proprioceptive information to change the parameters. Several customized solutions for the RHex robot have been developed and evaluated for scenarios including stair-climbing (Moore et al. 2002), upright balancing (Altendorfer et al. 2004), flipping (Saranli and Koditschek 2010) and flat terrain navigation (Burzyński et al. 2024). Step-climbing controller for WheTLHLoc is described in Bruzzone et al. 2024 where the phases are designed by hand and analytically solved for the kinematics.

Some robots physical transform their morphologies such that different control solutions can be deployed based on the currently active locomotion. Quattroped (S.-C. Chen et al. 2013), RHex-T3 (C. Sun et al. 2023), wheel-leg hybrid robot (Tadakuma et al. 2010), STransleg (Wei et al. 2023) and HyTRo-I (Lu et al. 2013) are robots which are able to completely transform themselves to separate morphologies and thereby apply different locomotion modes. The transition capability of the Quattroped robot between legged and wheeled modes makes it effective for several scenarios. A customized solution for planning the trajectories is developed in S.-C. Chen et al. 2011. RHex-T3, apart from being able to use the original RHex mode, can transform itself to be similar to legged mode or wheeled mode, each of which brings certain advantages. With integrated legs and wheels STransleg robot is capable of generating four distinct locomotion mode: four-wheeled, two-wheeled, four-legged and carriage, each of

which can be controlled separately. HyTRo-I has a similar separate wheel and leg setup resulting in three different locomotion modes.

Robot-specific control solutions are designed to exploit the specific locomotion mechanisms of the robot, ensuring optimal functionality within the intended environment. Furthermore, in the majority of cases, these solutions are also not computationally expensive. However, it should be noted that the implementation of such solutions can present certain challenges in terms of transferability to different robotic designs.

1.4.2 Model-based Controllers

Modeling of kinematics and dynamics of mobile robots is necessary for trajectory planning, design of controllers, simulation, etc. Full-order dynamics captures all the dynamics including all the forces and torques acting on the system. Newton-Euler equations or Lagrangian mechanics are used to derive the models and then the parameters estimated using system identification algorithms. Unfortunately, full-order dynamics is mostly complex and not tractable for real-time algorithms. Hence, reduced-order models are used to capture the main aspects of the model which are relevant for the scenario.

A number of reduced-order models have been developed to have stable yet agile motions for floating base robots. Most basic one is the static stability based on Center of Mass (CoM) which was projected on to the support polygon for estimating the tip-over stability. Further developments for stability margins include Energy Stability Margin (ESM) which is the potential energy required to tip-over a robot. Normalized Dynamic Energy Stability Margin (NDESM) (Garcia and De Santos 2005) extends this notion with the forces acting on the CoM. Development of Zero Moment Point (ZMP) (Vukobratović and Borovac 2004) based on Center of Pressure (COP) which in turn is a generalization of CoM delivered a model which turned out to be widely used for walking robots. Another dynamic model, centroidal dynamics (Orin et al. 2013), uses the robot's aggregate linear and angular momenta over time instead of its full-body dynamics. With the introduction of Hybrid Zero Dynamics (HZD) (Westervelt et al. 2003) models which incorporates both the continuous motion and discrete contacts in a lower-dimensional surface, nonlinear feedback controllers to generate stable locomotion could be generated.

Tip-over stability based on ZMP being constrained within the support polygon has been used in numerous locomotion controller for hybrid robots. The main purpose is to ensure stability during locomotion as well as during transitions between locomotion modes. Controller developed for ANYmal with wheels (Bjelonic et al. 2019; Viragh et al. 2019), RoboSimian (K. Byl and M. Byl 2015) and a quadruped bionic robot in Pengfei et al. 2005 uses ZMP to generate stable gaits or feedback controllers. More dynamics systems with multiple locomotion modes especially with varying contact constraints use HZD models or variants of this model. Examples of some bipedal robots include Cassie (Gong et al. 2019) and DURUS (Reher et al. 2020).

Controllers based on either complete models of the system, reduced order models or hybrid models tend to exhibit good performance characteristics while generating motions that are stable and optimal. The performance is yet heavily dependent on the fidelity of the models, resulting in degradation of performance due to unmodeled dynamics. Additionally, model identification, high computational cost, difficulty in design and implementation, etc. are certain cons of using model-based controllers.

1.4.3 Sampling-based Solutions

Controllers based on sampling address complex environments by generating and evaluating samples from the robot's control space to identify a solution. Typically, the controller attempts to ascertain the control inputs necessary to transition from one state to a desired state. The feasible path is determined by circumventing states or connections between states that are not feasible. Feasibility is generally defined by factors such as collision, under-actuation, joint limits and so on. However, constraints such as under-actuation and non-holonomic, which are predicated on continuous motion, can pose significant integration challenges in sampling-based controllers. Conversely, configuration-based constraints are often more straightforward to incorporate.

In scenarios characterized by rough terrain, where robots must adapt to local conditions through the planning of footstep and body posture, sampling-based solutions have gained significant popularity. However, these approaches are hindered by their computational inefficiency, stemming from the high dimensionality of the state space. Consequently, most research focuses on multi-stage or hierarchical planning, which initially generates a coarse plan to direct subsequent, more refined planning stages. Solutions for the planetary rover ATHLETE are provided in Hauser et al. 2008b and Hauser et al. 2008a, where the search alternates between stance search and transition search. In stance search, the contact or non-contact of the foot with ground is determined, and the continuous path between the stances is generated in transition search. In contrast, the hierarchical planning setup proposed by Reid et al. 2020 for the MAMMOTH rover is asymptotically optimal. This setup utilizes prior knowledge about the subspaces to bias the sampling distribution, thereby enhancing planning efficiency. In a similar vein, Brunner et al. 2013 study on hierarchical motion planning for the Telerob Telemax search and rescue robot initially plans the rough segments using only the robot's operating limits, subsequently refining the plan within the complete state space.

In the context of high-dimensional and complex environments, sampling-based approaches have performed well, offering a diverse array of applications with a relative ease of implementation. However, these approaches are deficient in their inability to effectively handle complex dynamics and are not deterministic, resulting in variations in outcomes despite identical inputs. Furthermore, despite the computational demands of these approaches, the resulting paths frequently exhibit a lack of smoothness and optimality.

1.4.4 Optimal Control Solutions

Optimal controllers are defined as controllers that attempt to solve a set of equations iteratively through the use of optimization algorithms. These equations model the relationship between the control input and the variables that are to be controlled. In the context of robotic systems, the inputs typically encompass the reference joint states (position, velocity, or torque), with the outputs representing the desired motion of the robot within its environment. In addition to the aforementioned equations, supplementary constraints can be stipulated, akin to those inherent in sampling-based methodologies. However, constraints pertaining to collision avoidance can prove challenging to incorporate within the framework of optimal control. There are different types of solvers available for solving the control problem. The selection of a particular solver is contingent upon the nature of the control equation and the imposed constraints. In general, linear solvers are designed to handle linear equations and constraints. Conversely, quadratic solvers are capable of handling quadratic control equations and linear constraints. Nonlinear solvers, on the other hand, are capable of handling nonlinear control equations. The selection of a particular solver is determined by the characteristics of the system and

the nature of the control problem. The temporal demands of the problem, particularly the necessity of real-time solutions, must also be taken into account when selecting an appropriate solver.

A multitude of optimal control formulations for the locomotion of hybrid locomotion robots have been documented in the literature. In instances where the computation time is less than the sampling time of the controller, the optimal controller can be implemented in the form of a Model Predictive Control (MPC) algorithm. In such cases, the iteration is repeated at each sampling time, thereby providing an updated state of the system at each time step. Traditionally, MPC relies on the dynamic model of the system being controlled. Another computational tool for optimal control that is being widely researched in robotics is Trajectory Optimization (TO), in which the model of the robot is used to generate a set of trajectories that satisfies a set of constraints. Once the control problem is formulated with decision variables, objectives and constraints, solvers for non-linear programming can be used to optimize the trajectories. It is possible to employ TO in either an open-loop or a MPC configuration, wherein solely the initial control output of the trajectory is transmitted to the robot.

A control framework based on TO is developed in Medeiros et al. 2020 for the wheeled-legged ANYmal. The trajectories, including the position of the robot base and the legs, are formulated as a non-linear programming problem using a reduced-order robot model, terrain map and stability constraints. The optimized trajectories are then executed on the robot using a hierarchical whole-body controller. In contrast, the real-time solution outlined in Bjelonic et al. 2020 involves the decomposition of the problem into distinct optimizations for the wheels and the base. An earlier work, Bjelonic et al. 2019, proposes a methodology where gaits are first selected from a library and footholds are subsequently optimized and fed into the motion optimizer. In the case of the HEAP robot, the solution for terrain adaptation in Jelavic and Hutter 2019 and Jelavic et al. 2020 utilizes TO as the planner, employing gait patterns and contact schedule optimization. A mixed solution where sampling based plan is further refined by TO is developed in Jelavic et al. 2023. The TO method is employed to generate more dynamic motions, including skidding, as outlined in Bellegarda and K. Byl 2019; Geilinger et al. 2018. Researchers have developed MPC-based controllers for wheeled-legged robots in the literature such as Bjelonic et al. 2021, J. Yu et al. 2023 and Pan et al. 2023.

Locomotion can be envisaged as the application of contact forces by robot limbs on an environment, irrespective of the end-effector. Redundancy in actuation allows for the simultaneous formulation and solution of multiple tasks, each pursuing a distinct objective. The controller, designated as Whole Body Control (WBC) (Sentis and Khatib 2006), can be cast as either an inverse kinematic or an inverse dynamic problem, with optimal control providing the solution. A number of prioritized tasks, including the corresponding constraints, can be formulated as a Quadratic Programming (QP) problem or a non-linear program and solved in a cascaded fashion. Some common tasks for locomotion of robots include balancing, floating base motion, motion tracking of CoM, limb motion tracking, obstacle avoidance, etc., with typical constraints including joint limits, torque limits, contact forces, friction constraints, etc. Hierarchical WBC has been employed in numerous robots for legged locomotion (Henze et al. 2015; J. Li et al. 2022; M. Liu et al. 2016) as well as hybrid locomotion (Bellicoso et al. 2018).

Controllers based on optimization have become a predominant approach for the locomotion of complex robotics systems, owing to their demonstrated performance and generalizability. The trajectories that are generated to satisfy the objective and constraints are smooth and optimal in nature. However, this approach is computationally intensive and requires both an explicit and accurate model of the robot system and an accurate model of the interaction with the environment. In the event that the model exhibits significant deviations from reality, the performance of the controller undergoes a

substantial degradation. Additionally, identifying a global minimum for the control problem is challenging, with the solution heavily reliant on the initialization provided for the decision variables.

1.4.5 Intelligent Controllers

Advancement in Artificial Intelligence (AI) and other machine learning techniques have enabled the development of controllers that are capable of applying these techniques to learn the policies for locomotion. As opposed to the analytical approaches mentioned so far, intelligent controllers do not need explicit models of the robot or the environment. Some intelligent control methods including Central Pattern Generators (CPG), evolutionary algorithms, etc. are inspired from biological systems. CPG (Marder and Bucher 2001) are neural circuits that, based on simple low-dimensional inputs, produces rhythmic patterns that can translate to locomotion including walking, running and swimming. These pattern generation method has been in used in several robot locomotion solutions (Eich et al. 2008a; Ijspeert 2008). Evolutionary algorithms are widely used to tune the parameters of locomotion controllers. There are other rule-based solutions which uses fuzzy logic, decision-trees etc. to create the controllers especially when multiple modes are involved and switching between different modes is required. Even though widely used for legged locomotion, limited research is available on the use of rule-based or evolutionary algorithms for hybrid locomotion robots.

Recently there are several control solutions developed which are based on artificial intelligence to learn the controllers. Most of these solutions make use of neural networks to represent the controllers, known as Deep Learning. Reinforcement learning is the method of finding out the actions a robot can take to maximize the rewards it obtains in an environment. If reinforcement learning is used to get the parameters of the neural networks, it is known as Deep Reinforcement Learning (DRL). The agent in a DRL is trained by observing the environment where the robot can take actions and obtain rewards. The policies which are the mapping from observations to actions, and the values functions which estimate the future rewards, are represented by deep neural networks. The networks are then trained using the data received by applying trial-and-error actions in the environment by the agent. DRL has been used in several aspects of robotics including perception, manipulation, control, interaction and locomotion. One of the main challenges in deep reinforcement learning is to design the reward function such that robot is able to learn the controller effectively.

Research towards use of DRL for hybrid locomotion robots is gaining popularity. Even though most of the work is focussed on wheeled-legged systems, developing gaits for other morphologies (J. Shi et al. 2020) is also being investigated. The solution in Baek et al. 2022 combines classical Linear Quadratic Regulator (LQR) with ensemble DRL to learn the controller for a wheeled humanoid robot. The ensemble DRL reduces the variance of the reinforcement learning by using multiple Soft-Actor-Critic (SAC) agents. Reinforcement learning based control for training the highly dynamic robot *evoBOT* is developed in Klokowski et al. 2023 to train the robot to perform balancing and dynamic locomotion. Mobility and navigation of a wheeled-legged robot is addressed using Hierarchical Reinforcement Learning (HRL) in Lee et al. 2024 for logistics application. The multiple layers that are trained are the mobility-aware navigation controller and the locomotion controller which are both networks trained using DRL. The tasks including the planning and path following is performed by the navigation controller which in turn sends the way points to the locomotion controller which generates the necessary gaits for the robot. Challenges in DRL including the need for elaborate designs of reward functions is addressed in Schwarke et al. 2023 by using intrinsic motivation with sparse rewards to perform simultaneous locomotion and manipulation tasks.

Intelligent controllers provide several advantages over analytical solutions by reducing the effort necessary in developing a model of the system and being adaptable to different scenarios without the need for explicit reprogramming. However, the performance of these methods do depend on the quality of the engineered rewards functions and the environment it is trained on. Additionally, these methods are also affected by generalization issues, interpretability limitations and safety concerns. As it is obvious from the literature, solutions are being developed to address these challenges and make the intelligent controllers more robust and reliable.

1.4.6 Comparison of the State-of-the-art Solutions for Hybrid Locomotion

The diverse categories of solutions employed for hybrid locomotion have been discussed with the aid of representative examples. In practice, the majority of these solutions employ a combination of these methods to develop the controllers, thereby amplifying or attenuating the advantages and disadvantages of the involved methods. A comparison of different controllers is necessary to understand the broad spectrum of solutions available and to identify the gaps in the research. Due to the vast array of robots and controllers, a direct comparison of their performance or the generated locomotion gaits is impractical. Consequently, the comparison in this section is based on the general characteristics of the controllers and robots. The evaluation criteria and the assessment itself are based on the author's subjective analysis of existing literature and research in the field.

A qualitative comparison of the state-of-the-art solutions for hybrid locomotion is presented in Table 1.2 with information including the details of the publication, details of the robot (name and type), type of controllers, brief description of the controller and the assessments for comparison. The criteria for the comparison of the controllers are as follows:

1. **Adaptability and Generalization (AG):** This criterion encompasses the ability and the effort necessary of the solution to be adapted to develop different controllers and for varying scenarios.

2. **Transferability and Scalability (TS):** The capability of the solutions to be transferred to a more complex system or another robot with different morphology is incorporated in this criterion.

3. **Development Complexity (DC):** The amount of effort and domain knowledge necessary for the design and implementation of the controller is captured here.

4. **Computational Requirements (CR):** This term encapsulates the computational resources necessary for the controller including training, tuning and real-time deployment.

Each criterion is assigned a rating from 1 to 5. The numerical values are represented by the number of '+' signs, where a higher value indicates a higher level of performance of the controller with respect to the criterion. The objective of this comparison is to provide a comprehensive overview of the various types of controllers and the robots on which they are employed. It should be noted that this comparison is not exhaustive; readers are encouraged to refer to the original publications for more detailed information. A comparative analysis of the performance of robots with distinct locomotion characteristics is provided in Wirkus et al. 2024a and Wirkus et al. 2024b.

Table 1.2: Comparison of the state-of-the-art solutions for hybrid-locomotion of robots.

Publication	Robot/Type	Controller Types*	Comparative Criteria‡				Remarks
			AG	TS	DC	CR	
"Reliable Stair Climbing in the Simple Hexapod 'RHex'", 2002 (Moore et al. 2002)	RHex / Passive hybrid hexapod with half-circle legs	CD	+	+	+++++	+++++	Locomotion is based on open loop gait patterns which are manually designed and adjusted for different scenarios.
"Model-based dynamic self-righting maneuvers for a hexapedal robot", 2004 (Saranli et al. 2004b)	RHex	MC, CD	+	+	+++++	+++++	Develops a robust flipping behavior using model-based feedback controller and hybrid pumping strategy to overcome torque limits.
"Asguard: A hybrid legged wheel security and sar–robot using bio–inspired locomotion for rough terrain", 2008 (Eich et al. 2008a)	Asguard / Four wheeled passive hybrid with spiked wheels	IC (CPG), CD	++	+	+++++	+++++	CPG trajectories are adapted based on the proprioceptive data of the legs.
"Motion planning for a six-legged lunar robot", 2008 (Hauser et al. 2008a)	ATHLETE / Six-legged lunar robot	SB (Graph search)	++	+++	+++	++	Planner utilizes a two-stage search with exploration of stance (mapping of feet to footfalls) and transition (stepping from one stance to another) graphs.
"Trajectory planning for stair climbing in the leg-wheel hybrid mobile robot quattroped", 2011 (S.-C. Chen et al. 2011)	Quattroped / Quadruped with transformable wheel or half-circle leg	CD	+	+	+++++	+++++	Trajectory planning is based on the geometric model of interaction between the legs and stairs.
"Hierarchical rough terrain motion planning using an optimal sampling-based method", 2013 (Brunner et al. 2013)	Telerob Telemax / Actively reconfigurable tracks	SB (Graph Search, Biased RRT*), HD	++	+++	+++	++	Hierarchical motion planning where a rough low-dimensional plan is refined using a high-dimensional planner.

Continued on next page

Table 1.2: Comparison of the state-of-the-art solutions for hybrid-locomotion of robots. (Continued)

"Anytime Hybrid Driving-Stepping Locomotion Planning", 2017 (Klamt and Behnke 2017)	Momaro / Wheeled-legged Quadrupedal		+++	++	+++	+++	Implements a sampling-based hybrid locomotion planner combining both driving and stepping locomotion where an abstract path from start to goal pose is expanded using step motion generator.
"TurboQuad: A Novel Leg–Wheel Transformable Robot With Smooth and Fast Behavioral Transitions", 2017 (W.-H. Chen et al. 2017)	TurboQuad / Quadruped with transformable wheel or half-circle leg	IC (CPG), CD	++	+++	++++	+++++	A Hopf oscillator generates the patterns of individual leg–wheel and overall synchronization is achieved using virtual springs.
"Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels", 2018 (Geilinger et al. 2018)	Skaterbots : Primarily wheeled-legged robots	OC (TO), MC (Centroidal Dynamics)	+++	++++	++	++	Optimization of the robot morphology based on the motion generated (walking, skating, rolling, etc.) using a novel TO formulation.
"Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Using Linearized ZMP Constraints", 2019 (Viragh et al. 2019)	ANYmal with wheels / Quadruped legged wheel	OC (Cascaded TO), MC, HD	+++	++++	++	++++	Formulates a cascaded trajectory optimizer for complex motions including driving, walking and turning by solving for angular, vertical and planar components of base and feet.
"Keep Rollin'—Whole-Body Motion Control and Planning for Wheeled Quadrupedal Robots", 2019 (Bjelonic et al. 2019)	ANYmal with wheels	OC (SQP), MC (ZMP), WBC, HD	++++	+++	+	++	The reference trajectories for the dynamic motion of the robot is generated using ZMP-based motion optimizer which is in turn tracked by a hierarchical WBC.
"Whole-Body Motion Planning for Walking Excavators", 2019 (Jelavic and Hutter 2019)	HEAP / Walking Excavator Robot	OC (TO, NLP)	++++	+++	++	+	A trajectory planning framework as NLP for a walking excavator in rough terrain by using a novel stability constraint formulation.

Continued on next page

Table 1.2: Comparison of the state-of-the-art solutions for hybrid-locomotion of robots. (Continued)

"Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip", 2019 (Bellegarda and K. Byl 2019)	Robosimian / Wheel-legged quadrupedal with passive wheels	OC (TO, NLP)	++++	++++	+++	++	Develops a TO framework for dynamic motion of a wheel-legged system without enforcing non-slip or non-skid conditions.
"Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover", 2020 (Reid et al. 2020)	MAMMOTH / Quadrupedal with adaptive wheels	SC, HD	++	+++	+++	++	Performs locomotion planning on cluttered environment using sampling based methods by utilizing a hierarchical structure for multiple subspaces and incorporate prior knowledge to improve performance.
"Rolling in the deep–hybrid locomotion for wheeled-legged robots using online trajectory optimization", 2020 (Bjelonic et al. 2020)	ANYmal with wheels	OC (TO), WBC, HD	+++	+++	+	+++	Online TO framework for walking-driving locomotion by separating wheel and base trajectories and tracking using hierarchical WBC.
"Deep Reinforcement Learning for Snake Robot Locomotion", 2020 (J. Shi et al. 2020)	Wheeled snake robot	IC (DRL)	+++	+++	++++	++++	Generates gait for traversal in different environments for a snake like robot by employing DRL.
"Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Driving in Challenging Terrain", 2020 (Medeiros et al. 2020)	ANYmal with wheels	OC (TO), WBC, HD	+++	++++	++	++++	Terrain aware TO which can generate optimal motion over challenging terrain with low computational complexity.
"Terrain-Adaptive Planning and Control of Complex Motions for Walking Excavators", 2020 (Jelavic et al. 2020)	HEAP	OC (TO), WBC, HD	+++	++++	++	++	Develops TO for generating references for end-effectors and joints which is then executed using a terrain adaptive WBC.

Continued on next page

Table 1.2: Comparison of the state-of-the-art solutions for hybrid-locomotion of robots. (Continued)

"Whole-body MPC and online gait sequence generation for wheeled-legged robots", 2021 (Bjelonic et al. 2021)	ANYmal with wheels	OC (MPC), WBC, HD,	++++	++++	+++	++	Employs a single-task MPC which captures the dynamics and rolling constraints without using heuristics.
"Hybrid LMC: Hybrid Learning and Model-based Control for Wheeled Humanoid Robot via Ensemble Deep Reinforcement Learning", 2022 (Baek et al. 2022)	SATYRR / Wheeled humanoid	IC (DRL), OC (LQR)	+++	+++	++	++++	Proposes a hybrid learning and model-based controller which combines ensemble DRL with a classical LQR where the learning component tries to reduce the residual error.
"RHex-T3: A Transformable Hexapod Robot With Ladder Climbing Function", 2023 (C. Sun et al. 2023)	RHex-T3 / Transformable RHex	CD	+	+	++++	+++++	Physically transforms the robot to adapt to the type of terrain.
"Wheel-Leg Collaborative Control for Wheel-legged Robots Based on MPC with Preview", 2023 (Pan et al. 2023)	Knee-wheeled legged robot	OC (MPC), MC	++	+	++++	+++	Designs a MPC by using a simplified model of a quarter car for traversing uneven terrain.
"Modeling and MPC-Based Pose Tracking for Wheeled Bipedal Robot", 2023 (J. Yu et al. 2023)	Wheeled-bipedal robots	OC (MPC)	++	++	+++	+++	Implements a hierarchical MPC for pose tracking of the torso by using a simplified dynamic model and cascaded state estimator.
"evoBOT – Design and Learning-Based Control of a Two-Wheeled Compound Inverted Pendulum Robot", 2023 (Klokowski et al. 2023)	evoBOT / Dynamically stable two-wheeled robot	IC (DRL)	++++	++++	+++	+++++	Designs a dynamic two-wheeled robot and proposes DRL controller for balancing and dynamic locomotion.
"Curiosity-Driven Learning of Joint Locomotion and Manipulation Tasks", 2023 (Schwarke et al. 2023)	Wheeled-legged quadrupedal	IC (DRL)	+++++	+++++	++++	++++	Learns locomotion and manipulation tasks in an end-to-end fashion, driven by curiosity in DRL and utilizing only sparse rewards.

Continued on next page

Table 1.2: Comparison of the state-of-the-art solutions for hybrid-locomotion of robots. (Continued)

"Kinematic Analysis and Application to Control Logic Development for RHex Robot Locomotion", 2024 (Burzyński et al. 2024)	RHex	MC, CD	+	+	+++	++++	Develops control strategies and performance enhancements based on a detailed kinematic model of the robot.
"Learning robust autonomous navigation and locomotion for wheeled-legged robots", 2024 (Lee et al. 2024)	Wheeled-legged quadrupedal	IC (HRL, Privileged learning)	++++	+++++	++++	++++	Learns to navigate in challenging urban terrains using HRL (navigation policy and locomotion policy) and privileged learning.

* Controller types: CD - Custom Designed Controller, MC - Model-based Controller, SC - Sampling-based Controller, OC - Optimal Controller, TO - Trajectory optimization, IC - Intelligent Controller, HD - Hierarchical Design.

† Comparative criteria: AG - Adaptability and Generalization, TS - Transferability and Scalability, DC - Development Complexity, CR - Computational Requirements.

‡ The evaluation is based on the author's subjective analysis.

1.5 Motivation

A review of the table reveals several interesting patterns and trends in the field of hybrid locomotion. It is notable that the majority of earlier controllers were either based on the design of the robot or employed a customized model for developing the controller. Subsequently, a shift towards sampling-based solutions emerged, driven by the high dimensionality of the state space. Advances in optimal control and the availability of computational resources led to the preference for these solutions due to their superior adaptability, generalization and scalability. However, the necessity for explicit modeling and the rigidity of mathematical models have led to the development of intelligent controllers that address these challenges. While bio-inspired intelligent controllers have been employed for an extended period, deep learning-based controllers are gaining popularity due to their capacity to learn policies without requiring explicit models. Controllers are being developed that prioritize adaptability, generalizability and scalability, while requiring less computation and development complexity.

Another aspect pertains to the pervasive utilization of hierarchy and mixture of methods in controllers. Hierarchical structures facilitate the management of the curse of dimensionality by dividing the problem into disparate levels of abstraction and addressing them sequentially from the top down. The higher levels of this hierarchy generate a preliminary solution, which is subsequently refined by the lower levels. Mixture of methods employs the strengths of diverse methods to generate controllers. A prevalent combination involves the utilization of a sampling-based planner, which generates a preliminary plan that is subsequently refined by an optimal controller. However, this combination has the disadvantage of increasing the development effort and computational requirements.

There are several aspects of hybrid locomotion that are not yet covered in the literature. In the control of passive hybrid locomotion systems, efficiency of locomotion and optimization are rarely addressed. For active suspension systems, common tasks like ground adaptation do not have efficient or generic solutions that account for multiple objectives. Additionally, there is little work on long sequence tasks with multiple locomotion modes. Current solutions, which rely on sampling-based and trajectory optimization methods, require significant design and development effort, and often result in controllers that are complex to develop and are computationally inefficient.

It is evident that the prevailing trend is towards the development of more generic solutions, which possess the capacity for expeditious adaptation to diverse scenarios and robotic platforms. Nevertheless, the selection of the controller type is dependent on the requirements of the scenario, the available computational resources and the developer's domain knowledge. Guided by this understanding, the objectives of the thesis are articulated in the proceeding section.

1.6 Definition of the Objectives

The thesis presents the development of various control software frameworks and controllers designed for utilization with hybrid locomotion robots, namely *Asguard*, *SherpaTT* and *ARTER*. The distinct characteristics inherent to each robot necessitate a customized controller, thereby significantly influencing the design of these controllers, including the designated controller class.

Asguard, a passive hybrid locomotion robot with spiky wheels, exemplifies this variability, with its design dictating the controller's parameters. SherpaTT and ARTER are similar in their working principle; however, the former is designed for planetary exploration and has limited computational resources available to it. In contrast, the latter is equipped with advanced sensors and computing capabilities,

enabling it to perform tasks requiring environmental perception. Additionally, the latter is capable of utilizing its manipulator for locomotion. These distinguishing characteristics give rise to the following objectives:

O-1 **Improving locomotion efficiency of a passive hybrid robot:** The passive hybrid robot, Asguard, integrates the characteristics of legged and wheeled motion into a unified design, albeit with constrained actuation capabilities. The implementation of a rudimentary control scheme, such as pure wheel control, results in suboptimal locomotion. The subsequent sub-objectives have been delineated with the intention of enhancing the robot's locomotive performance:

- O-1a **Improve position control of the wheels:** The development of any enhancement to the locomotion system necessitates the implementation of a joint-level controller that is capable of ensuring precise synchronization of the wheels. A prerequisite for the development process is the necessity of modeling the mechanical system in its entirety, inclusive of the non-linearities and hysteresis.
- O-1b **Study the effect of wheel synchronization on locomotion efficiency:** The synchronization of the wheels during straight motion may influence locomotion, a phenomenon that necessitates investigation to ascertain the most efficient locomotion strategy for different substrates at different velocities.
- O-1c **Improve point-turns:** Asguard's capacity for executing point turns may be hindered under conditions of elevated lateral traction from the ground. The development of effective strategies to mitigate this challenge is crucial.

O-2 **Control architecture for hybrid locomotion control:** Hybrid locomotion robots, particularly those equipped with an articulated wheeled-legged system, possess a multitude of degrees of freedom, each with distinct controller functionalities, sensors and other characteristics. To ensure the effective deployment of these systems, it is essential to develop control architectures capable of meeting the specific requirements associated with each application scenario. The subsequent sub-objectives have been defined for the purpose of achieving this objective:

- O-2a **Control architecture for the planetary rover, SherpaTT:** The computational demands of planetary rovers constitute a critical constraint, necessitating the selection of controllers that exhibit minimal resource utilization and the implementation of only the indispensable functionalities.
- O-2b **Control architecture for ARTER robot with autonomy and remote-control functionalities:** The control framework for ARTER must possess the capacity to execute complex functionalities, ranging from teleoperation with assistance functions to full autonomy in limited scenarios.

O-3 **Terrain adaption controllers:** The primary function of the wheel-on-leg design of robots is to adapt to uneven terrain which is dependent on the system design. To this end, two distinct controllers with differing complexities must be developed for SherpaTT and ARTER.

- O-3a **Terrain adaption for SherpaTT using force-torque sensors:** The controller for SherpaTT has only limited resources available for execution, thus excluding the use of high-level perception sensors. Nevertheless, high-fidelity force-torque sensors on each wheel provide the possibility to implement the terrain adaptation controller based on the data generated during the interaction of the wheels with the ground.

O-3b **Height-map for terrain adaption of ARTER:** A terrain adaption controller that satisfy multiple objectives is necessary for ARTER to successfully traverse uneven terrain. The results of mapping using high-level sensors, including the height map and upcoming terrain changes, are available to the controller.

O-4 **Stepping controller using manipulator:** The manipulator of ARTER serves the additional purpose of supporting its locomotion, especially when the robot limbs temporarily lose contact with the ground during stepping. The execution of this controller is dependent on the configuration of obstacles and requires temporal coordination of numerous joints. The sequences of motion for stepping have repeated patterns, which are to be executed in a specific order over a long time horizon.

The objectives enumerated herein are addressed in various chapters of the thesis, which is outlined in the next section.

1.7 Structure of the Thesis

The overall structure of the thesis, along with the corresponding publications, is shown in Figure 1.6. The structure of the remaining chapters is outlined below.

The development of the controllers for the passive hybrid robot, Asguard, is comprehensively elaborated in Chapter 2, which corresponds to the objectives O-1a, O-1b and O-1c. This chapter encompasses a detailed description of the system and the challenges encountered in the control aspect. Notably, it incorporates an innovative cascaded torque controller. Thereafter, an evaluation of the impact of wheel synchronization on the efficiency of forward motion is conducted. Additionally, a controller designed to enhance the efficacy of point-turn maneuvers is described.

The details of the wheel-on-leg SherpaTT robot and its associated controllers can be found in Chapter 3. This includes a brief introduction to the system, followed by a detailed elaboration on the structure of the control architecture with focus on teleoperation. Subsequently, the equations that facilitate the ground adaption assistance function with the legged wheels are discussed. The chapter culminates with the presentation of the outcomes of the laboratory experiments and the analysis of the results. The contents of this chapter successfully address the objectives O-2a and O-3a.

The objective O-2b is addressed in Chapter 4. It commences with the presentation of the ARTER walking excavator robot, accompanied by the kinematic modeling of the parallel kinematics. The subsequent section delves into the design of the control software architecture which is followed by the section that delves into a trajectory follower controller designed with a MPC approach, including the evaluation of the outcomes from simulations. The study concludes with a discussion of two use cases, highlighting the robot's application in remote and semi-autonomous operations.

Intelligent controllers for ARTER are developed for autonomous adaptation to uneven terrain in Chapter 5. The discussion encompasses the configuration of the learning setup and the design of the controllers including observations, rewards and actions for the deep reinforcement learning setup. The chapter culminates with a presentation of the results and an analysis of the developed solutions, thereby demonstrating fulfillment of objective O-3b.

In Chapter 6, the development of a more complex stepping locomotion controller is discussed. The objective O-4 is hereby pursued. The chapter's content includes an introduction to the concept of

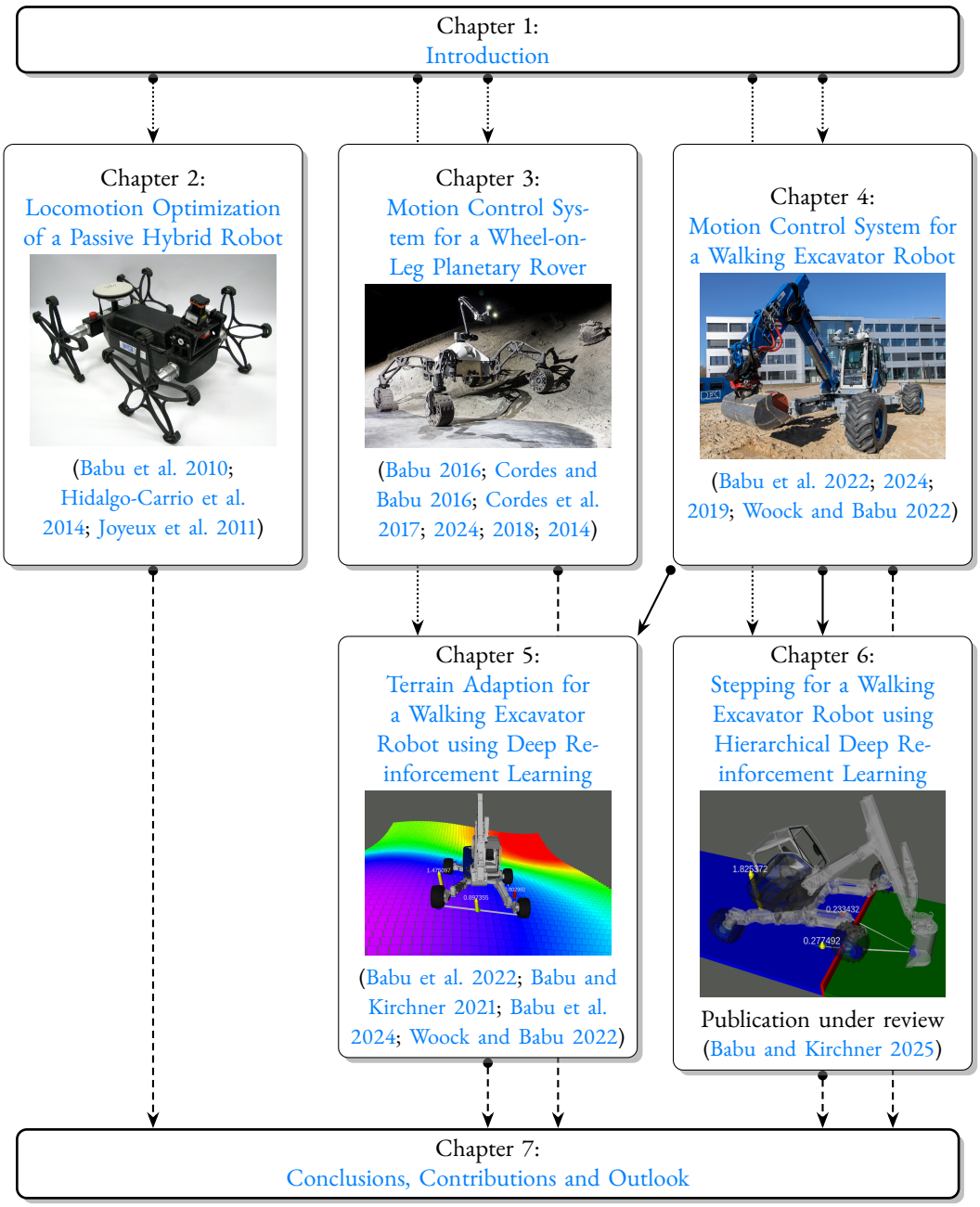


Figure 1.6: The chapter-wise structural breakdown of the thesis, along with the relevant interconnections and publications.

stepping in robotics; a review of the existing literature on the subject; and a foundation in mathematics for the techniques employed to address the problem. This introductory information is followed by the formulation of the setup, the design of hierarchies, and the design of individual controllers. The training results of the controllers at the lower and higher levels are presented.

Chapter 7 offers a synopsis of the work presented in the thesis, emphasizing the contributions made by the author. This is followed by an exploratory discussion of potential future directions and avenues for further research.

Locomotion Optimization of a Passive Hybrid Robot

This chapter discusses the research conducted on the locomotion of the passive hybrid robot, Asguard. The robot is equipped with a specialized wheel design that confers certain advantages typically associated with legged locomotion. This chapter delves into the studies undertaken to investigate the impact of such a design on locomotion and the methodologies employed to enhance the locomotion. Partial results of the presented work have been published in

(i) Hidalgo-Carrio, J., A. Babu, and F. Kirchner (2014). "Static forces weighted Jacobian motion models for improved Odometry". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, pp. 169–175

(ii) Joyeux, S., J. Schwendner, F. Kirchner, A. Babu, F. Grimminger, J. Machowinski, P. Paranhos, and C. Gaudig (2011). "Intelligent Mobility". *KI - Künstliche Intelligenz* 25:2, pp. 133–139. ISSN: 1610-1987. DOI: 10.1007/s13218-011-0089-8. URL: https://doi.org/10.1007/s13218-011-0089-8

(iii) Babu, A., S. Joyeux, J. Schwendner, and F. Grimminger (2010). "Effects of Wheel Synchronization for the Hybrid Leg-Wheel Robot Asguard". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2010), August*

The chapter commences with introduction to passive hybrid design in Section 2.1, followed by the system description in Section 2.2 which introduces the Asguard robot. This is followed by Section 2.3 elaborating on the individual controllers for motors and wheels. Subsequently, in Section 2.3.3, the computation of quasi-static force distribution for the legged-wheels is presented. Sections 2.4 and 2.5 then delve into the optimization of locomotion for longitudinal motion and turning motion, respectively. The chapter concludes with the summary and discussion.

2.1 Passive Hybrid Design

The Asguard system (Figure 2.1), developed at DFKI, was conceptualized with the objective of traversing unstructured and uneven terrains. It was part of the Intelligent Mobility (Joyeux et al. 2011) project, which sought to give the robot the ability to autonomously explore planetary or lunar surfaces. The robot's innovation lies in its hybrid leg-wheel design (Figure 2.1), which consists of five spike-like structures (Eich et al. 2008b). The hybrid wheel-leg design embraces both the simplicity of wheeled systems and the traversability of legged systems. The control of the motors, which is simple in nature, is sufficient to control the robot. This design principle can be extended to differential steering, facilitating the robot's maneuverability during turning maneuvers. The novelty of the design necessitates a com-

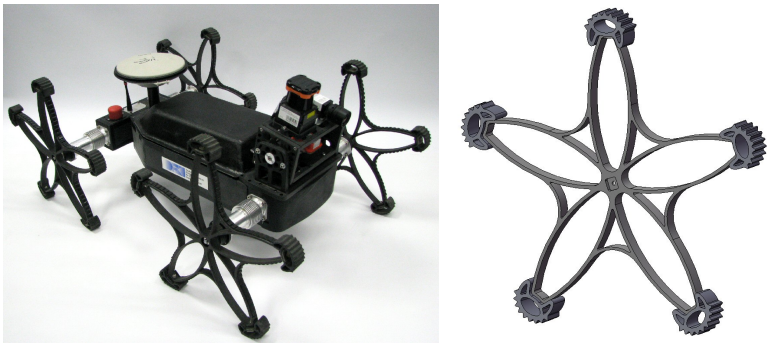


Figure 2.1: The Asguard robot is characterized by a distinctive hybrid leg-wheel design, incorporating a total of four passive wheels. Each wheel is composed of five spikes, which serve the purpose of legs.

prehensive understanding of the robot's locomotion to ensure the development of its capabilities and the establishment of its control requisites.

Leg-wheel hybrid designs exhibit certain advantages over pure leg or wheel designs. Primarily, its configuration is simple and robust, and it exhibits inherent stability in comparison to walking robots. Notably, even in the absence of power, the robot is capable of maintaining a stable posture, a capability that is comparable to that of wheeled robots. Secondly, leg-wheel hybrid designs with fewer DoFs exhibit enhanced navigation capabilities in challenging terrain, surpassing the performance of wheeled systems due to the leg-wheel's superior ground contact.

2.1.1 Related Controllers

Researchers have historically explored a variety of models of locomotion, predominantly drawing inspiration from the biological realm. A study of the motion gait pattern of Pika (Hackert et al. 2006) illuminates the manner in which gait patterns are employed in locomotion. A discernible transition in locomotion patterns is evident as speed undergoes change in animals. The gait pattern and the angle of attack (Blickhan et al. 2006; Hackert et al. 2006) transformation bear a resemblance to the synchronization between the wheels in Asguard. The angle of attack is defined as the angle at which the leg makes contact with the ground. This parameter becomes particularly salient at elevated speeds, where the robot's rebound capacity is optimized at a specific angle of attack. It is conceivable that the synchronization can be utilized to regulate both the bounding behavior and the angle of attack. However, inadequate synchronization can lead to an augmented angle of attack, consequently diminishing the robot's forward momentum. The present study is confined to lower velocity ranges, specifically between $0.2\,\mathrm{m/s}$ to $0.5\,\mathrm{m/s}$. At these speeds, the robot's bounding and the effect of the angle of attack are minimal.

A considerable body of research has previously been dedicated to the study of robot locomotion for passive-hybrid systems. A preliminary controller for the six-legged RHex robot has been designed in Saranli et al. 2000. An investigation into the automated gait adaptation of RHex can be found in Weingarten et al. 2004. The bounding locomotion of Scout II is depicted in Poulakakis et al. 2006. However, it should be noted that the applicability of these locomotion and controller designs to Asguard is precluded by design differences. It is noteworthy that all of these publications employ the dimensionless term specific resistance to assess the locomotion performance. The Gabrielli–von Karman diagram,

as referenced in Gregorio et al. 1997, provides a comprehensive comparison of specific resistance among a diverse array of land vehicles.

Asguard has served as a foundational subject in numerous publications and theses. The motivation and design of the robot are discussed in detail in Eich et al. 2008a and Eich et al. 2008b, respectively. The work of Eich et al. 2009 proposes a compliance control architecture based on proprioceptive data, endowing the robot with the capacity to adapt automatically to different slopes. The bounding behavior of the robot is studied in detail in Machowinski 2009. The legged-wheel design is analyzed and optimized in Högemann 2008. The usage of embodied data for localization and mapping is developed in Schwendner et al. 2014.

2.1.2 Motivation

The present chapter undertakes a study of the robot's locomotion with the objective of developing effective control strategies. The wheel design necessitates adapted control strategies, as a simple velocity control designed for a wheel system is effective but is not optimal. The challenge lies in achieving optimal performance by comprehending the robot's inherent physical capabilities. However, the analytical capabilities are constrained by the inherent complexities and difficult to model non-linearities of the system.

The hybrid wheel-leg design exhibits the disadvantages inherent in both designs as well. The non-smooth profile of the wheeled leg generates vibrations that are difficult to control or steer, resulting in suboptimal performance. Additionally, the absence of damping in the leg design leads to significant vibrations in the robot body, which may potentially result in damage to the hardware.

To address the turning maneuvers, the robot utilizes differential steering. The rotation of the wheels on opposing sides at different velocities results in the robot skidding laterally and turning. To execute this skid, the robot must generate a force that exceeds the frictional resistance offered by the ground. This can prove challenging on rough surfaces and, in some cases, nearly impossible on uneven surfaces. While the implementation of differential steering is straightforward, its efficacy is compromised due to its lack of smoothness.

The design of the wheels as a passive-hybrid necessitates the development of specialized controllers to generate locomotion patterns that can be utilized to take advantage of the unique capabilities of the robot. The primary objective of this chapter is to explore the optimization of locomotion for the Asguard robot. This objective entails the development of joint-level controllers that can orchestrate the movement of the wheels in a coordinated manner, thereby enabling the robot to synchronize its motion and achieve optimal performance during longitudinal movement. The cascaded joint-level controller utilizes the deflection of a flexible coupling, along with expected loads, to enhance controller performance. Additionally, the chapter delves into the optimization of turning motion by maximizing the torques available for turning. The thesis objectives that are addressed in this chapter are: O-1a, O-1b and O-1c.

2.2 System Description - Asguard

The robot body is measured at a length of 50 cm, with a leg length of 19 cm. Its maximum velocity is measured at 2.0 m/s. The robot's design prioritizes a weight distribution that is more pronounced at

the front axle (60 %) compared to the rear axle (40 %). The complete specifications of the robot are provided in Table 2.1.

Table 2.1: Specifications of the Asguard Robot.

Length	95 cm
Width	51 cm
Height	44 cm
Leg length	19 cm
Mass	12.9 kg
Motors	4×24 V DC motor
Gears	46:1 planetary gears
Body joint limit	−40° to 40°
Maximum speed	2 m/s

The system is composed of three primary components: the legged-wheels, motor drives and robot body. Two significant sources of flexibility within the system are the legged-wheel and the flexible motor coupling. As a result of this flexibility, the legs exhibit movement in both linear and radial directions, which can lead to a bumpy behavior at higher speeds in certain scenarios. In Högemann 2008, the linear and radial spring constants and damping parameters are optimized to mitigate these effects. The flexible coupling functions as a shock absorber, thereby reducing the jerks affecting the robot body.

The robot has been equipped with a passive joint, which connects the front and rear segments of the body. This additional DoF is essential in enabling the robot to maintain contact with the ground with all four wheels on uneven surfaces, thereby enhancing overall traction. The robot's leg tips are integrated with a soft foot, a design element intended to augment longitudinal friction and mitigate lateral friction, thereby enhancing the robot's skid-steering capabilities.

The robot has been equipped with a PC104 stack and an array of sensors, including Inertial Measurement Unit (IMU), Light Detection and Ranging (LIDAR), RGB-cameras and Time of Flight (ToF) cameras. Furthermore, a differential Global Positioning System (GPS) system provides the ground truth from which it is possible to validate localization algorithms. The robot's operational mode encompasses both autonomous function and manual control, which can be facilitated through the use of a custom-designed control pad or a joystick.

The wheel motors are brushed 24 V DC motors with a planetary gear system which has an input to output ratio of 46:1. To drive the motors, custom-designed H-Bridge motor boards are utilized. The drives are connected with incremental encoders for feedback control, which provide a tick resolution of approximately 70 µrad on the wheel side. The motor is connected to the wheel through a planetary gear system and a flexible coupling (Figure 2.2), which obscures the estimation of the actual wheel position. The coupling also introduces additional motion effects between the motor and the wheel.

The control loop has been implemented on the embedded system, utilizing the Orocos real-time framework [1] on top of a Linux operating system with RT-PREEMPT patches. The communication infrastructure between the PC104 and the H-bridge hardware, which serves to regulate the motors, employs a Controller Area Network (CAN) bus. Latency measurement results demonstrate that the control

[1] https://orocos.org/

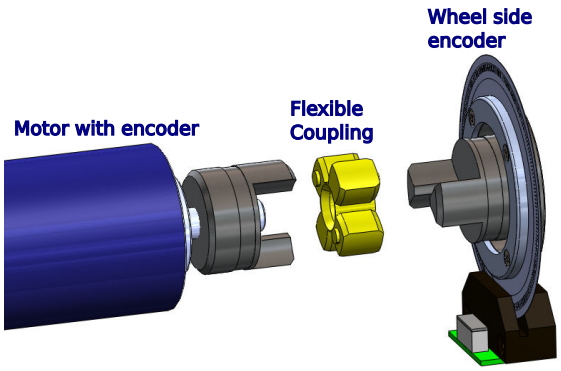


Figure 2.2: Design of the Asguard wheel drive with the motor on the left side, the flexible coupling in the middle and the encoder on the right side.

loop latency (between the sensors and the actuator) is less than 3 ms for a control frequency of 1 kHz. The software also includes a complete real-time data logging component, which enables the capture of the entire evolution of the control loop at its operating frequency.

2.3 Joint Controller Design

A typical wheeled robot utilizes a simple velocity controller at the joint level, a configuration that suffices for normal wheels. In contrast, the Asguard robot, with its passive hybrid design, necessitates a more sophisticated controller to ensure the synchronization between the spikes of different wheels. The controller must be capable of maintaining synchronization between the wheels while also preserving the offset between them to facilitate the execution of a desired locomotion pattern. Additionally, the controller must also be capable of handling external load disturbances and system non-linearities.

The control process is further complicated by the presence of a flexible coupling between the motor and the wheel. While the coupling absorbs shocks from the wheel and reduces the effects of jerks on the robot body, it introduces additional motion effects due to its flexibility. The controller must account for these effects to accurately follow wheel position references.

The drive with a flexible coupling can be modeled as a two-mass-spring-damper system, with high disturbance loads acting on the non-actuation side. The control problem of a flexible drive can be solved using a Generalized Proportional Integral (GPI)-based controller, as detailed in Becedas et al. 2009 and Singer and Seering 1989, or through pre-filtering of input, as outlined in Singer and Seering 1989 and Babu 2014. These methodologies are constrained by the following distinctive characteristics of this application: substantial deflection, considerable load disturbances, and significant non-linearities.

The most straightforward controller to construct is a rudimentary velocity controller on the motor side that utilizes solely the motor encoder. However, this controller is deficient in its inability to sustain wheel synchronization due to the substantial loads and disturbances that act upon the system.

This work proposes a solution that involves the implementation of cascaded controllers to enhance the performance of the position controller. The subsequent sections will delve into the design of the controller, the torque estimation, and the feed-forward torque computation, with the design of the robot being taken into consideration.

2.3.1 Cascaded Controllers

In order to enhance the position control performance of the controllers, a comparison was conducted between two distinct cascaded controllers. These controllers possess multiple feedback loops operating in parallel and regulating distinct process variables. They are configured in a cascaded manner, comprising inner and outer loops. The implementation of a cascaded design for the controller offers numerous benefits, including (i) non-linearities in the inner loop does not affect the outer loop, (ii) good disturbance rejection capabilities, (iii) ease of tuning the parameters, and (iv) the possibility to limit each process variable separately. For Asguard, two different cascaded controllers designs: Position-Velocity (PV) and Position-Velocity-Torque (PVT), are developed.

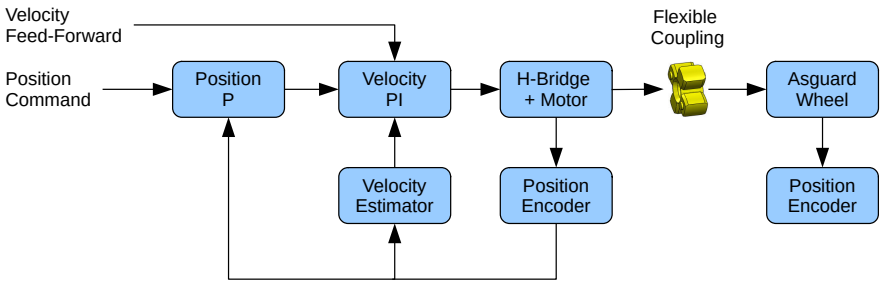


Figure 2.3: Design of the PV cascaded controller showing the two cascaded loops taking the feedback from the encoder on the motor side.

The PV controller is equipped with two loops that regulate the position and velocity of the motor, as illustrated in Figure 2.3. The reference position values are transmitted as command signals to the position loop, which is a Proportional (P)-controller. The position loop utilizes feedback from the position encoder located on the motor side. The inner velocity loop, which is a Proportional Integral (PI)-controller, utilizes the output of the position controller, along with the velocity feed-forward, as the reference. The feedback comprises the velocity estimated from the position encoder values. The output of the velocity controller is the Pulse Width Modulation (PWM) command for the H-Bridge drive. It should be noted that the flexible couple and the wheel are not included in the controller and do not directly influence the control.

The inclusion of the flexible coupling within the control loop is a potential avenue for enhancing the performance of the controller. A proposed solution involves the incorporation of a flexible coupling deflection control mechanism, which, in turn, facilitates the regulation of the torque output acting on the wheels. The design of the controller with an additional inner torque loop, which incorporates direct position and velocity feedback from the wheel side encoder, is referred to as the PVT controller.

The PVT controller, depicted in Figure 2.4, is composed of three cascaded loops: position, velocity and torque. The position loop is a P controller, the velocity loop is a PI controller, and the torque loop is a Proportional Integral Derivative (PID) controller.

The innermost loop is the torque controller, which receives input from the velocity controller and the feed-forward torque command. The feedback for the torque loop is obtained from the deflection of the flexible coupling, which is then converted into the estimated torque using the torque-estimator module. The encoder measurements from both the wheel and the motor side are used to compute the deflection of the coupling. The output of the torque controller is the PWM signal, which is then fed to the motor drive.

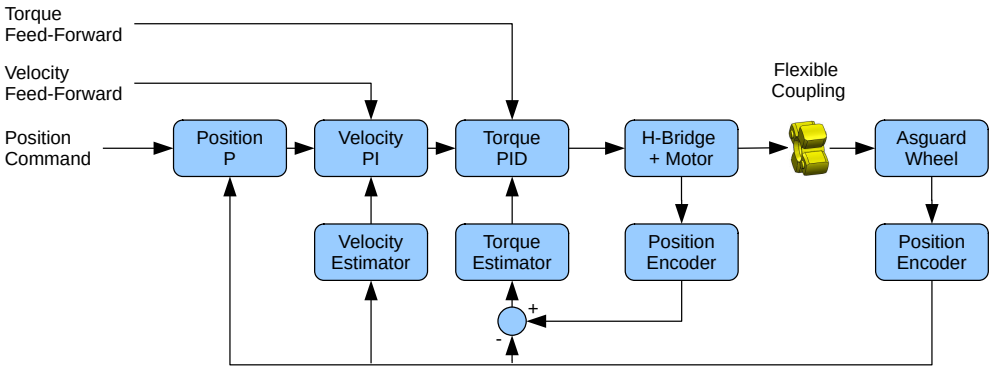


Figure 2.4: Design of the cascaded PVT controller with the three cascaded loops taking feedback from both the motor and wheel side encoders.

The middle loop is the velocity controller, which gets its input from the position controller and the feed-forward velocity command. The feedback for the velocity loop is obtained from the wheel-side encoder. The output of the velocity controller is the torque command, which is then fed to the torque controller. The outermost loop is the position controller, which receives input from the trajectory generator, and the feedback for the position loop is obtained from the wheel-side encoder. The output of the position controller is the velocity command, which is then fed to the velocity controller loop.

Table 2.2: Tuned parameters for PV and PVT controllers.

	PV	**PVT**
Position proportional	3.80	3.80
Velocity proportional	0.07	0.1
Velocity integral	0.65	0.0
Integral windup	0.06	–
Velocity smoothing	0.6	0.6
Velocity feed-forward	1.0	1.0
Acceleration feed-forward	0.0	0.0
Torque proportional	–	0.11
Torque integral	–	50.0
Torque derivative filtering	–	10.0
Torque feed-forward	–	1.0

The cascaded control loop is tuned in steps, with each loop being tuned sequentially. Initially, the innermost torque loop is tuned while the outer loops are disabled. Subsequently, the velocity loop is tuned while the position controller is disabled. Finally, the position loop is tuned. The tuned PV and PVT gains are given in Table 2.2.

As demonstrated in Figure 2.5, the PVT controller exhibits enhanced positioning accuracy in comparison to the PV controller. This enhancement is evident in the plots that illustrate the position reference and the actual measured position for both controllers. The position error is also plotted, providing a quantitative assessment of the deviation between the reference and the actual values. It should be

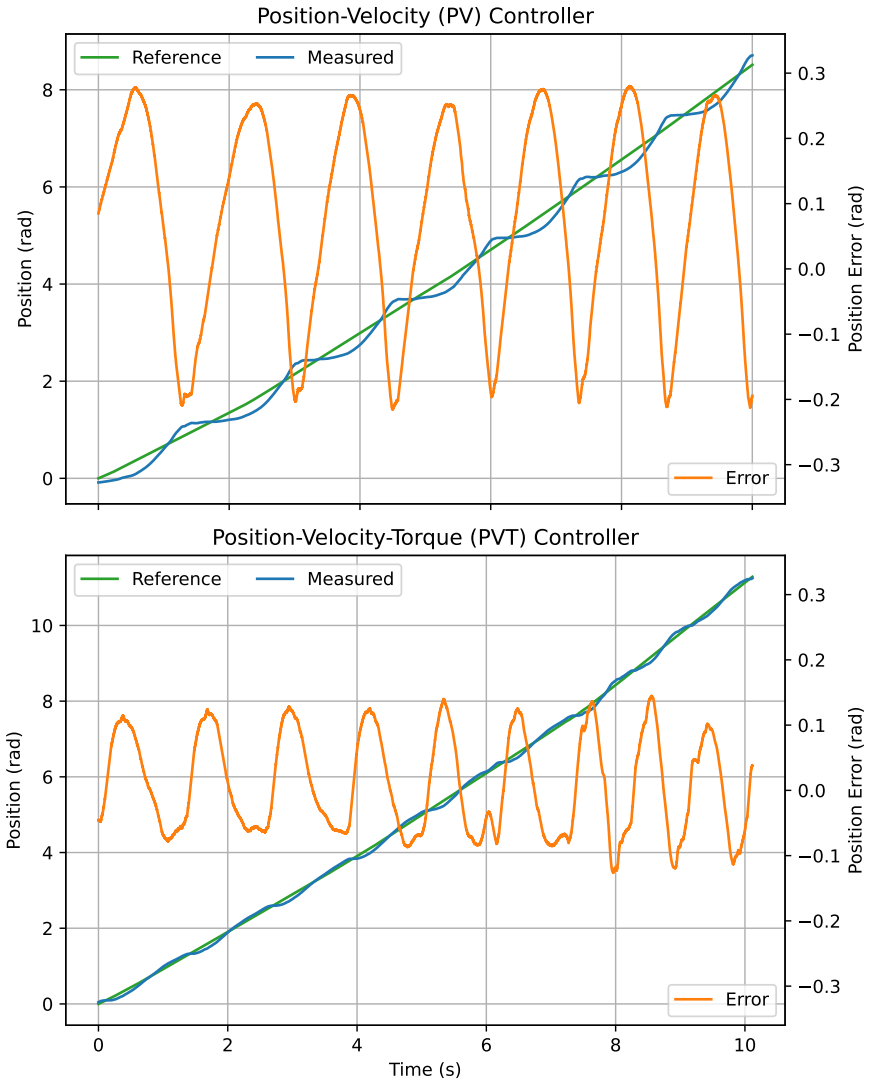


Figure 2.5: Comparison of position errors between PV controller, using only the motor side feedback and PVT controller, which has an additional wheel side feedback.

noted that in this particular instance, neither the velocity nor the torque feed-forwards were utilized by the controllers. The analysis reveals a substantial enhancement in positioning accuracy for the PVT controller in comparison to the PV controller. When similar reference position trajectory commands are applied, the Root-mean-squared error (RMSE) error of the former is 0.076 rad, while the latter is 0.172 rad. This represents a reduction of 56 % for the PVT controller in comparison with the PV controller.

The PVT-controller's design incorporates a module that estimates torques by leveraging the deflection of the flexible coupling. The subsequent section delves into the details of this torque estimation module, including the model and the parameter estimation aspects.

2.3.2 Torque Estimator using Deflection of the Flexible Coupling

The flexible coupling utilized in the motor drive constitutes a torsional spring-damper system, which possesses the capacity to absorb a certain degree of shocks generated by ground contacts. However, this coupling simultaneously introduces undesired motion effects between the motor and the wheel, thereby complicating the modeling of its dynamics. Nonetheless, if adequately modeled, the deflection of the coupling can be utilized to estimate the torque being applied to the wheel, thus enabling potential torque control.

The relationship between deflection and torque is characterized by significant hysteresis, necessitating a model to accurately describe it. The Bouc-Wen model, initially developed by Bouc 1967 and subsequently refined by Wen 1976, is a widely used model for hysteresis. It employs a differential equation formulation that can represent smooth hysteresis with a limited number of parameters. Notably, the model is both rate-independent and computationally efficient.

The model's formulation is presented below, followed by the extensions necessary to account for unmodelled dynamics in Asguard. This section also describes the method for calibrating the parameters.

The Bouc-Wen class of models (Oh et al. 2023) consists of two equations. The first equation describes the torques which is given by

$$T(t) = ak_i u(t) + (1-a)k_i z(t), \tag{2.1}$$

where

$$a := \frac{k_f}{k_i} \tag{2.2}$$

is the ratio between post-yield, k_f, and pre-yield (elastic) stiffness, k_i. k_i is given by

$$k_i := \frac{F(t)}{u(t)}, \tag{2.3}$$

which is the ratio between the yield force $F(t)$ and the yield displacement $u(t)$. The equation consists of two terms where the first term is the linear strain component and the second term is the hysteretic strain component.

The second equation describes the hysteretic displacement $z(t)$ which is a differential equation. It is a function of the deflection $u(t)$ and the deflection speed $\dot{u}(t)$ given by

$$\dot{z}(t) = A\dot{u}(t) - \beta|\dot{u}(t)||z|^{n-1}z + \gamma\dot{u}(t)|z(t)|^n, \tag{2.4}$$

where $A, \beta > 0, \gamma$, and n are dimensionless quantities, $z(t)$ is the dimensionless hysteretic state variable, and n is the smoothness exponent ($n \geq 1$). A, β, γ are the hysteresis shape parameters where A controls the amplitude while β and γ can be used to control the sharpness of the hysteresis loop.

The Bouc-Wen model has been demonstrated to be capable of capturing the hysteresis behavior of smooth hysteresis loops. However, in the case of flexible coupling, there are additional factors that must be integrated into the model. These additional factors include damping and gear play. Gear play is defined as the region around the zero deflection where the coupling has very low stiffness. This region is therefore approximated as a gear play region. The deflection offset is defined as the offset in

the deflection due to the gear play. The model that is extended with damping and deflection offset is given by

$$\bar{T}(t) = ak_i(u(t) - k_o) + (1 - a)k_i z(t) - k_d \dot{u}(t), \tag{2.5}$$

where k_o is the deflection offset and k_d is the damping constant.

The final extended torque model with gear play is given by

$$T_{ext}(t) = \begin{cases} 0, & |\bar{T}(t)| \leq T_{\text{play}}(t) \\ \bar{T}(t) - \text{sgn}(\bar{T}(t)) \cdot T_{\text{play}}(t), & \text{otherwise} \end{cases}, \tag{2.6}$$

where the gear play torque is given by $T_{\text{play}}(t) = ak_i \cdot \delta_b/2.0$ and δ_b is the gear play or backlash. The gear play torque is half of the torque required to move the coupling from one end of the gear play to the other end.

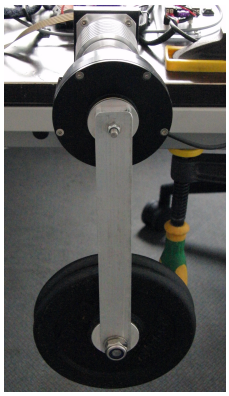


Figure 2.6: Calibration setup with the swinging mass for calibrating the model and estimating the hysteresis parameters.

A specialized calibration setup was engineered to calibrate the model parameters, as illustrated in Figure 2.6. The calibration setup consists of dumbbell masses affixed to a shaft, with the shaft's opposing end connected to the Asguard drive instead of the wheel. The swinging motion of the mass mimics a pendulum's oscillation, thereby generating a load torque dependent on the angle between the shaft and the vertical.

The equation of motion of the swinging mass is given by

$$I\ddot{\theta}_{wheel} + c\dot{\theta}_{wheel} + T_{ext}(t) = \tau(t), \tag{2.7}$$

where θ_{wheel} is the angular position of the wheel, I is the Moment of Inertia of the rotating mass, c is the viscous damping coefficient & $\tau(t)$ is the applied external force

The resultant torques applied by the coupling are given by

$$T_{ext}(t) = rM_m g \sin(\theta_{wheel} - \theta_{vert}) + M_m r^2 \ddot{\theta}_{wheel} + T_{friction}, \tag{2.8}$$

where M_m is the mass being rotated, r the length of the connecting rod, and θ_{vert} is the angle of the connecting rod to the vertical

The data from the calibration run are fitted with the model using the Levenberg-Marquardt optimization method, as described in Levenberg 1944 and Marquardt 1963. The model parameters are provided with limits. One of the issues encountered was the parameter interdependency between the parameters β and γ defined by

$$\|\gamma\| \leq \beta. \tag{2.9}$$

The capacity to establish limits in a dynamic manner is not available for this particular optimization algorithm. This issue is addressed by introducing a forced error if the specific condition is violated by the optimizer.

Acceleration estimation from the encoder position measurement by double differentiation has been shown to amplify quantization noise. Consequently, the study was not extended to higher dynamics, and the adequacy of the model for such cases remains uncertain. The coupling exhibits a distinct behavior around the zero deflection region compared to higher deflection. This region, characterized by its low stiffness, is approximated as a gear play region.

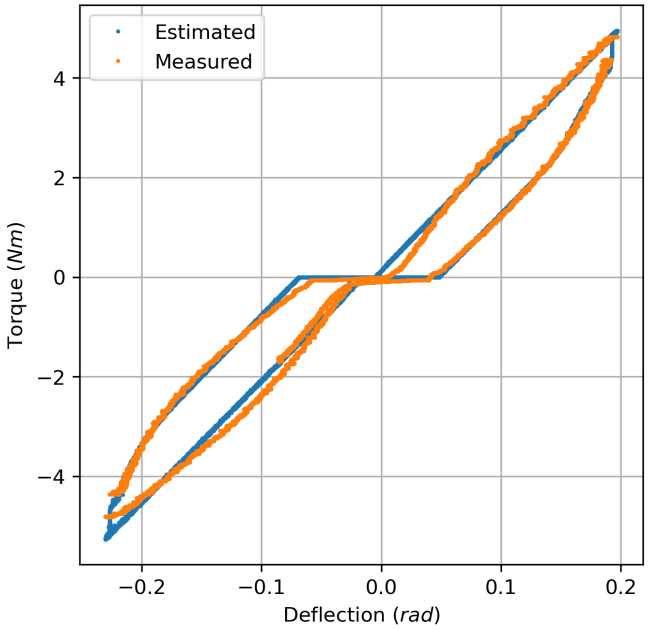


Figure 2.7: Deflection versus torque during the calibration process and the fitted model for the rear-right wheel.

The estimated model is compared with the calibration data using a deflection versus torque plot as shown in Figure 2.7, and the estimated values for the parameters for the four wheels are given in Table 2.3. The comparison shows a RMSE of 0.23 N m, which corresponds to 4.7 % at a maximum torque of about 4.9 N m.

The performance of the torque controller during the execution of the PVT controller, utilizing the parameters furnished in Table 2.2, is illustrated in Figure 2.8. The figure demonstrates the reference torque generated by the outer velocity loop, the estimated current torque and the error. During this forward motion, the reference torques range from −4.84 to 10.11 N m, and the torque error ranges from −1.67 to 1.39 N m newton meters. The RMSE is 0.63 N m, which is 6.28 % of the maximum reference torque, 10.11 N m. It is evident that the highest errors coincide with the reference value peaks.

Table 2.3: Estimated parameters of the extended Bouc-Wen hysteresis model for the four Asguard wheels.

	Front-Left	**Front-Right**	**Rear-Left**	**Rear-Right**
A	2.087 07	2.360 89	2.453 63	2.610 14
β	1.808 41	1.647 66	0.980 759	1.2354
γ	1.808 26	1.555 49	0.980 757	1.235 14
n	1.513 82	1.0	1.1474	1.0334
a	0.408 525	0.413 352	0.510 087	0.462 277
k_i	1.567 78	1.578 97	1.130 33	1.351 89
δ_b	0.651 14	0.640 117	0.255 157	0.398 766
k_o	0.571 874	0.0	1.032 89	0.334 653
k_d	−0.025 199	−0.017 962	−0.030 738	−0.027 954

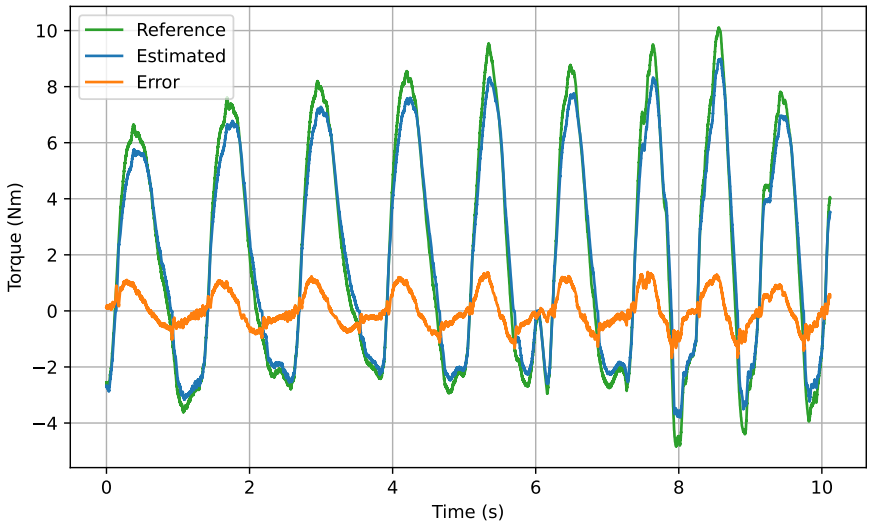


Figure 2.8: Torque controller performance during execution of the PVT controller.

The input for the torque controller is generated by the velocity controller with an additional feed-forward command, the physical properties of the robot being used to approximate the latter. This subject is addressed in the following section.

2.3.3 Estimation of Feed-Foward Torques

In the context where the anticipated load exerted on each wheel is supplied as a feed-forward signal to the torque control loop, the efficacy of the controller can be enhanced. To ascertain the load acting on each wheel, it is essential to undertake an analysis of the quasi-static forces acting on the robot. The normal force distribution across the wheels of the robot is predicated on static reaction forces stemming from the robot's weight. This computation is executed at each sample interval, taking into account the measured wheel positions and the estimated leg positions. This estimation is termed quasi-static force estimation, as it disregards the velocities and accelerations of the system.

The assumption underpinning this method is that the leg which is the lowest, in relation to the robot body plane, is in contact with the ground and coincident with the contact point frame C_{il}. This assumption is mostly valid for relatively even surfaces. However, for highly uneven surfaces, it is possible that the higher feet is in contact with the ground and the lowest is not. In such scenarios, to ensure a more accurate estimation, the addition of sensors to the feet to detect contact and forces is necessary, as previously outlined by Fondahl et al. 2012.

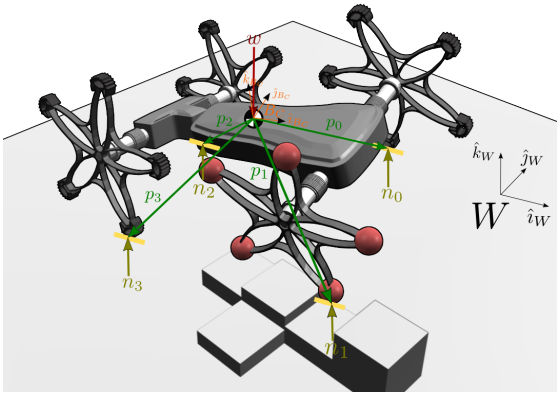


Figure 2.9: Free body diagram for quasi-static force computations of Asguard (Image source: Hidalgo-Carrio et al. 2014).

The free body diagram for computation of static forces is given in Figure 2.9. W is the world reference frame, B_C the body fixed reference frame with origin at the CoM and $\boldsymbol{w}$ is the weight of the robot acting along the gravity vector ($\hat{k}_W$ axis). Let $i = \{0, 1, 2, 3\}$, then n_i are the normal reaction forces from the ground due to the robot weight and $\boldsymbol{p}_i$ are the position vectors to the leg contact points. Henceforth, the corresponding coordinate systems are added to the representations.

A new reference frame B'_C (not shown in the Figure) can be defined with the origin coinciding with the CoM of the robot, but aligned to W. The terms for $\boldsymbol{n}_{B'i}$ and $\boldsymbol{w}$ in B'_C is given by,

$$\boldsymbol{n}_{B'_C i} = \begin{bmatrix} 0 \\ 0 \\ n_i \end{bmatrix} \text{ and } \boldsymbol{w} = \begin{bmatrix} 0 \\ 0 \\ M_r g \end{bmatrix}, \tag{2.10}$$

where n_i are the scalar reaction forces along the $\hat{k}_{B'}$, M_R is the mass of the robot and g is the acceleration due to gravity.

The objective is to derive the equations for the values of n_i. The equations are developed based on the fact that the robot has a free joint and this joint cannot transmit any torques. Therefore, the torques in the front and the rear part of the robot along this free joint are independent. With a quasi-static assumption, the following equations become valid:

1. Sum of forces along $\hat{k}_{B'_C}$ equals the weight of the robot such that

$$\Sigma n_i = M_r g. \tag{2.11}$$

2. Sum of torques along $\hat{\jmath}_{B_C}$ is zero such that

$$\Sigma(\boldsymbol{p}_{B_C i} \times \boldsymbol{n}_{B_C i})|_{y_{B_C}} = 0. \tag{2.12}$$

3. Sum of torques due to $\boldsymbol{n}_0$ and $\boldsymbol{n}_1$ along $\hat{\imath}_{B_C}$ is zero such that

$$(\boldsymbol{p}_{B_C 0} \times \boldsymbol{n}_{B_C 0} + \boldsymbol{p}_{B_C 1} \times \boldsymbol{n}_{B_C 1})|_{x_{B_C}} = 0. \tag{2.13}$$

4. Sum of torques due to $\boldsymbol{n}_2$ and $\boldsymbol{n}_3$ along $\hat{\imath}_{B_C}$ is zero such that

$$(\boldsymbol{p}_{B2} \times \boldsymbol{n}_{B2} + \boldsymbol{p}_{B3} \times \boldsymbol{n}_{B3})|_{x_{B_C}} = 0. \tag{2.14}$$

Let the rotation from B'_C to B_C and the position vector $\boldsymbol{p}_{B_C i}$ be given by

$$R_{B_C B'_C} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \text{ and } \boldsymbol{p}_{B_C i} = \begin{bmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \end{bmatrix}. \tag{2.15}$$

Using Equation (2.15), relationship between $\boldsymbol{n}_{B'_C i}$ and $\boldsymbol{n}_{B_C i}$ is given by,

$$\boldsymbol{n}_{B_C i} = R_{BB'} \boldsymbol{n}_{B'_C i} = \begin{bmatrix} r_{02} \\ r_{12} \\ r_{22} \end{bmatrix} n_i. \tag{2.16}$$

Computing torques from Equations (2.15) and (2.16) gives

$$\boldsymbol{\tau}_{B_C i} = \boldsymbol{p}_{B_C i} \times \boldsymbol{n}_{B_C i} = \begin{bmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \end{bmatrix} \times \begin{bmatrix} r_{02} \\ r_{12} \\ r_{22} \end{bmatrix} n_i, \tag{2.17}$$

$$\boldsymbol{\tau}_{B_C i} = \begin{bmatrix} p_{iy} r_{iz} - p_{iz} r_{iy} \\ p_{iz} r_{ix} - p_{ix} r_{iz} \\ p_{ix} r_{iy} - p_{iy} r_{ix} \end{bmatrix} n_i. \tag{2.18}$$

Using Equation (2.18) let,

$$\begin{bmatrix} p_{iy} r_{iz} - p_{iz} r_{iy} \\ p_{iz} r_{ix} - p_{ix} r_{iz} \\ p_{ix} r_{iy} - p_{iy} r_{ix} \end{bmatrix} = \begin{bmatrix} t_{ix} \\ t_{iy} \\ t_{iz} \end{bmatrix}. \tag{2.19}$$

Substituting Equations (2.17) to (2.19) in Equations (2.12) to (2.14), and combining them with Equation (2.11) gives,

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ t_{0y} & t_{1y} & t_{2y} & t_{3y} \\ t_{0x} & t_{1x} & 0 & 0 \\ 0 & 0 & t_{2x} & t_{3x} \end{bmatrix} \begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3 \end{bmatrix} = \begin{bmatrix} M_r g \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.20}$$

The set of linear equations Equation (2.20) can be solved for n_i by,

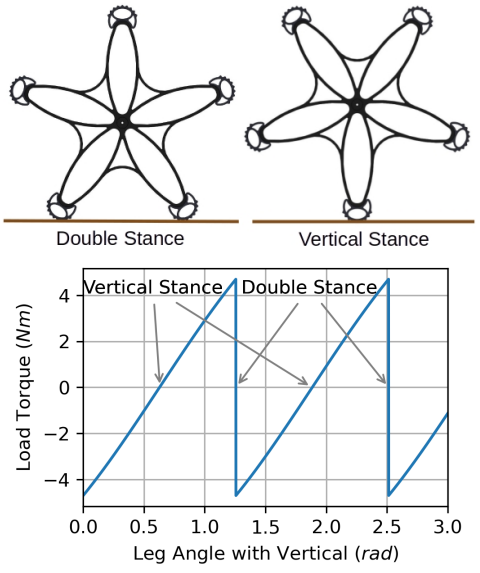
$$\begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ t_{0y} & t_{1y} & t_{2y} & t_{3y} \\ t_{0x} & t_{1x} & 0 & 0 \\ 0 & 0 & t_{2x} & t_{3x} \end{bmatrix}^{-1} \begin{bmatrix} M_r g \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.21}$$

The reaction forces computed using Equation (2.21) at every time step goes as input to the weighted Jacobian odometry model developed in Hidalgo-Carrio et al. 2014. These values, along with the vector to the wheel axis, are used to compute the torsional load acting on each wheel. The load acting on each wheel is then used as feed-forward to the torque controller.

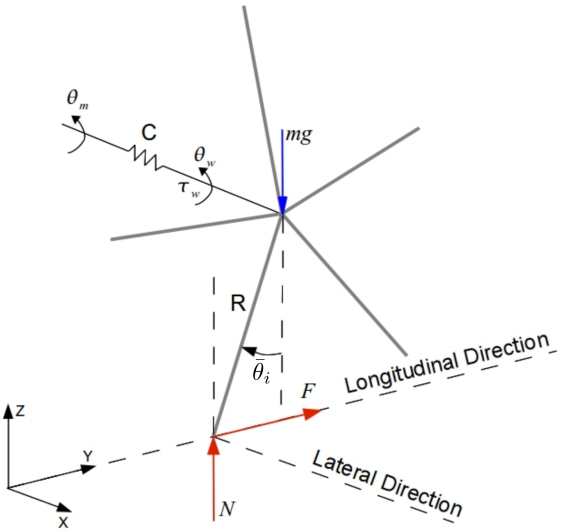
This section has developed the joint controllers required to control the position of the wheels. These controllers can then be used to optimize the robot's longitudinal and turn motions, as described in the following sections.

2.4 Longitudinal Movement Optimization

It can be observed that longitudinal movement for Asguard is challenging and inefficient on hard ground surfaces. The robot tends to lose ground contact, which hinders its ability to generate sufficient traction to propel forward. The robot's behavior can be influenced to some extent by modifying its locomotion patterns, defined as the relative motion of the wheeled legs.



(a) The DS and VS position of the wheels are depicted on the top. The plot of external torsional load acting on the wheel as it moves between the two stances are shown below.



(b) The free-body diagram of the wheel depicting the forces and torques acting on the wheel.

Figure 2.10: Asguard wheel stances, the external loads and the free-body diagram of forces acting on the wheeled-leg.

In legged robots, the leg motion alternates between swing and stance phases. A rudimentary velocity controller for Asguard occasionally resulted in these phases hindering each other (Eich et al. 2008b) and, on occasion, flipping the robot. Asguard can exhibit these two phases concurrently. Primarily, the legged-wheels display two distinct stances: Vertical Stance (VS) and Double Stance (DS), as illustrated in Figure 2.10a. The position corresponding VS is characterized by the contact of a single leg, oriented vertically relative to the body. Conversely, DS denotes the configuration where both legs are in contact with the ground simultaneously. The blocking of the swing phase occurs at the point designated by DS. The objective of longitudinal movement optimization seeks to mitigate the impact of this blocking by facilitating effective synchronizations between wheels.

The external static load acting on the wheel is determined by its position, as illustrated in Figure 2.10a. During the transition from the VS to DS position, the external load torque is positive, while in the transition from DS to VS, the load torque is negative. The load transfer from one leg to another at the DS is not abrupt or discontinuous in reality.

As illustrated in Figure 2.10b, the forces and torques acting on the wheel are determined by the external forces, which include the normal ground reaction force N and the traction force F. The angle between the leg and the gravity vector, denoted by the angle of inclination, denoted by the symbol $\bar{\theta}_i$, is a key factor in determining the normal ground reaction force, or N. The motor exerts a torque, denoted as τ_w, on the wheel via the flexible coupling, which is a function of the coupling's deflection $(\theta_w - \theta_m)$. This deflection is employed to estimate the torque acting on the wheel, as outlined in Section 2.3.2.

However, when the controller views the robot as a wheeled system, neglecting its legged properties, the result is a vibrating system with inefficient locomotion. Synchronizing the wheels to have a particular relative orientation at a specific moment in time allows the robot to emulate the gait patterns of legs. This influences the energy transfer between different parts of the robot, effectively smoothing the locomotion and improving its efficiency. The synchronization can be achieved by introducing offsets between the wheels.

2.4.1 Locomotion Efficiency Metric

Locomotion can be typically evaluated using specific resistance or Froude number. The term specific resistance for locomotion was introduced in Gabrielli and Karman 1950. It is defined as the energy consumption per unit distance per unit weight (Gregorio et al. 1997; Siciliano and Khatib 2008) and given by

$$\epsilon = \frac{E}{Mgd}, \tag{2.22}$$

where E is the energy consumed, M is the mass of the vehicle, g is the acceleration due to gravity and d is the distance traveled. It has since been used in numerous literatures to compare wide range of locomotion.

An alternative metric for measuring locomotion efficiency, the Froude number (Siciliano and Khatib 2008), is commonly usually used to characterize animal locomotion. It is calculated as

$$F_r = \frac{V^2}{gh}, \tag{2.23}$$

where V is the velocity of walking or running, g is the acceleration due to gravity and h is the height of the hip joint from ground.

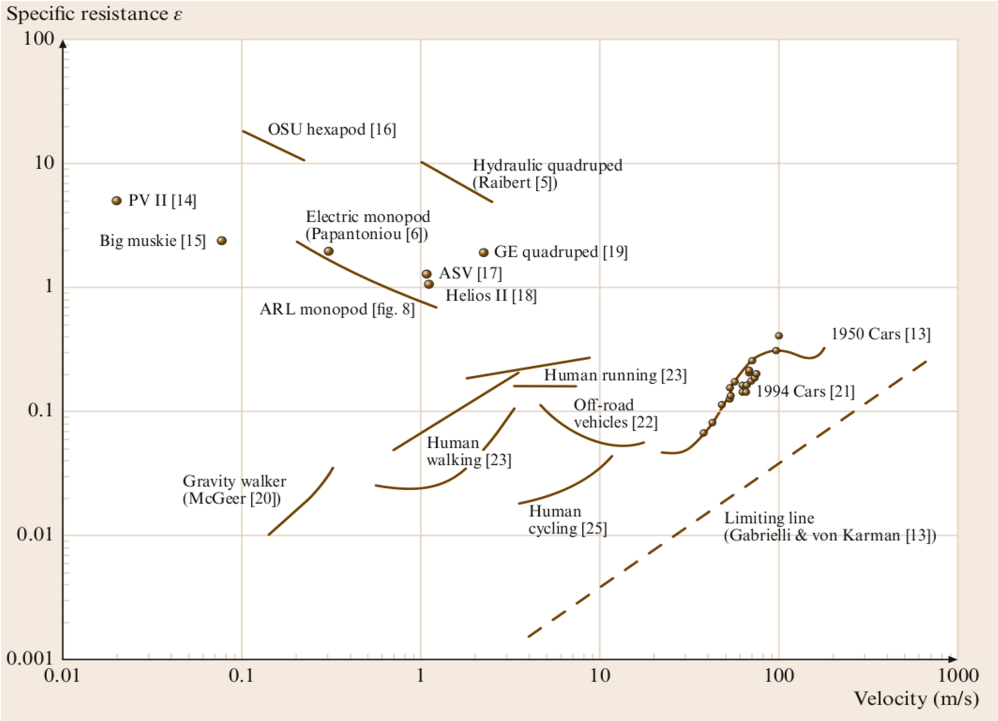


Figure 2.11: Comparison of specific resistance for different robots, animals and vehicles. (Image source: Siciliano and Khatib 2008)

In comparison to Froude number, Specific resistance is better suited for analyzing robot motions and hence will be used in this work. To put the specific resistance (ε) values in perspective, cars have a specific resistance in the range 0.07 to 0.4. Specific resistance for a human walking is between 0.02 to 0.1. RHex robot using automated tuning methods obtained a specific resistance of 0.6 (Weingarten et al. 2004). Quadruped Robot running with Bounding Gait (Talebi et al. 2000) had a specific resistance of 0.32. McGeer's gravity walker shows the lowest specific resistance among walking robot at 0.01. Among powered legged robots, ARl Monopod (Gregorio et al. 1997) with ε=0.7 shows the lowest specific resistance.

The energy consumed in our case is calculating the integral of power consumed by the motor. So the total energy is given by

$$E = V_{app} \int I(t) PWM(t) dt, \tag{2.24}$$

where V_{app} is the battery voltage, $I(t)$ is the current measured at the motor and $PWM(t)$ is the PWM input given to the motor. When the current is measured at the source, the equation becomes

$$E = V_{app} \int I(t) dt. \tag{2.25}$$

The equation for specific resistance (Equation (2.22)) can be used to estimate the error propagation due to measurement inaccuracies. The error propagation(*Appendix V: Uncertainties and Error Propagation* 2004) for multiplication is given by

$$(x \pm \delta_x)(y \pm \delta_y) = (xy) \pm (\sqrt{y^2\delta_x^2 + x^2\delta_y^2}, \tag{2.26}$$

and the error propagation for division is given by

$$\frac{(x \pm \delta_x)}{(y \pm \delta_y)} = \frac{x}{y} \pm \left(\sqrt{\left(\frac{x}{y^2}\right)^2 {\delta_y}^2 + \left(\frac{1}{y}\right)^2 {\delta_x}^2} \right). \tag{2.27}$$

Average value and expected errors for $energy = 300.0 \pm 30.0\,\mathrm{N\,m}$, $distance = 8.0 \pm 0.25\,\mathrm{m}$ and $mass = 13.0 \pm 0.1\,\mathrm{kg}$. Using Equations (2.26) and (2.27) and the average values given above, the error propagation was estimated to be approximately $10.0\,\%$.

2.4.2 Locomotion Patterns using Positional Offsets

A potential enhancement of locomotion in Asguard can be achieved by maintaining a steady flow of energy within the system. To elucidate the underlying principle, we propose two scenarios. In the initial scenario, the leg-wheels are assumed to operate in perfect synchronization, with no offset. At the VS, the robot's height results in the accumulation of potential energy. This potential energy is subsequently converted into kinetic energy as the legged wheel undergoes rotation. However, this kinetic energy is abruptly removed when the wheel reaches the DS, creating unnecessary vibrations. Additionally, to move from the DS to the VS, the motor should provide additional energy and lift the robot up.

In the second scenario, assume the Front-Left (FL) wheel is offset from the Rear-Left (RL) wheel by $\frac{\pi}{5}\mathrm{rad}$. Analogously, the Front-Right (FR) wheel exhibits an identical offset to the Rear-Right (RR) wheel. When the front wheels are in the VS position, the back wheels are in the DS position. As the wheels rotate, the potential energy from the rear part is transformed into kinetic energy, thereby assisting in overcoming the DS block and acquiring potential energy for the VS of the front part. The transfer of energy between these states is likely to impact locomotion efficiency, which is also subject to variation depending on system velocity and surface material properties. This principle can be applied to both left and right wheels.

In order to define the offsets, let the angular position of the wheel be $\theta_{w,i}$, where

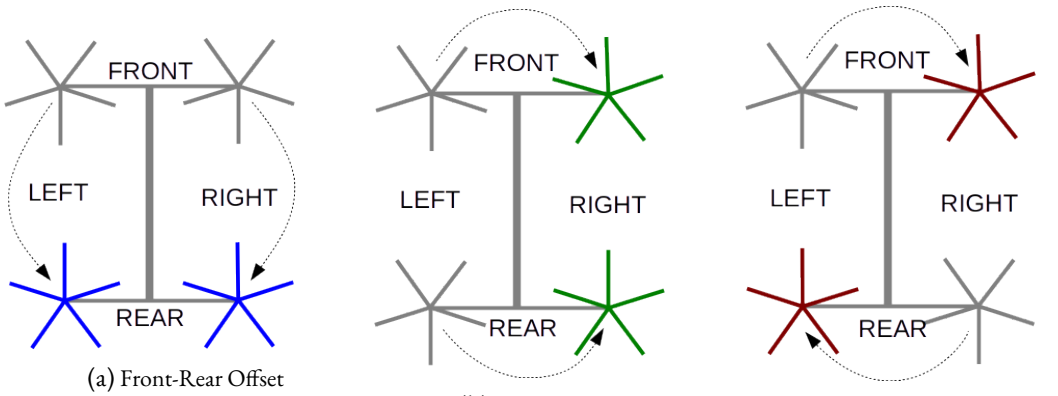
$$i \in \{FL, FR, RL, RR\}$$

are the four wheels of the robot. The leg position which is the angle made by the lowest leg with the vertical axis of each defined by

$$\bar{\theta}_i = \theta_{w,i} \bmod \frac{\pi}{5},$$

assuming that the zero position is synchronized for all motor and wheel encoders.

Three different offsets, as depicted in Figure 2.12, are defined to parametrize the locomotion patterns: Front-Rear Offset (FRO), Left-Right Offset (LRO) and Front-Rear Cross Offset (FRCO). The FRO ($\phi_{FR} \in [0.0, 1.0]$) is the difference in normalized offset between the front and the rear legs (Fig-



(a) Front-Rear Offset (b) Left-Right Offset (c) Front-Rear Cross Offset

Figure 2.12: Wheel offsets for generating locomotion patterns during longitudinal motion.

ure 2.12a). LRO ($\phi_{LR} \in [0.0, 1.0]$) is the normalized offset between left and right legs (Figure 2.12b). The FRCO ($\phi_{FRC} \in [0.0, 1.0]$) is the offset between wheels of the left and right side when the diagonal wheels are in synchronization (Figure 2.12c). The angular values for the offsets can range from 0 to $\frac{\pi}{5}$ rad, which covers half of the angle between two adjacent legs. Due to the symmetry of the wheel design, the negative values can be ignored. The offsets provide the following constraints for the trajectory generation:

$$\phi_{FR} = |(\bar{\theta}_{FL} + \bar{\theta}_{FR}) - (\bar{\theta}_{RL} + \bar{\theta}_{RR})| \frac{10}{\pi}, \tag{2.28}$$

$$\phi_{LR} = |(\bar{\theta}_{FL} + \bar{\theta}_{RL}) - (\bar{\theta}_{FR} + \bar{\theta}_{RR})| \frac{10}{\pi}, \tag{2.29}$$

and

$$\phi_{FRC} = |(\bar{\theta}_{FL} + \bar{\theta}_{RR}) - (\bar{\theta}_{FR} + \bar{\theta}_{RL})| \frac{10}{\pi}. \tag{2.30}$$

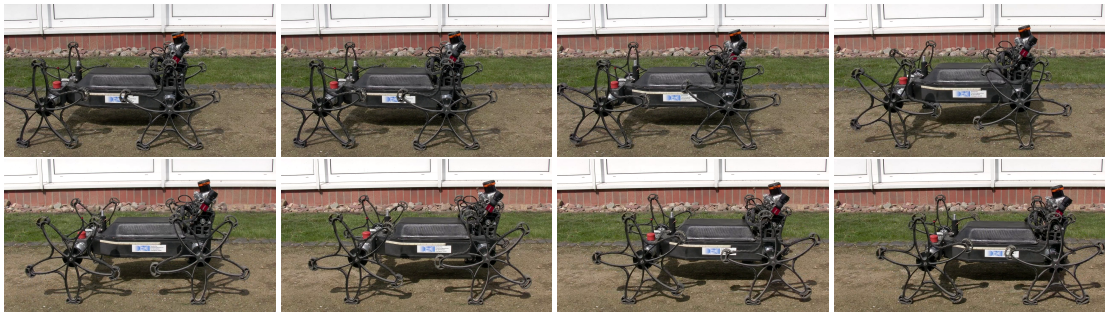
Figure 2.13 shows a sequence of images[2] with different wheel synchronizations.

2.4.3 Experimental Setup for Longitudinal Motion

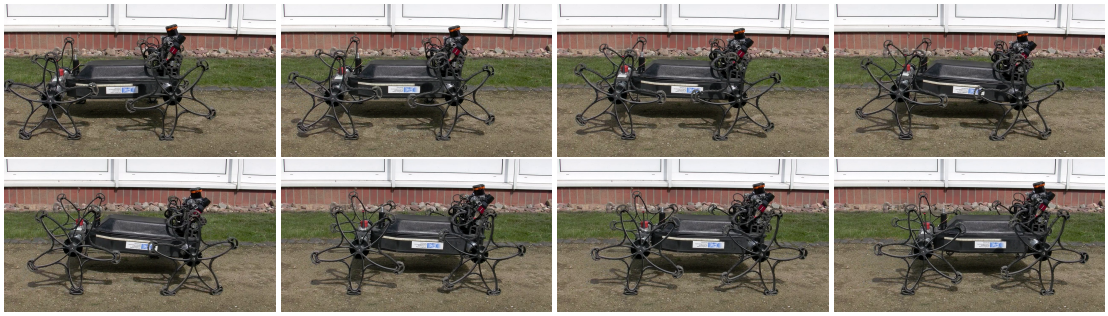
An experimental setup was developed to collect data to evaluate locomotion efficiency during straight-line motion. The robot was made to move in a straight line from an initial position, through a start position, and through a stop position. To measure the time difference between the start and finish, two time-synchronized cameras are utilized. The time difference is then used to extract the corresponding data from the log. The actual data extracted is the log from log start to log stop (Figure 2.14).

The tests were performed on both hard ground and sand. Hard ground has low damping compared to the legs, while sand has high damping effect compared to the legs. The robot forward velocities are assigned in the ranges low ($\sim$0.2 m/s), medium ($\sim$0.4 m/s) and high ($\sim$0.5 m/s). At velocities higher than 0.5 m/s, the robot tends to deviate from the designated track. Figure 2.14 illustrates the experimental setup.

[2] The corresponding video footage of the longitudinal motion of Asguard with different types of offsets can be found at https://drive.google.com/file/d/12rcrYX_YInsf_jSi4a41OskqsiJy2esH/view?usp=sharing.



(a) $\phi_{FB} = 0$



(b) $\phi_{FB} = 1.0$

Figure 2.13: Sequences of images (row-wise from left to right) showing sample wheel synchronizations during the experiments.

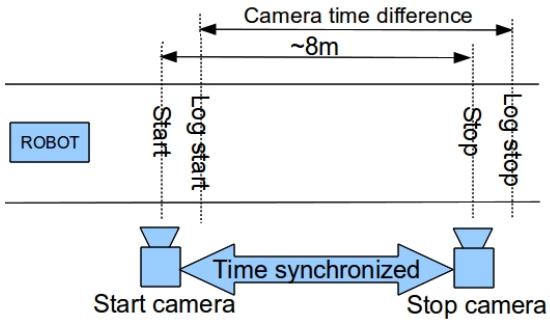


Figure 2.14: Experimental setup for evaluation of the effects of wheel offsets on the efficiency of longitudinal motion.

The offsets ϕ_{FR} and ϕ_{LRC} were assigned the values 0, 0.5 and 1.0 and for the offset ϕ_{LR}, the values 0, 0.25, 0.5, 0.75 and 1.0 are used.

The robot systematically logs the battery voltage, PWM input and current at a frequency of 1.0 kHz. The data logged during the time difference measured by the camera is then used to calculate the specific resistance. The length of the track which is measured for each test setup, is approximately 8.0 m from start to finish.

The vertical movement of the robot's body was also measured and analyzed using a motion capture system to understand the motions that affect the system. Vibrations on the robot body can affect the proper functioning of the robot sensors and can damage the components in the long run. Unnecessary vibrations reduce the efficiency of locomotion and are therefore an interesting parameter to investigate,

as they give indications as to why some configurations are better than others. The amplitude and frequency of the vibrations should be as low as possible.

In this work, the vibration was analyzed only for limited configurations, and only important results are presented. This limitation arises from the constrained field of view of the motion capture system. To expand the understanding of the subject, separate experiments were conducted using a 3D motion capture system to capture the vertical vibrations of the robot. The motion capture system from Qualysis captures the X-Y-Z motion of the robot at 100 Hz using a single marker. The three Infra-Red (IR) cameras are positioned in a manner that intersects along the track.

2.4.4 Vertical Movements

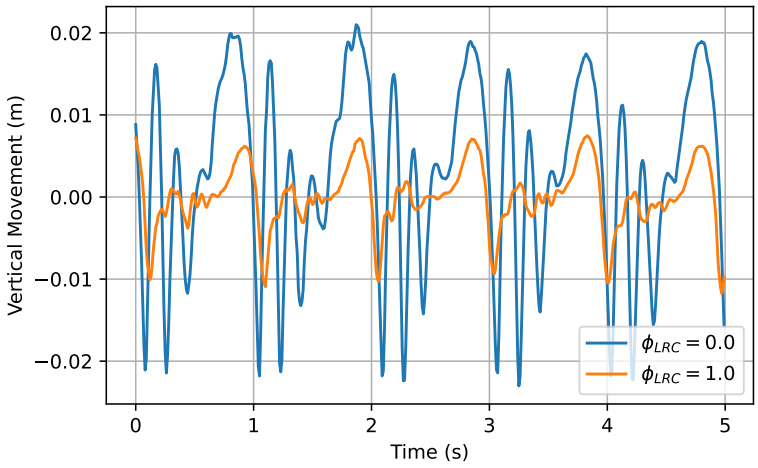
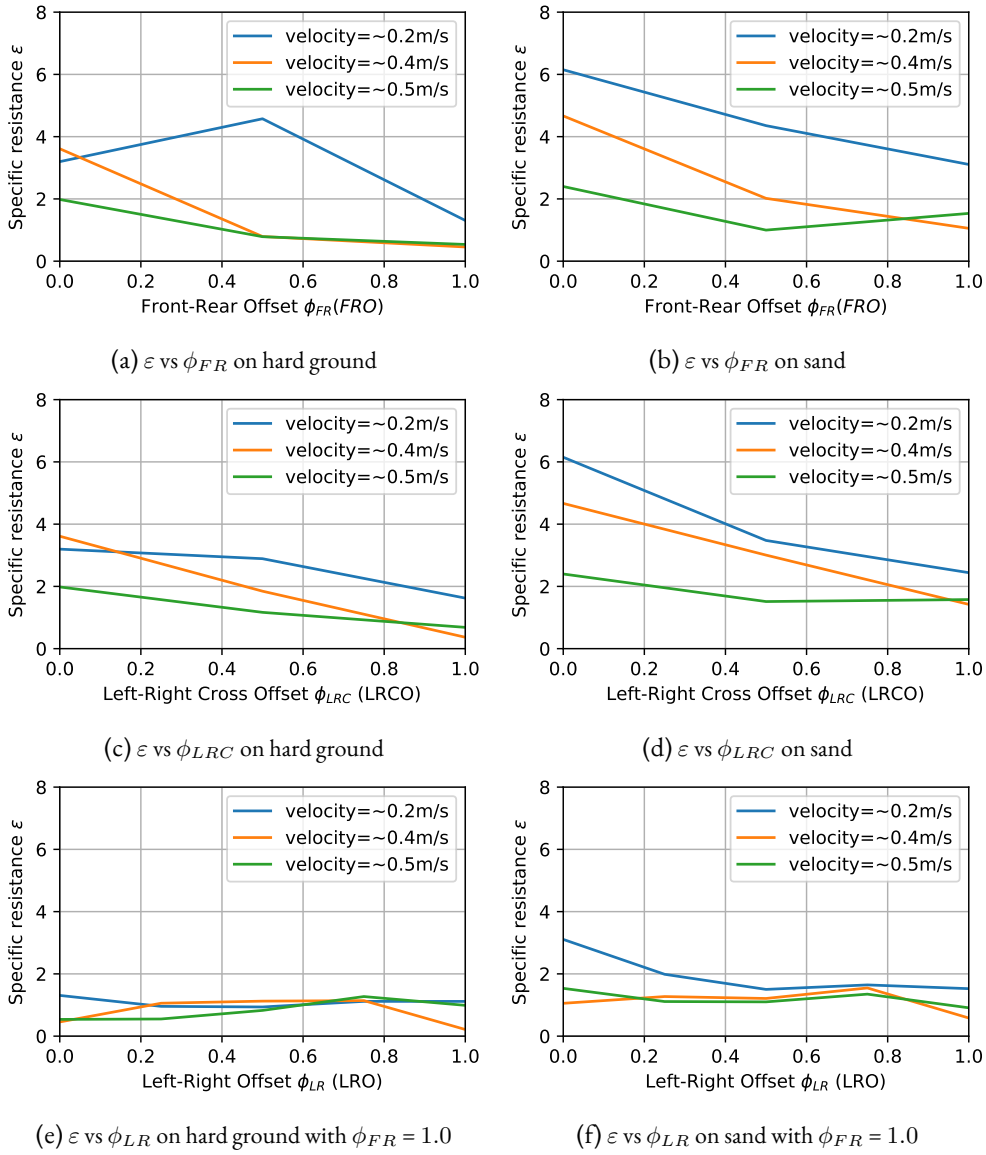


Figure 2.15: Comparison of vertical movement at $0.2\,\mathrm{m/s}$ for the locomotion patterns $\phi_{LRC} = 0.0$ and $\phi_{LRC} = 1.0$.

A sample of the data acquired from the motion capture system showing the vertical movement is shown in Figure 2.15. At ϕ_{LRC} =1.0, the vibrations are reduced to a great extent when compared to ϕ_{LRC} =0.0. The standard deviation of the vertical movement reduced by $66.2\,\%$ from $0.0116\,\mathrm{m}$ to $0.0039\,\mathrm{m}$. The vibrations show two different phases, one corresponding to the movement from DS to VS, as shown between the interval 0.5 and $1.0\,\mathrm{s}$. The second phase, for example, from 1.0 to $1.5\,\mathrm{s}$, shows the motion corresponding to transition from VS to DS. As it can be clearly seen, the vertical motion during both these phases is substantially reduced for ϕ_{LRC} =1.0. The second phase is more relevant due to the higher frequency of vibration that can increase the wear and tear of the robot hardware. Nevertheless, use of ϕ_{LRC} =1.0 introduces an additional rotational roll motion which can have effect on the perception sensors and their corresponding algorithms.

2.4.5 Locomotion Efficiency

The effect of change in specific resistance is shown to be dependent on velocity; lower velocities demonstrate higher responsiveness to changes in offsets, while higher velocities are more efficient, irrespective of the presence of offsets. This is more evident for the offsets ϕ_{FR} and ϕ_{LRC}, while less evident for the offset ϕ_{LRC}.



(a) ε vs ϕ_{FR} on hard ground

(b) ε vs ϕ_{FR} on sand

(c) ε vs ϕ_{LRC} on hard ground

(d) ε vs ϕ_{LRC} on sand

(e) ε vs ϕ_{LR} on hard ground with ϕ_{FR} = 1.0

(f) ε vs ϕ_{LR} on sand with ϕ_{FR} = 1.0

Figure 2.16: Specific resistance for different locomotion modes on hard ground (left) and on sand (right).

In general, increasing ϕ_{FR} resulted in enhanced efficiency across velocities and ground types. Figures 2.16a and 2.16b show the effect of increasing the ϕ_{FR} on sand and hard ground respectively. For example, at $0.2\,\mathrm{m/s}$ speed, increasing ϕ_{FR}, reduces specific resistance from 6.2 to 3.1 on sand. An exception to this trend was found on hard ground at $\phi_{FR} = 0.5$ while driving at a velocity of $0.2\,\mathrm{m/s}$ (Figure 2.16a) where the specific resistance is higher than the lower offset. This value seems to deviate from the trend and further investigation is required to determine its cause. Figures 2.16c and 2.16d show the effect of increasing ϕ_{LRC}. The trend here is similar to that of ϕ_{FR}.

Figures 2.16e and 2.16f show the effect of ϕ_{FR}=1.0 and varying ϕ_{LR}. The results show consistently low specific resistance and proving that only having ϕ_{FR} =1.0 could be used to improve the locomotion efficiency. This is particularly important given that the offset is unaffected by differential turning controllers. The average specific resistance for hard ground and sand with ϕ_{FR}=1.0 are 0.91 and 1.42

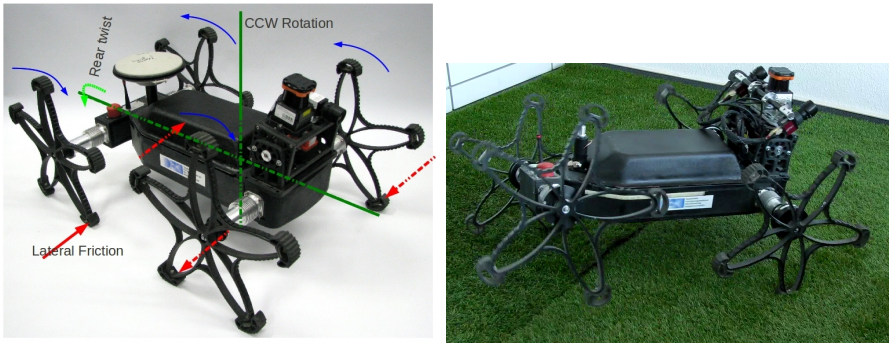
respectively, which is an efficiency improvement of 90 % and 77 % respectively, when compared to the corresponding worst-cases. The least specific resistance for both ground types were found to be when ϕ_{FR}=1.0 and ϕ_{LR}=1.0, at a longitudinal velocity of 0.5 m/s. In this case, specific resistances of 0.214 and 0.582 were recorded for hard ground and sand respectively.

Compared to other systems, Asguard's higher specific resistances are similar to those of many active walking robots. The lower range of specific resistances achieved by wheel synchronization is slightly higher than that of human running and older cars.

The type of ground does not seem to significantly impact the specific resistance trends. Although hard ground slightly reduces the specific resistance compared to sand, possibly due to the higher damping and traction in sand. The variance of the specific resistance was not determined and occasionally, some outliers were also found in the collected data.

2.5 Turn Movement Optimization

Differential steering makes it imperative that the robot wheels must slip laterally. On a high traction ground it is additionally hard, resulting in wheels unable to turn and triggering the over-current protection of the motors. During some instances, the passive joint of the robot could also flip, resulting in a very undesired pose for the robot. In this section, we discuss the turn dynamics and the controller used to improve the turning motion.



(a) Loads acting while turning (b) Passive joint flipping during turning

Figure 2.17: Dynamics of turning movement of Asguard (left) and the flipping (right) of rear during turning.

Several forces and torques should act on the robot to realize the turning motion. Figure 2.17a shows the various forces acting on the robot while trying to turn counter-clockwise (seen from the top of the robot). The front wheels are acted on by frictional forces from the left to the right of the robot, and the rear wheels from right to left. This force couple, since it is acting away from the axis of the passive joint, creates a torque about the passive joint. When this torque is high enough, the rear part, being lighter than the front part, twists. This results in lifting the rear-left wheel in air (Figure 2.17b), rendering it useless for turning tasks. In the resulting redistribution of loads, the front-left and the rear-right wheels take the maximum amount of load. These wheels become the critical wheels in the case of counter-clockwise rotation. For clockwise-rotation, the front-right and the rear-left wheels become the critical wheels.

In order to improve on the turn controller, the torque available to turn should be maximized. The proposed solution here, called the Load-Optimized Turning (LOT) controller, works by turning only

when the load torques of the pivot wheel is positive (Figure 2.10). When the load torque of the pivot wheel is negative, all four wheels act in the same direction repositioning the wheels to single stance. This also helps in untwisting the twist caused by the turn phase. Since the front part of the robot has higher weight distribution, the load torques of front wheels are higher. Hence, the pivot wheel is selected to be FL for counter-clockwise turn and FR for clockwise turn.

The pseudocode for the LOT-algorithm to convert the linear v_{cmd} and angular ω_{cmd} velocity commands to wheel velocities $\boldsymbol{\omega}_{ref}$ is detailed in Algorithm 1. In addition to the velocity commands, the normalized position of the wheels $\bar{\boldsymbol{\theta}}$ is given as input. The stance $\mathcal{S} \in \{\text{VS}, \text{DS}\}$ gives the last stance that was crossed by the pivot wheel which could be DS or VS. The motion mode $\mathcal{M}_{prev} \in \{\text{TM}, \text{FM}, \text{RM}\}$ has three different modes: TURN mode (TM), FORWARD mode (FM) and REVERSE mode (RM). The input also includes the normalized wheel positions $\bar{\boldsymbol{\theta}}^{prev}$, stance $\mathcal{S}_{prev}$ and motion mode $\mathcal{M}_{prev}$ from the last time step.

Initially the average wheel velocities ω_{avg} and the differentials Δ are computed. The pivot wheel is then selected based on the direction of the turn command. The algorithm also uses two functions CROSSED_VS and CROSSED_DS to determine if the pivot wheel has crossed the VS or DS respectively. If the pivot wheel has crossed the VS, the stance is set to VS and the motion mode is set to TM. If the pivot wheel has crossed the DS, the stance is set to DS and the motion mode is set to FM or RM based on the last motion mode. During TM, the pivot wheel which is in VS and can help in providing higher torques to turn. The wheels perform longitudinal forward and reverse motions during FM and RM to reorient the pivot wheel.

The algorithm then sets the wheel velocities based on the motion mode. The TURN motion mode sets the wheel velocities to turn the robot, with higher speeds for rear wheels. The FORWARD motion mode moves all the wheels forward and similarly the REVERSE mode moves all the wheels with negative velocities.

In order to evaluate the controller, both the normal point turn controller and the LOT-controller was tested[3] on artificial turf which has high traction. The results show that using the pure velocity-based normal point turn controller failed to complete even one full turn either due to over-current in the motors or due to flipping of the passive joint. On the contrary, the LOT-controller was able to finish more than 10 consecutive complete point-turns without any failure.

The use of this algorithm provides higher torque outputs during turning, reduces the flipping of the passive joint and also reduces the over-currents. This is achieved by ensuring that the robot turns only when the load torques of the pivot wheel are positive. The velocities and torques of all the four wheels while turning to the left (counter-clockwise seen from top) of the robot are shown in Figures 2.18 and 2.19 respectively. The reference values (orange color) and the actual value (blue) for both velocities and torques are shown in the plot. The controller mode (T: TURN, F: FORWARD, R: REVERSE) is also plotted as a dotted line. The highlighted background in the plot indicates TM and the white ones indicate either FM or RM. The reference velocity that is generated by the algorithm goes as input to the velocity-controller which in turn generates the reference torques.

The velocity controller is able to follow the commanded velocity albeit with high variance. The variance is higher on the FL and RR wheels due to high torques the wheels need. The torque plots similarly

[3]The deficiencies of the normal point turn controller and the turn motion based on the LOT-controller, being tested on an artificial turf, can be seen in the video provided in https://drive.google.com/file/d/16wpm6P0LwC7Mra9zl-QbRHdiGrfQO0LK/view?usp=sharing.

Algorithm 1 Generation of control velocity commands for improving turning motion using Load-Optimized Turning controller.

procedure LOTCONTROLLER($v_{cmd}, \omega_{cmd}, \bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\theta}}^{prev}, \mathcal{S}_{prev}, \mathcal{M}_{prev}$)

Require:

$v_{cmd} \in \mathbb{R}$: Linear velocity command

$\omega_{cmd} \in \mathbb{R}$: Angular velocity command

$\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\theta}}^{prev} \in [-\frac{\pi}{5}, \frac{\pi}{5}]^4$: Normalized leg stances (current and last time step) for wheels FL, RL, FR and RR respectively

$\mathcal{S}_{prev} \in \{\text{VS, DS}\}$: Stance of the pivot wheel in the last time step

$\mathcal{M}_{prev} \in \{\text{TM, FM, RM}\}$: Motion mode in the last time step

Ensure: $\boldsymbol{\omega}_{ref} \in \mathbb{R}^4, \mathcal{S} \in \{\text{VS, DS}\}, \mathcal{M} \in \{\text{TM, FM, RM}\}$

$r_w \leftarrow 0.178, w_t \leftarrow 0.515$ ▷ Wheel parameters constants

$\omega_{avg} \leftarrow \frac{v}{r_w}, \Delta \leftarrow \frac{\omega w_t}{r_w}$ ▷ Computes wheel velocities and differentials

$PW \leftarrow \begin{cases} FL, & \Delta > 0 \\ FR, & \text{otherwise} \end{cases}$ ▷ Selects the pivot wheel

$\mathcal{S} \leftarrow \mathcal{S}_{prev}, \mathcal{M} \leftarrow \mathcal{M}_{prev}$ ▷ Initialize stance and mode

if CROSSED_VS($\bar{\boldsymbol{\theta}}_{PW}, \bar{\boldsymbol{\theta}}^{prev}_{PW}$) **then**

  $\mathcal{S} \leftarrow$ VS ▷ Sets vertical stance

  $\mathcal{M} \leftarrow$ TM ▷ Sets TURN motion mode

end if

if CROSSED_DS($\bar{\boldsymbol{\theta}}_{PW}, \bar{\boldsymbol{\theta}}^{prev}_{PW}$) **then**

  $\mathcal{S} \leftarrow$ DS ▷ Sets stance

  if $\mathcal{M}_{prev} =$ FM **then**

    $\mathcal{M} \leftarrow$ RM ▷ Sets REVERSE motion mode

  else

    $\mathcal{M} \leftarrow$ FM ▷ Sets FORWARD motion mode

  end if

end if

if $\mathcal{M} =$ TM **then**

  $\boldsymbol{\omega}_{ref} \leftarrow \omega_{avg}\begin{bmatrix}1.0 & 1.0 & 1.0 & 1.0\end{bmatrix}^T + \Delta\begin{bmatrix}-2.0 & -4.0 & +2.0 & +4.0\end{bmatrix}^T$ ▷ Sets wheel velocities for TURN motion

else if $\mathcal{M} =$ FM **then**

  $\boldsymbol{\omega}_{ref} \leftarrow (\omega_{avg} + \Delta)\begin{bmatrix}1.0 & 1.0 & 1.0 & 1.0\end{bmatrix}^T$ ▷ Sets wheel velocities for FORWARD motion

else if $\mathcal{M} =$ RM **then**

  $\boldsymbol{\omega}_{ref} \leftarrow (\omega_{avg} - \Delta)\begin{bmatrix}1.0 & 1.0 & 1.0 & 1.0\end{bmatrix}^T$ ▷ Sets wheel velocities for REVERSE motion

end if

return $\boldsymbol{\omega}_{ref}, \mathcal{S}, \mathcal{M}$

end procedure

show the reference and actual torques of the wheels. Since the controller is turning to the left of the robot, the torques of the FL and RR wheels are higher compared to the other wheels. The use of the LOT controller ensures that the wheels are able to generate high torques, especially during the turn-

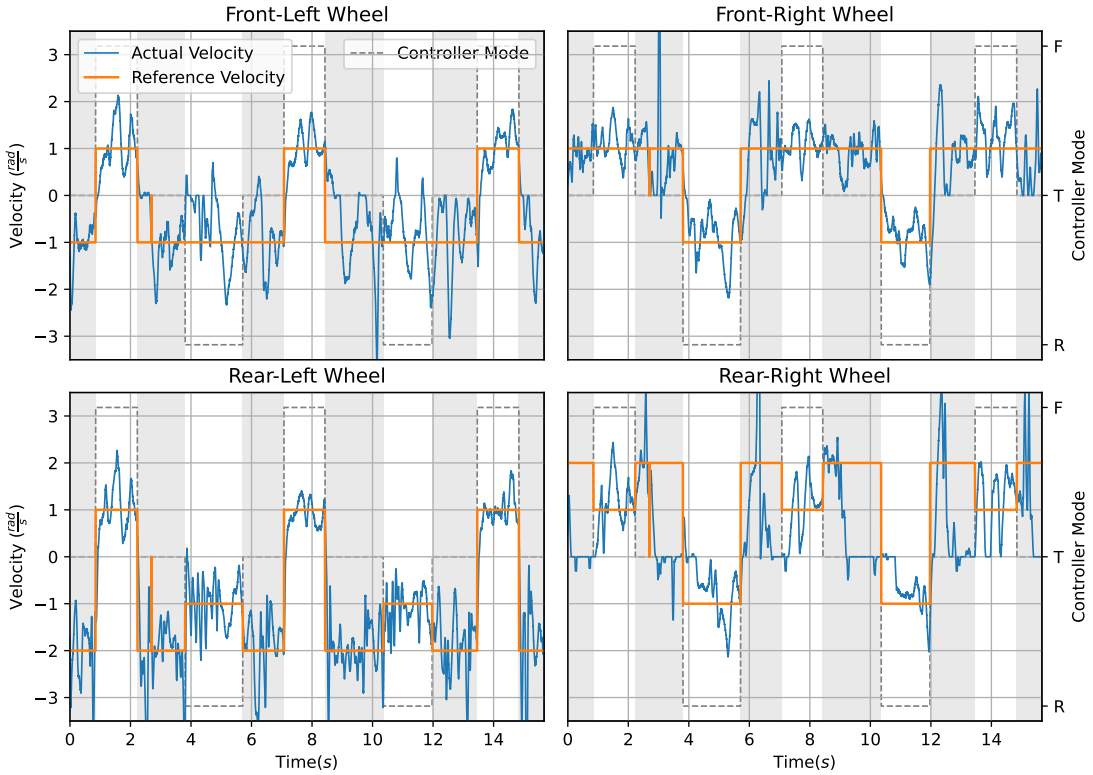


Figure 2.18: Reference (orange) and actual (blue) velocities while turning using LOT controller.

ing phase. The use of torque controller in the cascaded loop ensures that the motor does not cross the current limits and hence stays protected from over-currents.

The LOT-controller is slower compared to simple differential steering, hence activated only if the simple differential steering cannot cope with the loads. This has the potential to improve the efficacy of the turning motion on substrates that provide higher lateral friction. If the ground has very high friction or is an uneven solid surfaces (log pile or stones), the controller may still fail.

2.6 Summary and Discussions

This chapter details the research endeavors undertaken to enhance the locomotion efficiency of the passive-hybrid robot, Asguard. The robot's design, integrating features of both wheeled and legged systems, exhibits suboptimal performance when utilizing conventional wheel-based controllers. To address this limitation, the controller enhancement should incorporate the legged aspect of the design while using only the existing DoFs. The control of this under-actuated system is comprised of three primary components: the joint controller, the analysis of wheel synchronization during longitudinal movement, and the turn motion optimization.

A simple velocity controller on the motor side is inadequate for precise position control of the wheel side, a deficiency attributable to the flexible coupling that connects the motor and the wheel. To address this limitation, a cascaded position-velocity-torque controller has been developed to enhance the wheel position control accuracy. The innermost torque loop necessitates the measurement of the

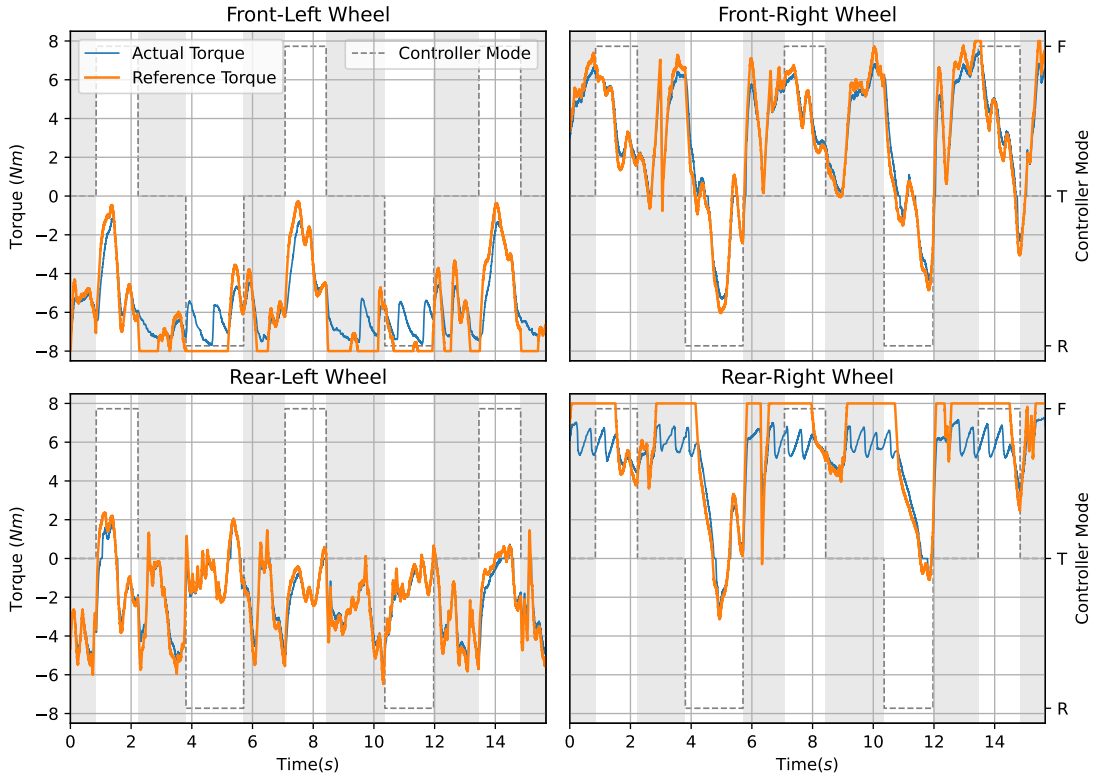


Figure 2.19: Reference (orange) and actual (blue) torques while turning using LOT controller.

torques acting on the wheel, which is accomplished by measuring the deflection of the flexible coupling using encoders on both the motor and wheel sides. The relationship between the deflection and the actual torque is modeled using the Bouc-Wen model hysteresis model, which is extended to include damping and effects of gear play. The estimated model demonstrated an accuracy of over 95 % in its representation of the actual torques, while the torque controller exhibited an average error of approximately 6.3 % in its tracking of the reference torque. This cascaded controller was able to enhance the accuracy of the wheel position control by 56 %, as compared to the controller without the torque loop. The controller also leveraged the anticipated load torques, operating under the assumption of quasi-static motion, as feed-forward to further enhance performance.

The wheels, which can be likened to multiple straight legs, can be synchronized to generate different locomotion patterns during longitudinal motion. Our approach endeavors to reduce energy loss during locomotion by enhancing the potential and kinetic energy exchange within the system. This was achieved by controlling the wheel synchronizations, specifically when the wheels' legs contact the ground relative to each other. Three different offsets were defined to model this and to generate the reference trajectories. The variation in the specific resistances was found to range from that of walking robots to that of human running or that of older cars. The most significant observation was that the specific resistances were lowest when the offset between the front and rear wheels was at its maximum. This is irrespective of the left-right offsets, meaning that the high efficiency can be maintained even while turning. It was also observed that the specific resistance can be reduced by up to 90 % when compared with the worst-case scenario. In the future, an automated method for determining efficiency could be developed to extend the study to higher speeds and further locomotion patterns.

Additionally, the use of the LOT controller enhanced turning on high-traction surfaces. The controller ensures that the robot turns only when the load torques of the pivot wheel are positive, thereby enhancing the controller's effectiveness, though at the expense of turn speed. During phases where the load torques are negative, the robot moves back and forth to reposition the wheels. The LOT controller reduced instances of flipping and over-currents during turning. During the testing phase on the artificial turf surface, the conventional controller demonstrated an inability to complete even a single turn, whereas the LOT controller exhibited the capacity to execute multiple turns without encountering any errors.

In general, the controllers developed for Asguard demonstrated a marked improvement in locomotion efficiency during longitudinal motion and efficacy for turn motion, successfully achieving the objectives O-1a, O-1b and O-1c.

Motion Control System for a Wheel-on-Leg Planetary Rover

This chapter presents the controllers for the innovative wheel-on-leg planetary exploration rover, SherpaTT. The focus of this chapter is on the development of the software control framework SherpaTT-Motion Control System (MCS) and the controller for adapting to uneven terrain for SherpaTT. Partial results of the presented work have been published in

(i) Cordes, F., A. Babu, and T. Stark (2024). "Sherpa, a family of wheeled-leg rovers". In: *Biologically Inspired Series-Parallel Hybrid Robots*. Ed. by S. Kumar, A. Mueller, and F. Kirchner. 1. Auflage. Vol. 514. Elsevier Science, pp. 281–304. ISBN: 978-0-323-88482-2. URL: https://shop.elsevier.com/books/biologically-inspired-series-parallel-hybrid-robots/kumar/978-0-323-88482-2

(ii) Cordes, F., F. Kirchner, and A. Babu (2018). "Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain". *Journal of Field Robotics* 35:7, pp. 1149–1181. DOI: 10.1002/rob.21808. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21808

(iii) Cordes, F., A. Babu, and F. Kirchner (2017). "Static Force Distribution and Orientation Control for a Rover with an Actively Articulated Suspension System". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2017)*

(iv) Babu, A. (2016). *Ground Adaption Process for SherpaTT*. technical report. DFKI GmbH, pp. 84–91

(v) Cordes, F. and A. Babu (2016). "SherpaTT: A Versatile Hybrid Wheeled-Leg Rover". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2016), June*

(vi) Cordes, F., C. Oekermann, A. Babu, D. Kuehn, T. Stark, and F. Kirchner (2014). "An active suspension system for a planetary rover". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014), June*, pp. 17–19

This chapter is structured as follows: After introducing SherpaTT in Section 3.1, and the description of the robot system in Section 3.2, the control software is elaborated in Section 3.3. The subsequent section, Section 3.5, details a controller that adapts to uneven terrain using a custom developed controller. This section also contains details of the experimental setup for evaluating the controller, as well as the results. Finally, the chapter is summarized, and the contributions are listed.

3.1 Adaptive Suspension Rover

Mobile robots have been utilized in extraterrestrial applications to facilitate exploration of the environment, soil sample collection, and other measurements. Despite their durability, these robots have historically exhibited conservatism with respect to terrain difficulty, a consequence of their constrained locomotion capabilities. It is noteworthy that some scientifically intriguing locations are situated in challenging terrains, as evidenced by the works of Huntsberger et al. 2007 and Nesnas et al. 2012. Accessing these sites necessitates different locomotion approaches: an energy-efficient locomotion for traversing large distances and another to adapt to steep slopes and uneven terrains. This objective is achieved by combining these locomotion modes into the same design, the wheel-on-leg design. This hybrid locomotion design combines the efficiency of wheeled locomotion with the adaptability of legged locomotion. The legged design can even be utilized to step over some obstacles.

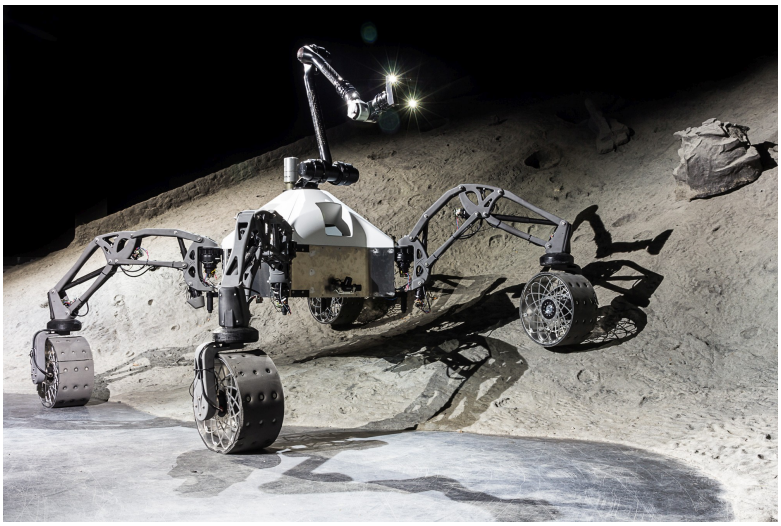


Figure 3.1: The SherpaTT robot is a planetary rover with an adaptable suspension system developed for planetary exploration. The picture shows the robot placed in an artificial crater environment at DFKI.

The SherpaTT robot (Figure 3.1) was developed at DFKI for the purpose of traversing irregular terrain with varying substrates in planetary exploration scenarios. The wheel-on-leg design of this novel rover was developed as part of the TransTerrA (R. Sonsalla et al. 2014) project. The active suspension system offers several advantages, including: (i) high terrain mobility for the rover, (ii) ability to actively move the robot's Center of Gravity (CoG) with respect to the footprint, (iii) control of contact forces at wheel-ground contact points (load distribution), (iv) orienting the main body irrespective of the terrain, and (v) ability to move the body without moving the wheels. In most cases, the legs are capable of adapting to the ground using force-torque sensors without the need to stop and reconfigure.

The rover is capable of manipulating and deploying other compatible modules with its custom-designed Electromechanical Interface (EMI) on the manipulator or the bottom, as detailed in R. Sonsalla et al. 2017. The control software structure SherpaTT-MCS and the Ground Adaption Process (GAP) controller, which are described here, serve to abstract the complexity of controlling the robot to high-level commands. The controls developed in this chapter are based on low-level sensors, such as force-torque sensors, orientation measurement, and a reactive control approach. High-level control approaches, including the use of a height map of the environment and motion planning for the system or suspension

configuration planning, are not utilized for ground adaption of SherpaTT. The thesis objectives O-2a and O-3a are addressed here.

3.2 System Description - SherpaTT

The SherpaTT system has three main mechanical components: (i) body, (ii) legs and (iii) manipulator. The body serves as the central component to which the four identical legs and the manipulator are attached. The four legs are attached to the body in such a manner that they are, in their default positions, horizontally and laterally symmetric. The manipulator is attached to the top part, ensuring even reachability to all the sides of the robot. The manipulator possesses six DoFs, with an EMI-interface serving as the end-effector, facilitating attachment to compatible interfaces such as a battery module, camera module, sampling module, or other small robots (Dettmann et al. 2011). The manipulator can also be utilized to support locomotion during stepping, a mode not addressed in this chapter. The legs, being the most critical component for locomotion, feature a wheel-on-leg design with three DoFs, enabling the leg part to move the wheel freely in 3D. The wheel part has two DoFs for steering and driving. The wheels can have different designs based on the application and terrain substrate type.

The control and corresponding software design can be categorized as reactive and deliberative layers. The reactive layer responds directly to sensor input by transmitting joint commands, whereas the deliberative layer accumulates sensor data to construct a comprehensive environmental model and subsequently plans actions for both immediate and long-term contexts. The low-level control of joints and data acquisition of sensors is facilitated by Field Programmable Gate Array (FPGA)-based electronics, which serve to implement the controllers and communicate with higher-level computers. The reactive layer constitutes the mid-level control, which is elaborated upon in Section 3.3. Similarly, the deliberative layer constitutes the high-level control and is described in Section 3.4. The mid-level and high-level control software is based on the ROCK framework (Joyeux and Albiez 2011).

3.2.1 Leg Kinematics

Each leg of SherpaTT consists of five joints: *Pan*, *Inner-Leg*, *Outer-Leg*, *Wheel-Steering* and *Wheel-Drive*. The joints, Inner-Leg and Outer-Leg, use parallel mechanisms to transform the actuation motion to joint motion. Both these joints use parallelograms for this purpose which has the property of imparting translational motion to the next link without any rotational movements. Modeling of the parallelogram is performed in the low-level joint control and is abstracted to the higher control layers for which the legs are modeled as a serial kinematic chain. A conversion between the linear actuator length pushing the parallelogram and the virtual rotational joint is performed locally in the joint actuator's control electronics. Once the leg is considered as a serial kinematic, well-known modeling procedures can be employed. The forward and inverse kinematic equations, as well as the Jacobian, are described here. The structure of the SherpaTT leg is shown in Figure 3.2.

Let $\begin{bmatrix} X_0 & Y_0 & Z_0 \end{bmatrix}^T$ be the coordinates of the translation of the leg coordinate system from the Body Coordinate System (BCS). L_{il} and L_{ol} are the lengths of the links corresponding to Inner-Leg and Outer-Leg respectively. Let the position of the joints Pan, Inner-Leg, Outer-Leg and Wheel-Steering

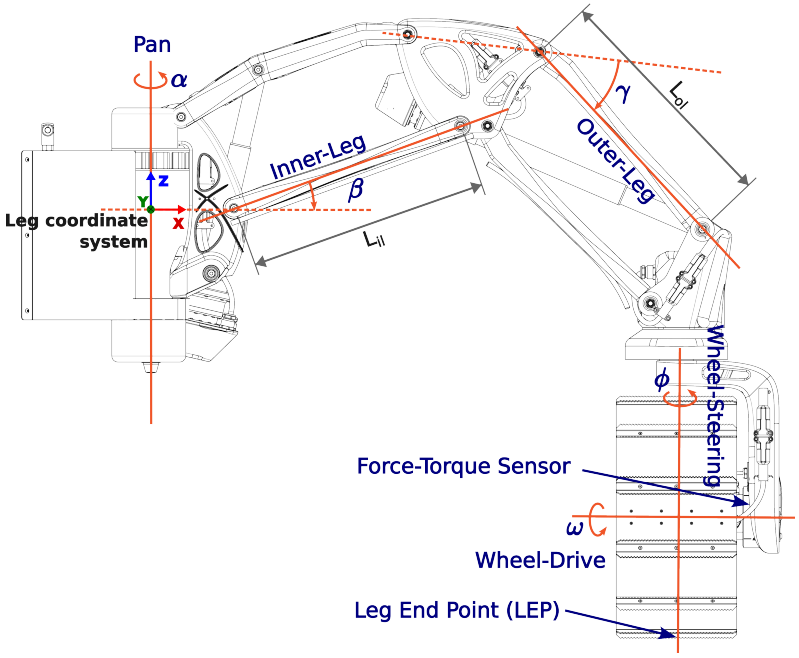


Figure 3.2: Design of the SherpaTT leg depicting the parallel-kinematic structure, the serial kinematic joints, and the location of the force-torque sensor and the LEP.

be represented by α, β, γ and ϕ respectively. The position of LEP can be computed using the first three joints as

$$P_{LEP}(\alpha,\beta,\gamma) = \begin{bmatrix} X_0 + (X_c + L_{il}\cos\beta + L_{ol}\cos\gamma)\cos\alpha \\ Y_0 + (Y_c + L_{il}\cos\beta + L_{ol}\cos\gamma)\sin\alpha \\ Z_0 + Z_c + L_{il}\sin\beta + L_{ol}\sin\gamma \end{bmatrix}, \tag{3.1}$$

where X_c, Y_c and Z_c are the constant offsets in x, y and z directions within the BCS. They are calculated by adding the all the constant offsets. Computing the Jacobian based on Equation (3.1) will result in

$$\mathbb{J}(\alpha,\beta,\gamma) = \begin{bmatrix} (-r - X_c)\sin\alpha & -L_{il}\sin\beta\cos\alpha & -L_{ol}\sin\gamma\cos\alpha \\ (r + Y_c)\cos\alpha & -L_{il}\sin\alpha\sin\beta & -L_{ol}\sin\alpha\sin\gamma \\ 0 & L_{il}\cos\beta & L_{ol}\cos\gamma \end{bmatrix}, \tag{3.2}$$

where $r = L_{il}\cos\beta + L_{ol}\cos\gamma$.

The inverse kinematic of the leg is also computed by considering only the first three joints. The last two joints, Wheel-Steering and Wheel-Drive, do not contribute to the position of the Leg End Point (LEP). In order to compute the inverse kinematic, the LEP coordinate system is converted to cylindrical coordinates with respect to the leg coordinate system placed in the Pan joint. The objective is to compute the joint angles α, β and γ, corresponding to the Pan, Inner-Leg and Outer-Leg joints. Let the desired cylindrical coordinate for LEP be $\begin{bmatrix} r & h & \theta \end{bmatrix}^T$, corresponding to radius, height and angle,

respectively. Since the coordinate origin coincides with Pan joint $\alpha = \theta$. The computation for β and γ, given r and h is as explained below. Let

$$t_0 = \frac{r^2 + h^2 + L_{il}^2 - L_{ol}^2}{2L_{il}}, \tag{3.3}$$

$$t_1 = r^2 + h^2 - t_0^2, \tag{3.4}$$

and

$$t_2 = \frac{r^2 + h^2 - L_{il}^2 - L_{ol}^2}{2L_{il}L_{ol}}. \tag{3.5}$$

A solution exists only if $t_1 >= 0$, else the solution is imaginary. For real solutions, the following solves for Inner-Leg

$$\beta = \frac{2\arctan\left(h \pm \sqrt{t_1}\right)}{r + t_0}, \tag{3.6}$$

and the following for the Outer-Leg

$$\gamma = \beta \pm \arccos\left(t_2\right). \tag{3.7}$$

3.3 Structure of SherpaTT-MCS

The mid-level control software of SherpaTT, which acts as the interface between the commands from the operator or high-level and the joint-level control, is called the SherpaTT-MCS. The general structure of the SherpaTT-MCS is shown in Figure 3.3. This section gives an overview and details of the different modules. A more detailed discussion is provided in Cordes 2018; Cordes et al. 2024; 2018.

The SherpaTT-MCS provides a reactive layer of controllers that are necessary for the proper functioning of the higher level software (e.g. autonomy modules). The main functionalities of SherpaTT-MCS are (i) *Driving* for moving the platform in a desired direction using the wheels, (ii) *Leg movement* to change the footprint and to react to uneven terrain, and (iii) *Assistive controllers* that abstract the complexity for the higher level and also ensures the safety of the robot.

The architecture of the SherpaTT-MCS is composed of two primary input groups: high-level commands and sensor inputs. High-level commands represent the instructions issued by high-level controllers or an operator during remote operation, while sensor inputs refer to the feedback received from sensors that inform the state of the controllers. The SherpaTT-MCS outputs are also comprised of two groups: joint commands and operator info. Joint commands refer to controller commands issued to the low-level, which govern the movement or positioning of individual joints in velocity or position modes. The term operation info signifies the information disseminated to the operator regarding the system's status.

The high-level commands are motion command 3D, footprint and body-posture. The motion command 3D has commands for the longitudinal, lateral and rotational velocity of the robot body, generated solely by the motion of the wheels. In contrast, the commands of the motion of the legs are provided by the footprint and body-posture commands. The footprint command gives the position of the individual LEP in the leg coordinate system, either as Cartesian or as Cylindrical coordinates. This command moves the leg joints to reach a desired LEP. The body-posture command enables the

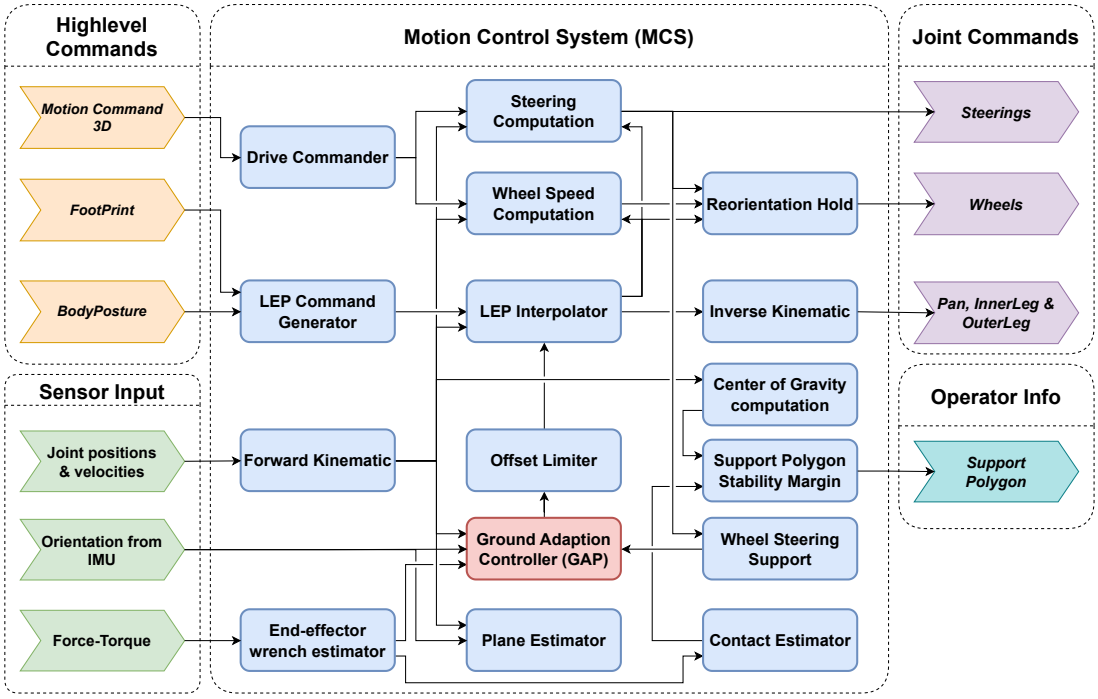


Figure 3.3: Block diagram depicting the structure and connections of the mid-level control for SherpaTT called the SherpaTT-MCS.

manipulation of the robot's body in 6D, through the coordination of its leg joints to achieve a desired offset.

The sensor inputs consist of the position and velocities of all the joints, orientation (roll and pitch) from the IMU, and data from force-torque sensor. The force-torque sensor values are 3D forces and 3D torques.

The drive control commands is responsible for generating the reference commands for the steering wheel's angular displacement and the velocities of the vehicle's wheels, thereby facilitating its movement on the ground. The control of drive movement involves the integration of multiple modules. The drive commander serves as a converter, translating the input motion command 3D into the desired velocities in the body's coordinate system. These converted commands are then utilized by the modules of steering computation and wheel speed computation to calculate the steering angles and the wheel's rotational velocities, respectively. The module reorientation hold is designed to halt the rotation of the wheels in the event that the desired and actual steering angles are not aligned.

The movement of the legs is governed by high-level commands, with the commands foot-print and body-posture being integrated by the module LEP command generator to generate the desired LEP commands. These commands are subsequently transmitted to the LEP interpolator. The interpolator then performs interpolation of the movement of the LEP, ensuring that joint velocities remain within predefined limits. This interpolation is executed in the cylindrical coordinate of the leg, thereby minimizing the necessary movement of the wheel steering. The interpolated LEP positions are then transmitted to the inverse kinematic module, as detailed in Section 3.2.1. This module converts the interpolated LEP positions into joint positions, which are then utilized to command the joint electronics.

One of the most useful assistive functions is the GAP module, which has the primary function of maintaining contact with the ground. GAP modules require input from forward kinematics, orientation from IMU and end-effector wrench estimator to provide LEP positions, robot body orientation and contact forces on the wheel, respectively. Utilizing these inputs, the GAP modules regulate the vertical position of the leg, thereby enabling indirect control over contact, contact forces and body orientation (roll and pitch). The vertical offsets output by the GAP module are written to the LEP commands, which are limited by the offset limiter module. Various implementations of the GAP module have been developed in Cordes et al. 2018 and Cordes 2018. The implementation of the controller for controlling different factors simultaneously is detailed in Section 3.5.

In addition to the previously mentioned modules, several others have been implemented to facilitate the robot's operation. The CoG is computed and provided as input to the support polygon module to estimate the stability of the current posture. This information can then be visualized by the operator. Under certain conditions, such as high ground friction, the wheel steering may lack sufficient power to turn. The wheel steering support module has been developed to address this issue by releasing the vertical load to reduce friction. Additionally, the module contact localizer has been implemented to localize the contact point and force on the wheel. This module is intended to be used in the future to improve stability estimation and reduce slippage.

3.4 High-level Control

The high-level control modules of the robot are responsible for Simultaneous Localization And Mapping (SLAM), path planning and execution of the robot base, and trajectory planning and execution of the manipulator. These functionalities are then used to create missions with more autonomy features. The high-level commands are then used as input to SherpaTT-MCS.

The SLAM, a custom implementation known as Slam3D, is open-source, and can be found in DFKI Slam3D. The library provides interfacing for both ROS and ROCK and Slam3D offers a frontend for graph-based SLAM in 3D space. Different solvers for graph optimization can be implemented as backends by providing the interface. Slam3D is capable of utilizing point-cloud data, GPS data and wheel odometry data as constraints for optimizing the graph

A path planner generates a path from a given pose to a target pose based on the obstacles and terrain map generated by the SLAM module. The path planning is a custom implementation based on Limpert et al. 2015 in Robot Construction Kit (ROCK), which uses search-based planning on graphs in discrete environments. A set of motion primitives is generated a priori based on the robot's kinematic constraints. These primitives are then combined to form a graph, with consideration given to collisions and the validity of the state. The final path is computed using the ARA* algorithm developed in Likhachev et al. 2003. The generated path is then followed using the trajectory tracking algorithm described in Micaelli and Samson 1993.

ROCK-based motion planning for manipulators, as described in Asadi and Natarajan 2014, is a custom implementation based on Open Motion Planning Library (OMPL) (Şucan et al. 2012). Additionally, it interfaces with other optimization-based planners. The kinematic library and collision computations are abstracted such that other algorithms can be easily interfaced.

3.5 Ground Adaption Controller using Force-Torque Sensors

GAP is a vital assistance function that serves to reduce the operational burden on the operator or high-level algorithms. The primary objectives of the controller are to (i) maintain wheel contact with ground for stability, and (ii) distribute the loads more evenly. The secondary objectives include (i) maintaining a desired ground clearance, and (ii) orientations (roll and pitch). These objectives are achieved by controlling the vertical motion of the legged wheels. The controller uses input from sensors, including vertical forces from the force-torque sensor, orientation from the IMU, and joint position and velocity measurements.

3.5.1 Controller Design

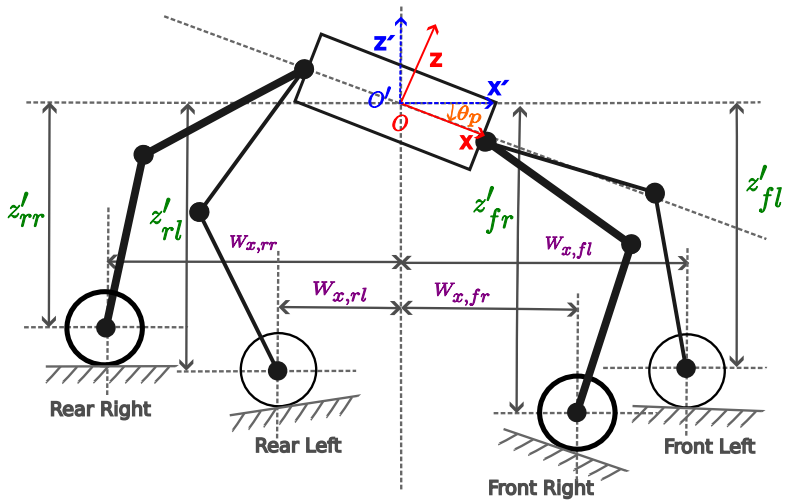


Figure 3.4: Depiction of the kinematic model for the GAP controller.

The controller maps the joint velocities of the leg joints to the locomotion factors which are defined by the GAP objective. The leg joints that influence the vertical motion are Inner-Leg and Outer-Leg, whose positions are denoted by β_i and γ_i, respect. i is the leg index with values $\begin{bmatrix}0 & 1 & 2 & 3\end{bmatrix}$, representing FL, FR, RL and RR legs respectively. The corresponding joint velocities are denoted by $\dot{\beta}_i$ and $\dot{\gamma}_i$.

The locomotion factors that are being controlled are (i) the mean height of all legs (h_m), (ii) the cross–force difference (f_c) (iii) the roll angle of the body (θ_r), and (iv) the pitch angle of the body (θ_p).

The factors represent the objectives of the GAP controller, with the mean height employed to maintain a desired ground clearance and the cross-force ensuring ground contact of the wheels and stability. The roll and pitch angles can be controlled to maintain a desired orientation.

As illustrated in Figure 3.4, the schematics for deriving the equations for the GAP controller are shown. It presents a side view of the body, with the pitch and the legged wheels in contact with an uneven ground. Two distinct coordinate systems are employed: the BCS, with its center at O; and the Gravity Aligned Coordinate System (GCS), with its center at O'. The BCS is attached to the robot body and moves in conjunction with it. The yaw angle of the GCS is equivalent to that of the BCS, and the roll and pitch angles of the GCS are set to zero. The z-axis of the GCS aligns with the gravitational vector. The roll and pitch offsets between the BCS and the GCS are θ_r and θ_p, respectively.

In order to derive the controller equations, it is assumed that the robot's footprint remains constant while the controller is operational. The vertical position of the legs is denoted by z_i' and the x-offsets are denoted by $W_{x,i}$. Analogously, though not illustrated in the schematics, the y-offsets are denoted by $W_{y,i}$.

The leg heights for all legs are defined by $\mathcal{Z}' = \begin{bmatrix} z'_{fl} & z'_{fr} & z'_{rl} & z'_{rr} \end{bmatrix}^\top$ with the values given in the third row of Equation (3.1) for each leg. It is important to note that, given the nature of the GAP controller, which allows for vertical movement exclusively, no motion occurs along the X and Y axes. The mapping of the vertical motion of the legs to the locomotion output that is being controlled is given by

$$\mathcal{Z}' = \begin{bmatrix} h_m + h_c - W_{y,fl} \sin\theta_r + W_{x,fl} \sin\theta_p \\ h_m - h_c + W_{y,fr} \sin\theta_r + W_{x,fr} \sin\theta_p \\ h_m - h_c - W_{y,rl} \sin\theta_r - W_{x,rl} \sin\theta_p \\ h_m + h_c + W_{y,rr} \sin\theta_r - W_{x,rr} \sin\theta_p \end{bmatrix},$$

where h_m is the mean height of all legs, h_c is the cross height offset between diagonally opposite legs, $W_{x,fl}$ and $W_{y,fl}$ are the X and Y positions of the LEP in the BCS respectively, and θ_r and θ_p are the body roll and pitch angles respectively.

Taking the time derivative of the above equation and separating the locomotion control terms results in

$$\dot{\mathcal{Z}}' = \begin{bmatrix} 1 & +1 & -W_{y,fl}\cos\theta_r & +W_{x,fl}\cos\theta_p \\ 1 & -1 & +W_{y,fr}\cos\theta_r & +W_{x,fr}\cos\theta_p \\ 1 & -1 & -W_{y,rl}\cos\theta_r & -W_{x,rl}\cos\theta_p \\ 1 & +1 & +W_{y,rr}\cos\theta_r & -W_{x,rr}\cos\theta_p \end{bmatrix} \begin{bmatrix} \dot{h_m} \\ \dot{h_c} \\ \dot{\theta_r} \\ \dot{\theta_p} \end{bmatrix}. \tag{3.8}$$

These intermediate and decoupled controlled variables can be now written as a proportional controller in terms of the error as below

$$\begin{bmatrix} \dot{h_m} \\ \dot{h_c} \\ \dot{\theta_r} \\ \dot{\theta_p} \end{bmatrix} = \begin{bmatrix} K_h(h_m - h_{m,ref}) \\ K_f(f_{\bar{z},fl} - f_{\bar{z},fr} - f_{\bar{z},rl} + f_{\bar{z},rr}) \\ K_r(\theta_r - \theta_{r,ref}) \\ K_p(\theta_p - \theta_{p,ref}) \end{bmatrix}, \tag{3.9}$$

where $h_{m,ref}$ is the reference mean height, $f_{\bar{z},i}$ are the vertical ground reaction forces for each wheel, and $\theta_{r,ref}$ and $\theta_{p,ref}$ are the roll and pitch references respectively. K_h, K_f, K_r and K_p are the proportional gains for the height, cross-force, roll and pitch controllers respectively.

The complete controller equation is formed by combining Equations (3.8) and (3.9)

$$\dot{\mathcal{Z}}' = \begin{bmatrix} 1 & +1 & -W_{y,fl} & +W_{x,fl} \\ 1 & -1 & +W_{y,fr} & +W_{x,fr} \\ 1 & -1 & -W_{y,rl} & -W_{x,rl} \\ 1 & +1 & +W_{y,rr} & -W_{x,rr} \end{bmatrix} \begin{bmatrix} K_h h_{error} \\ K_f f_{error} \\ K_r \cos\theta_r \theta_{r,error} \\ K_p \cos\theta_p \theta_{p,error} \end{bmatrix}. \tag{3.10}$$

The above equation gives the vertical velocity of the leg in terms of the locomotion control terms in GCS. These velocities can be transformed in to BCS by rotating about X-axis by θ_r and Y-axis by θ_p. If the θ_r and θ_p values are close to zero, the controller can be approximated by the following equation,

where the cosine terms are approximated as 1.0. This can be directly used without transforming the coordinates to give

$$\dot{Z}' = \begin{bmatrix} 1 & +1 & -W_{y,fl} & +W_{x,fl} \\ 1 & -1 & +W_{y,fr} & +W_{x,fr} \\ 1 & -1 & -W_{y,rl} & -W_{x,rl} \\ 1 & +1 & +W_{y,rr} & -W_{x,rr} \end{bmatrix} \begin{bmatrix} K_h h_{error} \\ K_f f_{error} \\ K_r \theta_r \\ K_p \theta_p \end{bmatrix}. \tag{3.11}$$

The joint velocities can be computed by using the Jacobian defined in Equation (3.2). The Z-axis velocities for the legs are provided by Equation (3.11). Plugging in these values into the Jacobian equation and inverting the Jacobian will give the joint velocities. Alternatively, the LEP velocities can be integrated and given as offsets to the inverse kinematics module of SherpaTT-MCS.

The controller gains are tuned separately such that the controlled variables converge for step input with a bit of overshoot, but settles shortly afterwards. This gives adequate performance during continuous operation and the controllers responds adequately. The tuned values are $K_h = 0.01$, $K_f = 1.0 \times 10^{-7}$, $K_r = 0.003$ and $K_p = 0.003$. Setting a gain to 0.0 will disable that particular controlled variable.

3.5.2 Experimental Setup

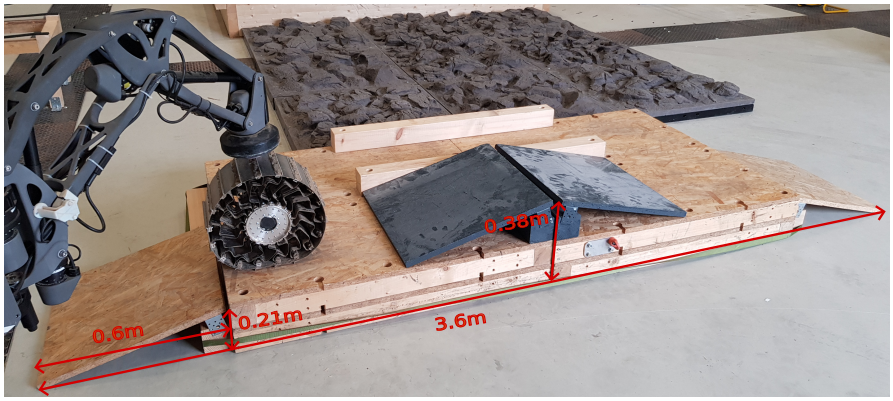


Figure 3.5: Experimental setup for evaluation of GAP using wooden boards for emulating an uneven terrain. The setup has a length of 3.6 m, maximum height of 0.38 m and a maximum slope of ∼20°.

Multiple experiments were conducted to evaluate the performance of the controller. The first set of experiments is designed to evaluate the performance of the controller during the driving of the rover over uneven terrain. For this purpose, an experimental setup was developed with wooden boards assembled to emulate a varying uneven terrain (see Figure 3.5 for setup and dimensions). The robot moves forward at a velocity of 0.02 m/s. The left wheels of the robot traversed an obstacle, while the right wheels moved on a flat surface. Initially, a reference run is conducted without activating the GAP-controllers. This is followed by a run with the controller activated, with all reference values set to zero, and one with both the roll and pitch controllers deactivated. The results from the experiments are presented in the following section.

In the next set of experiments, the step response of the individual components of the controllers are evaluated. The robot is placed on a flat surface with all the wheels on ground and the GAP-controller

activated. Then a reference input is given as step input separately. Values of ± 0.1 m, ± 0.1 rad and ± 0.1 rad were used respectively for mean height ($h_{m,ref}$), roll (θ_r) and pitch (θ_p). The controllers used for the experiments are the one in Equation (3.11) along with the offsets for legs, instead of Jacobian.

The step response for the simultaneous input for all the components are also evaluated. In this experiment, values of {0.1 m, 0.05 rad, 0.05 rad}, {-0.1 m, -0.05 rad, -0.05 rad} and {0.0 m, 0.0 rad, 0.0 rad} were given as step input in sequence. Each of the set of values represent mean height ($h_{m,ref}$), roll (θ_r) and pitch (θ_p) respectively.

3.5.3 Performance of Ground Adaption Process

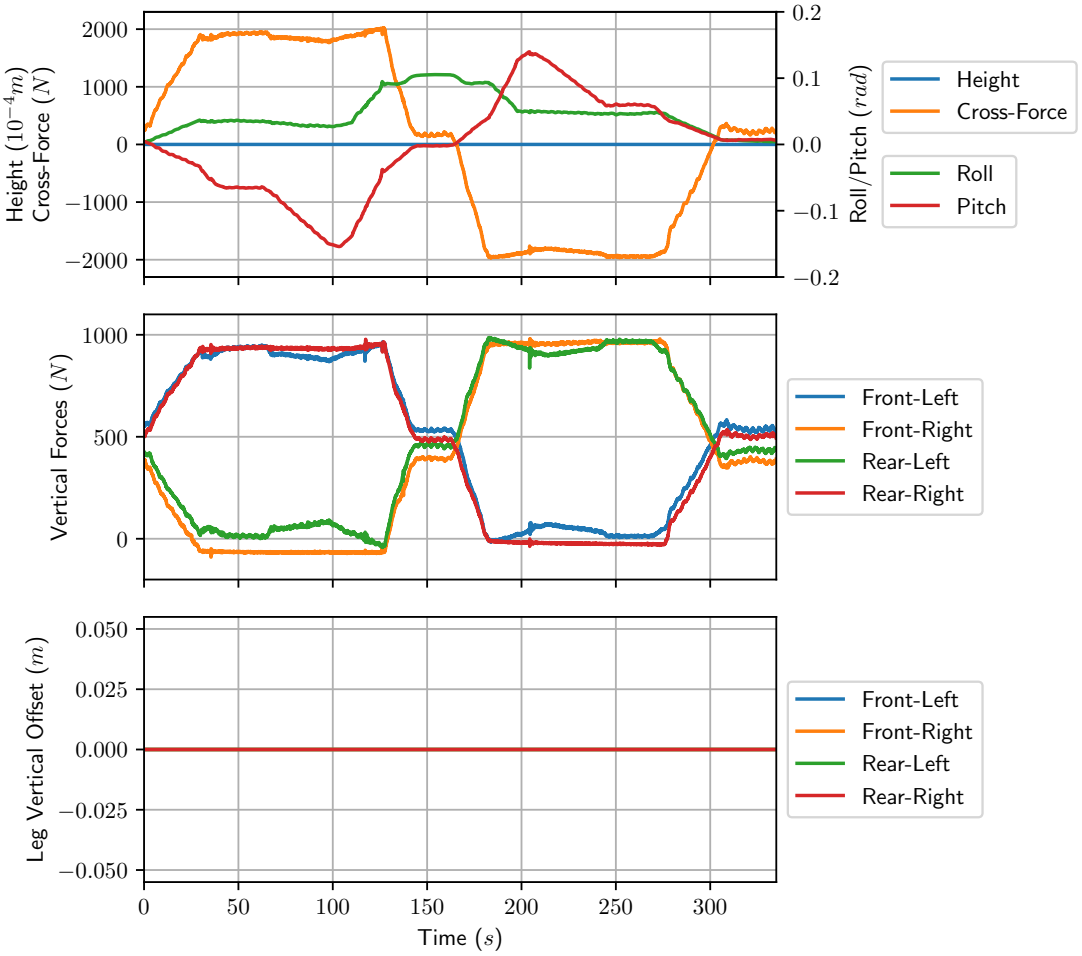


Figure 3.6: Reference experiment while driving over obstacle with GAP deactivated.

The performance of the controller while moving over the uneven terrain is presented here. The evaluation is conducted using three experiments with three different modes of the controller: not active, fully active and only force and height. The results are illustrated in Figures 3.6, 3.7 and 3.8 respectively. Each figure is composed of three subplots. The top plot illustrates the changes in the error in controlled variables as the robot moves over the experimental setup. The middle plot displays the ver-

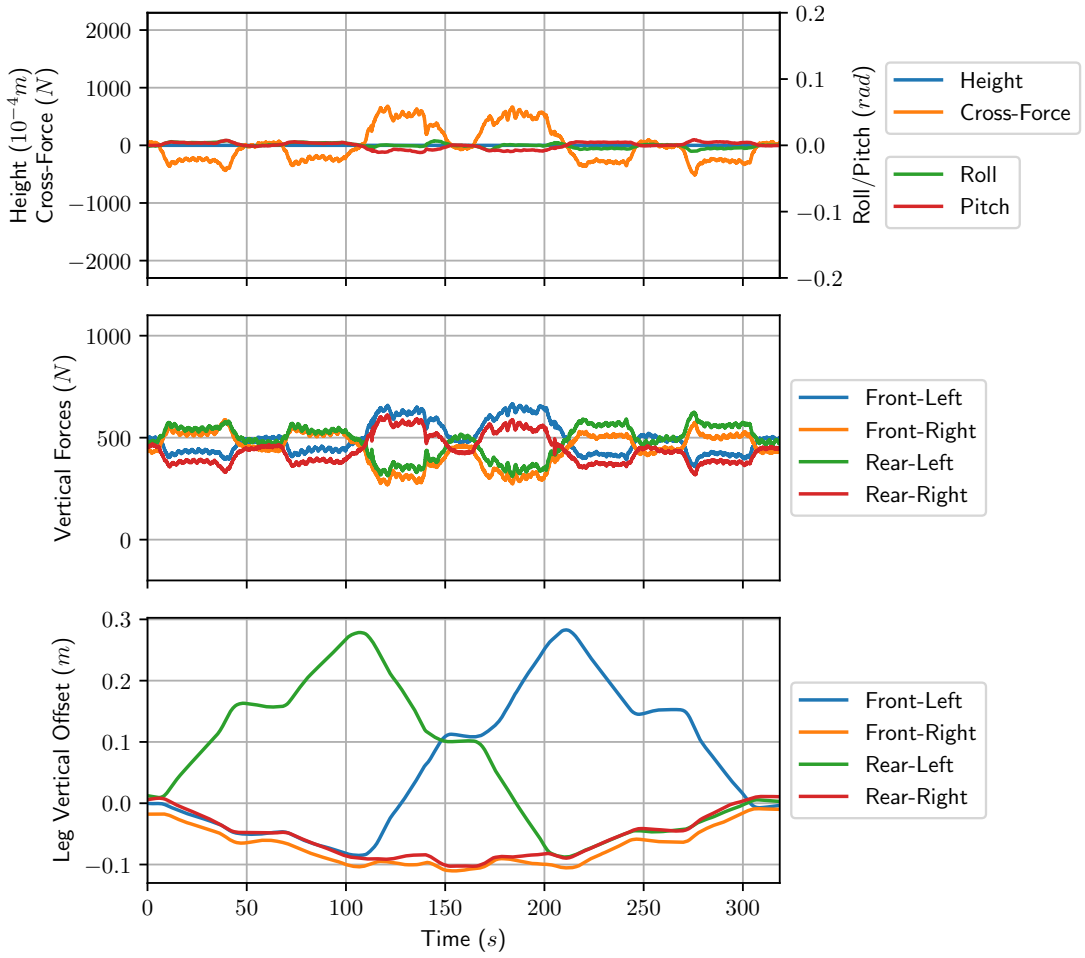


Figure 3.7: Performance of GAP driving over obstacle with all the controlled variables active and reference set to zero.

tical forces acting on each leg. The bottom plot shows the vertical movement of the leg, generated by the controller.

As a point of reference, the evolution of the control factors without the activation of the controller is illustrated in Figure 3.6. Since the controller is deactivated, there is no vertical movement of the legs and no change in the mean height. However, substantial variation exists in the cross-forces, ranging between $-2000\,\mathrm{N}$ and $2000\,\mathrm{N}$. This is also evident in the vertical forces acting on each leg which shows a variation between the range $-50\,\mathrm{N}$ and $1000\,\mathrm{N}$, with a standard deviation of $386.04\,\mathrm{N\,m}$. High variations can also be observed for both roll and pitch angles which shows a maximum deviation of $0.1\,\mathrm{rad}$ and $0.15\,\mathrm{rad}$ respectively. The mean roll and pitch deviations are $0.055\,\mathrm{rad}$ and $0.073\,\mathrm{rad}$ respectively. These observations align with the anticipated outcomes, indicating that the robot's legs experience a loss of contact with the ground and exhibit a propensity to tip over as it navigates obstacles. The substantial load disparity between the legs has the potential to compromise the system's integrity.

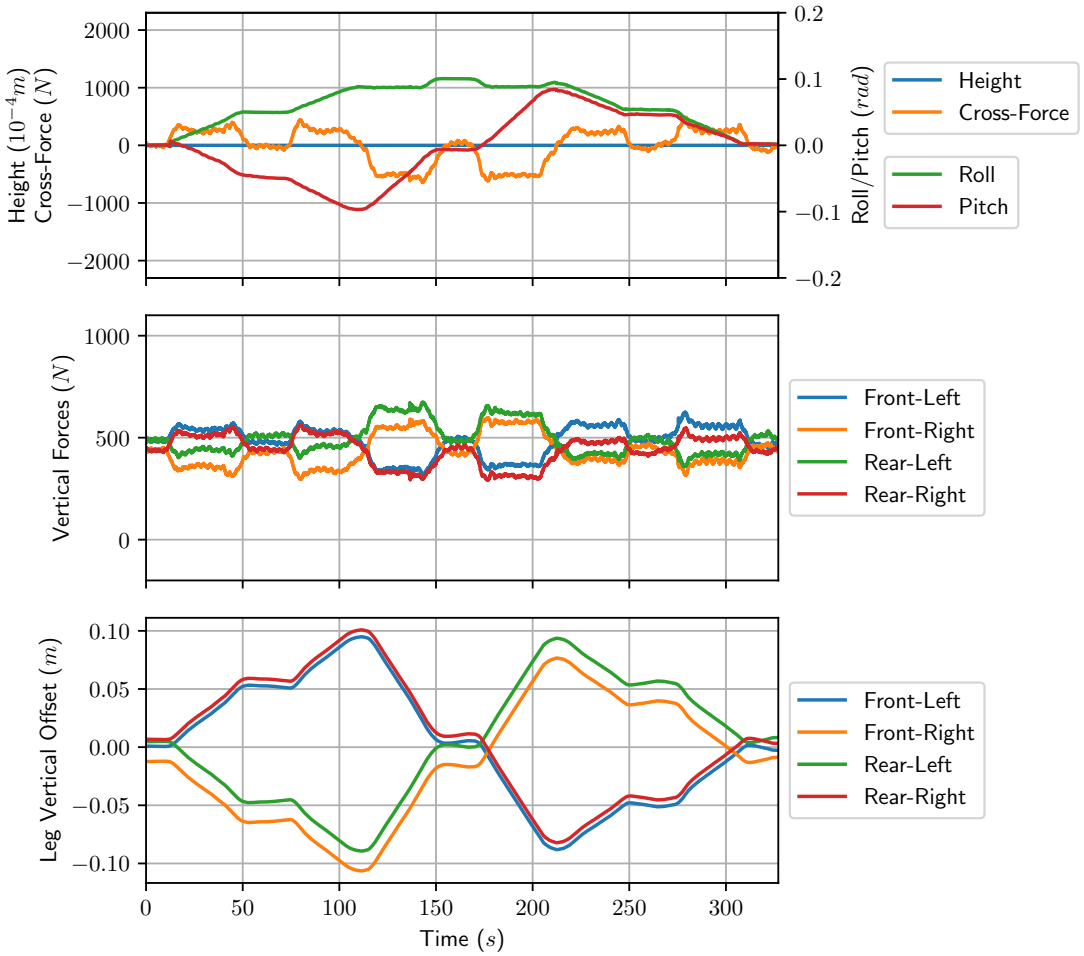


Figure 3.8: Performance of GAP with only mean height and force-error controllers activated.

The results of the same movement over the obstacle with all the controllers activated in shown in Figure 3.7. The vertical movement of the leg shows that the left legs approximately follows the profile of the obstacle. The left legs move vertically between $-0.8\,\mathrm{m}$ and $0.28\,\mathrm{m}$ while the right legs move down to a minimum of $-0.1\,\mathrm{m}$. The results of the controlled variables show that the average height is maintained at $0.0\,\mathrm{m}$ and the cross-force error is within the range between $-517\,\mathrm{N}$ and $680\,\mathrm{N}$, ensuring both good ground clearance and good force distribution within the wheels. This is also evident in the vertical force plot which shows an even distribution and at no point does the wheel lose contact with the ground. The standard deviation of the vertical forces is 79.15 , which is an approximately 80 % reduction compared to the forces without the controller. The roll and pitch deviations are limited to a maximum of $0.011\,\mathrm{rad}$ with quadratic mean of $0.003\,\mathrm{rad}$ and $0.005\,\mathrm{rad}$ respectively. In comparison with the case when the controller is not activated, the quadratic mean values for roll and pitch are within 5 % and 7 % respectively.

The performance of the system when only the height and cross-force controllers are activated is illustrated in Figure 3.8. The errors for height and cross-force are analogous to the experiment when all the controllers are active, yet the maximum errors for roll and pitch are now around $0.1\,\mathrm{rad}$. It is evident that the roll and pitch controllers of GAP are not always necessary and, when activated, limit

the range available for control of the other controlled variables. The vertical leg offset demonstrates a more symmetrical variation in comparison to the scenario where all the controllers are active. The values are within the range of ± 0.1 m, while the performance is analogous to that achieved when all the controllers are active. This renders the controller without roll and pitch capable of navigating more challenging terrain.

The results from further experiments, including step input for different controlled variables, are detailed in Appendix B.

3.6 Success Stories

Extensive testing and utilization of the SherpaTT-MCS and the ground adaptation controller in pure cross-force mode has been carried out in numerous tests[1]. In addition, detailed testing and evaluation of complex operational scenarios in challenging Mars-analogous environments were carried out as part of the projects FT-Utah [2] and FACILITATORS [3]. During one of the field tests, SherpaTT successfully completed a 1.3 km autonomous journey while navigating obstacles, operating using SherpaTT-MCS. The robot and the developed solutions have been incorporated into more than ten research projects, with the contributions from this work having been extensively employed and validated.

3.7 Summary and Discussions

The SherpaTT-MCS is the control software structure for the planetary rover, which is controlled remotely or by high-level controllers. The primary aims of the software are twofold: first, to simplify the control process by providing more straightforward interfaces and commands to high-level controllers; and second, to furnish usable information as feedback. The software incorporates a wide range of features, including driving, steering, stability, posture change and kinematics. Additionally, several assistive functions are implemented to further facilitate control from higher levels.

The most relevant assistive function is the automatic adaptation to changes in terrain, which is achieved by a custom controller, GAP. This controller uses the vertical motion of the legs to simultaneously control multiple states of the robot, with objectives including maintaining ground contact, evenly distributing loads on the legs, and maintaining a desired attitude. The controller's design involves mapping the vertical motion of the legs to the desired robot postures including body height, pitch, roll and load distribution. The controller has been implemented and tested in a labor test environment through a series of experiments, which demonstrate its effectiveness across varying terrains. The results demonstrate that the GAP controller enhances various parameters of locomotion in comparison with the absence of controller activation. The findings of the study demonstrated that a reduction of 80 % in the standard deviation of the vertical forces was achieved, alongside an approximate 95 % and 93 % reduction in roll and pitch errors, respectively.

This chapter describes two main contributions to the dissertation: (i) development and implementation of SherpaTT-MCS, and (ii) design and testing of a custom GAP controller. The robot and the developed SherpaTT-MCS have been successfully utilized in more than ten projects and at least three

[1] Video footage of the robot's motion while incorporating the controller in an outdoor uneven terrain is available at the following link: https://drive.google.com/file/d/1Eiz2ZpOjKlqGjs4WJlHkRop_0rU3j4iW/view?usp=sharing

[2] https://robotik.dfki-bremen.de/en/research/projects/ft-utah

[3] https://robotik.dfki-bremen.de/en/research/projects/facilitators-og6

field tests in Mars-analogous environments. This demonstrated the effectiveness and robustness of the solutions developed here, especially the force-leveling component of the GAP. This corresponds to the thesis objectives O-2a and O-3a respectively.

Motion Control System for a Walking Excavator Robot

This chapter discusses the development of the software control framework, ARTER-MCS, for controlling a walking excavator robot in remote control as well as autonomous scenarios. Partial results of the presented work have been published in

(i) Babu, A., P. Willenbrock, J. Tiemann, F. Bernhard, and D. Kuehn (2024). "ARTER: a walking excavator robot". In: *Biologically Inspired Series-Parallel Hybrid Robots*. Ed. by S. Kumar, A. Mueller, and F. Kirchner. 1. Auflage. Vol. 514. Elsevier Science, pp. 235–261. ISBN: 978-0-323-88482-2. URL: https://shop.elsevier.com/books/biologically-inspired-series-parallel-hybrid-robots/kumar/978-0-323-88482-2

(ii) Woock, P. and A. Babu (2022). "Autonome Robotersysteme in der Altlastensanierung". *Handbuch Altlastensanierung und Flächenmanagement*. Handbuch Altlastensanierung und Flächenmanagement 93. Aktualisierung, 3. Aufl.5111. Ed. by V. Franzius, M. Altenbockum, and T. Gerhold

(iii) Babu, A., L. C. Danter, P. Willenbrock, S. Natarajan, D. Kuehn, and F. Kirchner (2022). *at - Automatisierungstechnik* 70:10, pp. 876–887. DOI: doi:10.1515/auto-2022-0056. URL: https://doi.org/10.1515/auto-2022-0056

(iv) Babu, A., K. Y. Yurtdas, C. E. S. Koch, and M. Yüksel (2019). "Trajectory Following using Nonlinear Model Predictive Control and 3D Point-Cloud-based Localization for Autonomous Driving". In: *2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic*. IEEE

The chapter is structured as follows: The ARTER system is described in Section 4.1 and Section 4.2, followed by the software structure of ARTER-MCS in Section 4.3. Next, examples of inverse and forward solutions for the parallel kinematic mechanisms are developed in Section 4.4. The following Section 4.5 details the trajectory following using Nonlinear Model Predictive Control (NMPC), including the problem formulation and evaluation of performance in simulation. Finally, Section 4.6.2 describes the evaluation of the control software in the soil-sampling scenario. The chapter is summarized at the end.

4.1 Walking Excavator

In contemporary industrial contexts, heavy-duty machinery plays a pivotal role in a wide range of sectors, including construction, agriculture, mining and waste management. These machines are designed to perform a diverse array of tasks with high efficiency and precision. A portion of these tasks have already been automated. However, a significant proportion of tasks remain repetitive and, in cer-

tain instances, hazardous. The integration of remote-control functionality and supplementary assistance systems has the potential to enhance both safety and productivity. The transfer of technologies from the robotics domain, encompassing sensing, control and planning, can facilitate this development.

The utilization of teleoperation, advanced driver assistance functions, and limited autonomy features is already prevalent in these vehicles to a certain extent. As research in robotics and artificial intelligence continues to advance at a rapid pace, the scope for more immersive teleoperation and higher levels of autonomy is expanding, which are necessary for operation in challenging environments. A significant environmental challenge confronting heavy-duty vehicles and mobile robots is traversing unstructured terrains, characterized by obstacles and high slopes with varied substrates.

To address these challenges, walking excavators (also known as "spider excavators") have been developed. These specialized machines possess the capability to traverse unstructured terrains, thus enhancing their operational effectiveness in challenging environments. In comparison to conventional excavators, walking excavators exhibit superior mobility. Their capacity to traverse wide ditches, ascend onto platforms, and operate in mountainous terrain, building infrastructure in otherwise inaccessible areas, exemplifies their utility. The design of walking excavators, which involves wheel-on-leg configuration, enables precise adjustment of the position of each individual wheel through movement of the legs, thereby facilitating navigation on unstructured terrain.

Adaptive suspension systems of this nature are frequently utilized as research platforms on uneven terrain. The application of manipulators for locomotion in conjunction with wheels is a rarity in the field of robotics research. Jud et al. 2021 details the design and development of the walking excavator robot Hydraulic Excavator for an Autonomous Purpose (HEAP), which has been retrofitted with custom-developed hydraulic valves to control the joints. The HEAP robot has demonstrated an ability to adapt to the terrain through the utilization of these force-control valves.

Figure 4.1: ARTER is a walking excavator robot being developed by DFKI.

The ARTER system, as illustrated in Figure 4.1, is based on the Menzi-Muck M545 walking excavator, which is well-suited for off-road conditions due to its exceptional mobility. The robot is being

developed by DFKI as part of the ROBDEKON (Robotic Systems for Decontamination in Hostile Environments) competence center, which is responsible for the research of autonomous or semi-autonomous robotic systems, considering both nuclear and waste site decontamination scenarios (Petereit et al. 2019).

The base vehicle has been fitted with numerous sensors, hydraulic valves, computation hardware, etc., making it remotely controllable and facilitates autonomous functionalities. The robot's ability to utilize the manipulator for locomotion confers a remarkable advantage, manifesting in three distinct locomotion modes. (i) *Driving mode* where only the wheels are in contact with the ground; (ii) *Climbing mode* where all the wheels are in contact with the ground and the manipulator is used to support driving on terrain with steep or slippery slopes; (iii) *Stepping mode* where the manipulator is used to lift the wheels from the ground, typically two wheels simultaneously, and step over obstacles.

The robot's capacity for such demanding functionalities requires the development of sophisticated control software in order to make the most of all its features. The design of the MCS, designated as the ARTER-MCS, is delineated in this chapter. The primary objective of the design is to empower the robot with remote control capabilities and autonomy modes, along with the flexibility to seamlessly transition between these modes when necessary. The ARTER-MCS design also incorporates all the necessary modules to facilitate the functioning of the different autonomy and assistance functions. The objective O-3b is addressed here.

4.2 System Description - ARTER

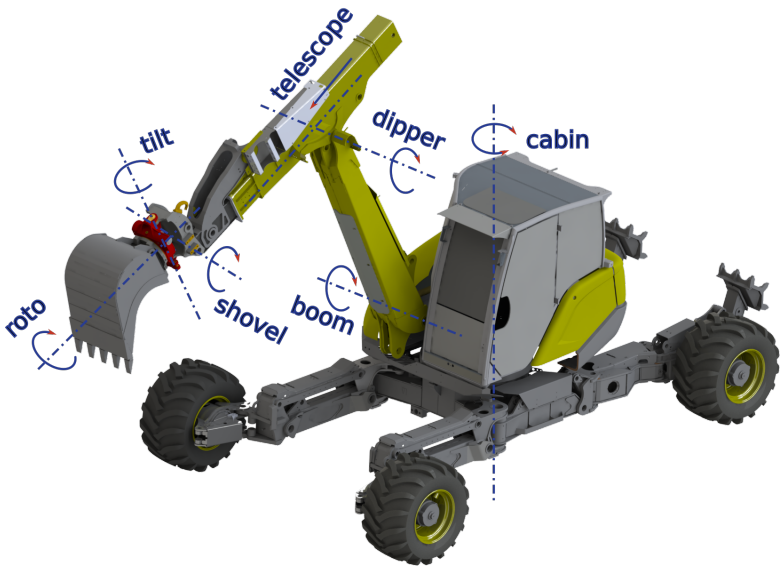


Figure 4.2: Kinematics of the ARTER manipulator.

The ARTER system, with a mass of approximately 13 tons, exhibits 27 degrees of freedom. The chassis constitutes the foundation upon which five kinematically distinct structures are affixed: The first is the manipulator, followed by two front legs and two rear legs.

The manipulator is composed of seven joints: Cabin, Boom, Dipper, Telescope, Shovel, Tilt and Roto, as illustrated in the Figure 4.2. The driver cabin is connected to the chassis through the Cabin

joint, which possesses infinite rotational capacity. This joint is the primary component responsible for enabling the end-effector to move laterally. The joints, designated as the Boom, Dipper and Telescope, facilitate the vertical and longitudinal movement of the end-effector. The joints, namely the Shovel, Tilt and Roto joints, are particularly instrumental in enabling the end-effector's angular movement. The Rototilt attachment, which is an additional component, provides the Tilt and Roto joints.

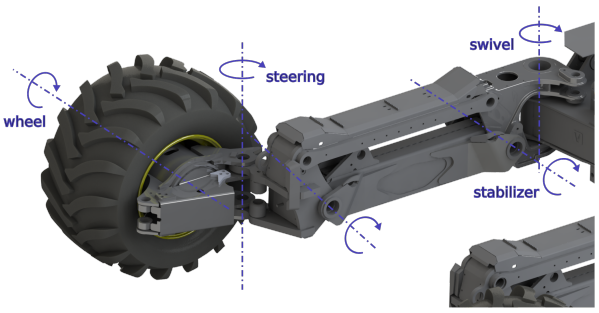


Figure 4.3: Kinematics of the ARTER front legs.

As illustrated in Figure 4.3, the front legs of the vehicle are composed of four joints: the Swivel joint, the Stabilizer joint, the Steering joint and the Wheel joint. The function of the Swivel joint is to facilitate lateral movement of the wheel, while the stabilizer joint enables vertical movement. The parallelogram kinematics of the Stabilizer joint ensures that the wheels do not pitch. The Steering joint, on the other hand, is responsible for orienting the wheel in the desired direction for vehicle steering. The Wheel joint facilitates rotation, thereby imparting the longitudinal velocity to the vehicle.

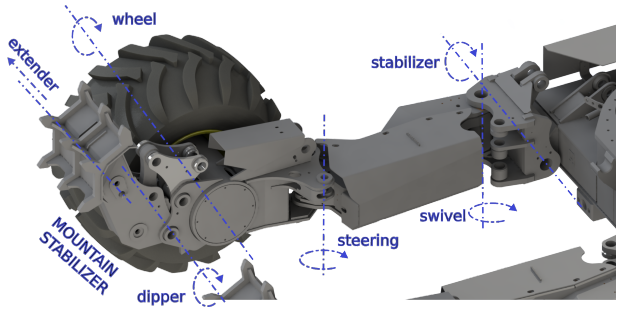


Figure 4.4: Kinematics of the ARTER rear legs.

The rear legs exhibit three primary distinctions in comparison to the front legs, as illustrated in Figure 4.4. The absence of a parallelogram structure in the Stabilizer joint allows for the possibility of pitch angles in the wheels. The kinematic tree places the Stabilizer joint first followed by the Swivel joint. An additional structure, designated as the Mountain Stabilizer is connected to the Steering link. This structure is employed to ensure stability on high slopes by providing a secure anchorage to the ground. There are two joints in this structure: Claw-Dipper and Claw-Extender.

4.3 Control Software Structure

The ARTER-MCS is the control software responsible for regulating the robot's movement. It is designed to operate the robot in teleoperation or autonomous modes, as reflected in its design. The

structure is divided into three layers: The low-level, mid-level and high-level layers. The mid-level and the high-level are described in the following Sections 4.3.1 and 4.3.2 respectively. The design of the low-level, however, is not a direct contribution of this work and is hence described in Appendix C.

4.3.1 Mid-level Control

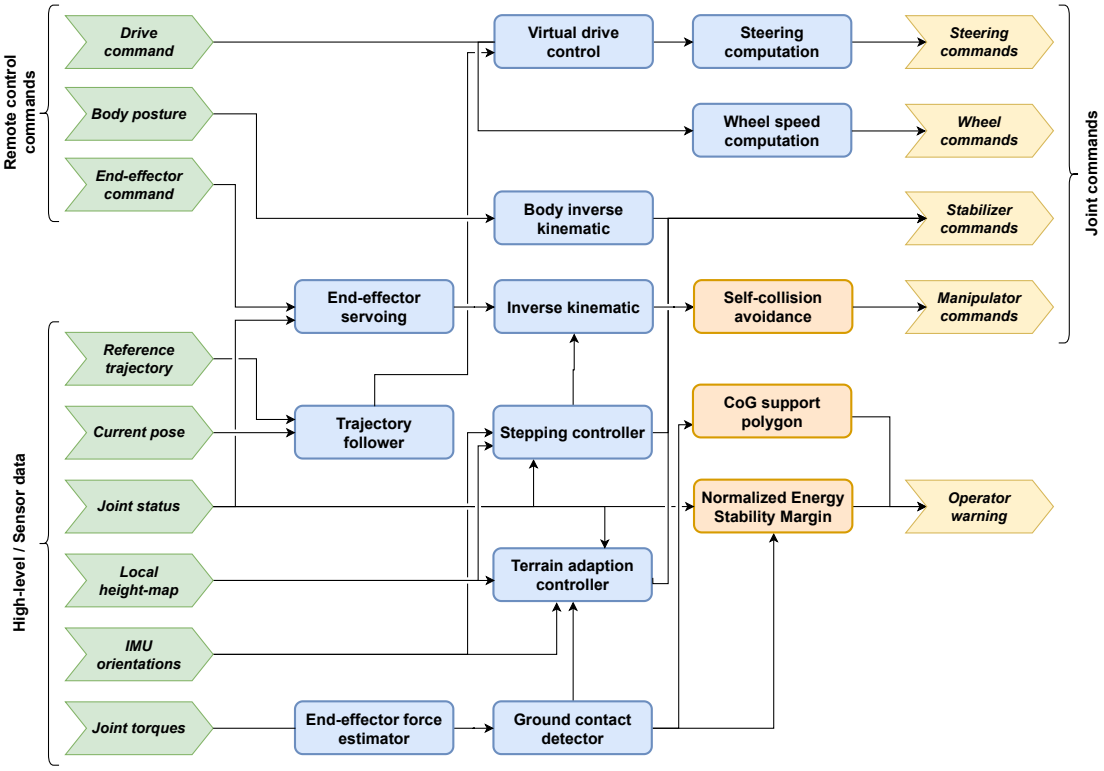


Figure 4.5: Mid-level control structure of ARTER-MCS.

The mid-level control layer of ARTER-MCS provides controllers for use with remote control, safety controllers and monitors, and interfacing between high-level and low-level controllers. The general structure is illustrated in Figure 4.5.

In the course of remote control operation, commands such as the drive command, body posture and end-effector commands are processed and sent to the low-level joint controllers. The drive command encompasses the steering and wheel velocity commands. This command is then transmitted to the virtual drive control module, which in turn maps the drive commands to a bicycle kinematic model. The steering computation module is responsible for converting the reference steering angle of the model to the command reference joint angles of the robot steering joints. The wheel speed computation module, in turn, computes the wheel speeds based on the virtual drive speed.

The body posture command provides the desired body posture velocities for the robot, including height, roll, pitch and other related parameters. The body inverse kinematic module, in conjunction with a transform function, then converts this command to corresponding velocities for the robot's stabilizer joints. This facilitates the operator's control over leg heights.

The end-effector twist command for the manipulator is routed through the servoing module, as outlined in Coleman et al. 2014, which computes the desired end-effector position. This position is then utilized by the inverse kinematic module to calculate the joint angles. The joint angles are subsequently transmitted to the low-level controllers, provided that the self-collision checking module does not detect any potential self-collisions.

A set of controllers has been developed to facilitate the functioning of different high-level tasks. The trajectory follower, which accepts the reference trajectory and the robot's current pose as inputs, will command the virtual drive to ensure that the robot base follows the desired path. Two controllers have been implemented: one based on model predictive control (Babu et al. 2019) and a second based on pure-pursuit (Jelavic et al. 2021b).

The terrain adaptation controller autonomously adjusts the active suspension system to uneven terrain by moving the legs up and down. The controller's multiple goals include maintaining stability, avoiding ground collisions, and keeping the wheels in contact with the ground. The controller's functionality is further delineated in Section 3.5 and in Cordes et al. 2017, which emphasizes its ability to maintain both ground contact and the robot's physical orientation. A more sophisticated controller, leveraging reinforcement learning, is developed in Chapter 5 to automatically adapt to the terrain based on elevation map, contact estimation and orientation information. The detection of ground contact is facilitated by the estimation of end-effector forces, which, in turn, is contingent upon the measurement of joint torques. It is important to note that the end-effector forces are only partially observable, as not all joint torques are measured.

The stepping controller is an assistive autonomous function responsible for generating the sequence of motions necessary to traverse an obstacle, gap, or step. The connections are analogous to the terrain adaptation controller, with the additional inclusion of commands to the manipulator to assist in locomotion. A solution employing hierarchical reinforcement learning and action masking is developed in Chapter 6.

A primary factor that must be considered when controlling such a robotic platform with adaptive suspension is the tip-over stability. In the field of robotics, numerous stability margins have been developed to address this issue. A comparative analysis of these margins, as detailed in the study by Garcia et al. 2002, has identified the Normalized Energy Stability Margin (NESM) (Hirose et al. 1998) as a particularly effective solution for robots operating on uneven terrain. The NESM is employed for two primary purposes: first, to facilitate the learning process of the terrain adaptation controller, and second, to serve as a warning system for the operator.

4.3.2 High-level Control

The high-level ARTER-MCS functions as the interface between the operator or mission control and the mid-level or low-level controllers. The high-level controls for ARTER are depicted in Figure 4.6.

The high-level system receives input from both the mission control and sensor data. The foot-print changer module, which takes the desired footprint as input, controls the manipulator, and the swivel and stabilizer joints in the legs. The foot-print changer module executes a set of predefined sequences to manipulate the manipulator at the front to lift the front wheels and then move the swivel joints to the desired angles. This sequence is then repeated for the rear side.

The manipulator planner, which is based on the method outlined in Coleman et al. 2014, takes the desired manipulator pose as the command and also the local OctoMap (Hornung et al. 2013), which is a

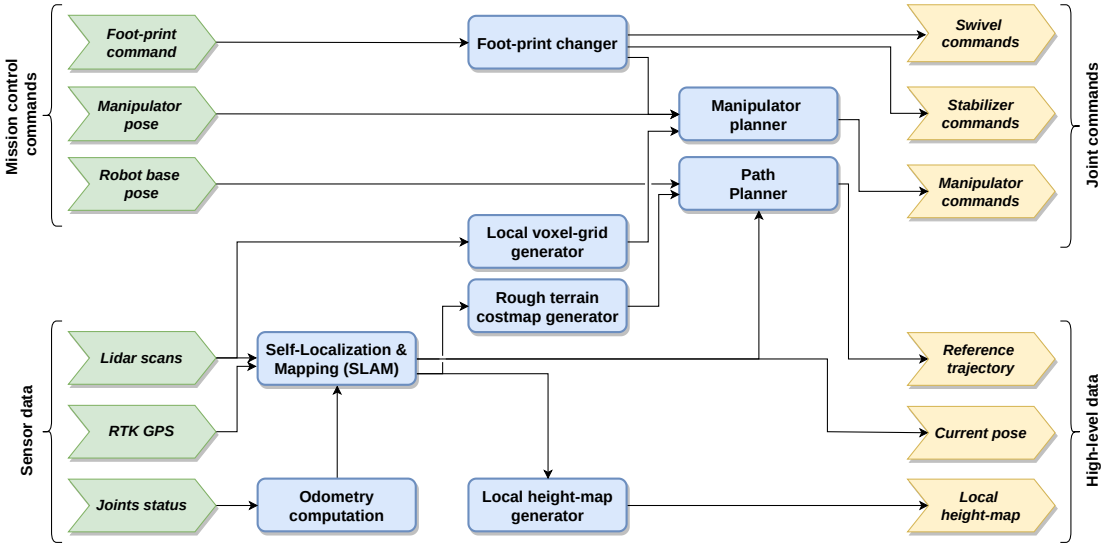


Figure 4.6: High-level control structure of ARTER-MCS.

probabilistic 3D occupancy grid map. The generation of the OctoMap is facilitated by the utilization of scans obtained from three different LIDARs. The planned motion is executed by transmitting commands to the manipulator.

The path planner facilitates the planning of the robot base's trajectory from a 2D pose (x, y and yaw) in the map to another. The 3D environment is currently represented as a grid map (Fankhauser and Hutter 2016) and converted to the corresponding 2D costmap by considering terrain features such as obstacles, roughness and slope. The subsequent planning of the trajectory is executed by the Smac planner, as devised in Macenski et al. 2020. This planner utilizes the Hybrid-A* algorithm. The planner's output is the trajectory, which is subsequently transmitted to the mid-level controller.

The SLAM module is responsible for generating a 3D map of the environment and estimating the robot's pose within the map. This function is enabled by the utilization of the Slam3d (DFKI Slam3D) package. The module is primarily based on LIDAR data and computed odometry of the vehicle. Additionally, inertial navigation system with Real-Time Kinematic (RTK)-GPS data is utilized serves as an input source. The localization and map of the environment from SLAM are then used to generate a local height map, which is in turn utilized by the mid-level controllers.

4.4 Kinematic Modeling

The ARTER is a series-parallel hybrid system, in which the actuator and the joint do not share the same axis for many joints. The movement of each parallel kinematic joint is determined by the movement of the actuator through a complex set of mechanical connections. The design of such a configuration arises from the need to increase the load capacity, improve the stiffness of the joints and improve the dynamic performance. In comparison to hydraulic motors, hydraulic linear cylinders are advantageous due to their higher control accuracy. This has led to the adoption of hydraulic cylinders with parallel kinematic for most joints, which are then linked in series.

In order to ensure effective control of the joint, it is essential to model the kinematic properties of the parallel mechanisms for both forward and inverse kinematic. This corresponds to the relationship between the joint states with the actuator states including position, velocity and forces. Pressure sensors have been attached to the cylindrical actuators, which can then be used to compute the effective forces or torques at the joint level. The combination of these measurements with the kinematic data of the manipulator facilitates the estimation of virtual forces acting at the end-effector. The higher levels of control are abstracted from the details of the control at the joint level, and are only aware of the series kinematic aspect of the legs and manipulators.

The estimation of joint states is facilitated by the utilization of linear or rotary encoders, which are mounted on the actuator, the joint, or an intermediate joint within the parallel mechanism. The determination of the location is influenced by mechanical constraints and practical limitations. This results in the establishment of three distinct coordinate-spaces (detailed in Appendix C) for each joint: actuator, joint and sensor spaces. The conversion between these separate spaces is necessary for control, as the reference (joint space), measured (sensor space) and commanded (actuator spaces) values should be in the same coordinate space. This conversion can be achieved by employing both the inverse and forward models of the parallel kinematics, which vary according to the specific joint in question. The subsequent sections detail examples of this computation for the Shovel and Dipper joints.

4.4.1 Shovel Joint Kinematics

The parallel kinematic mechanism for the Shovel joint is shown in Figure 4.7a. The pivots in the mechanism are denoted by A, B, C, D and E, where A, D and E are the fixed pivots, and B and C are the moving pivots. The constant distance between the pivots in the mechanism are denoted as a, b, c, d and e representing lengths of $\overline{AE}$, $\overline{BC}$, $\overline{CD}$, $\overline{DE}$ and $\overline{BE}$ respectively. The variable distances of segments $\overline{BD}$ and $\overline{CE}$ are represented by l_1 and l_2 respectively. The length of $\overline{AB}$ is the actuator length (s) and the output joint angle (θ) is the angle given by $\angle CDE$. In case of the shovel, the sensor is mounted on the pivot E which can directly provide the value of φ_1 and φ_2 which are the angles $\angle AEB$ and $\angle BED$ respectively.

Actuator to Joint Space Mapping

The following develops the joint angle θ in terms of s. Adding the angles at pivot E gives the following relationship

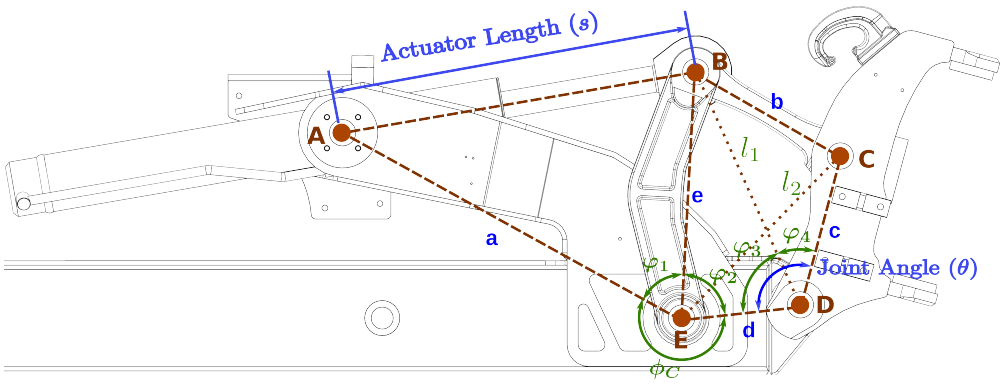
$$\varphi_1 + \varphi_2 + \phi_C = 2\pi, \tag{4.1}$$

where ϕ_C is a constant formed by the reflex angle of $\angle AED$. The angles $\angle EDB$ and $\angle BDC$, represented by φ_3 and φ_4 respectively, can be added to get the joint angle

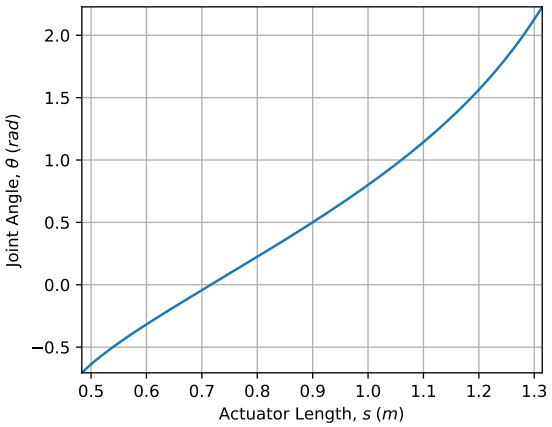
$$\varphi_3 + \varphi_4 = \theta. \tag{4.2}$$

The mechanism can be separated into two well known parallel mechanism: Inverted Slider Crank Mechanism (ISCM) and Four-Bar Mechanism (FBM). ABE forms the ISCM with s as the input linear distance and φ_1 as the output angle. The FBM is formed by the pivots $BCDE$ with φ_2 as the input angle and θ as the output angle. $\overline{BE}$ is the common segment in both the mechanism. Using the law of cosines for the $\triangle ABE$ gives the relationship

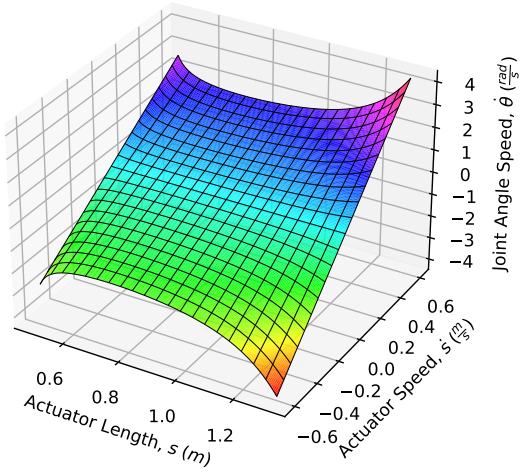
$$s^2 = a^2 + e^2 - 2ae\cos(\varphi_1), \tag{4.3}$$



(a) Shovel joint kinematic design.



(b) Shovel joint position kinematics.



(c) Shovel joint velocity kinematics.

Figure 4.7: Kinematics of the Shovel joint along with solutions for the parallel kinematics (Image source: Babu et al. 2024).

which can be combined with Equation (4.1) and rewritten as

$$\varphi_2(s) = 2\pi - \phi_C - \arccos\left(\frac{a^2 + e^2 - s^2}{2ae}\right), \tag{4.4}$$

giving the relationship between linear position of ISCM and the input angle of the FBM. The kinematics of the FBM are computed below. Using law of cosines for triangles $\triangle BED$ for the angles φ_2 and φ_3 gives the equations

$$l_1^2 = d^2 + e^2 - 2de\cos(\varphi_2) \tag{4.5}$$

and

$$e^2 = l_1^2 + d^2 - 2l_1 d\cos(\varphi_3) \tag{4.6}$$

respectively. Writing Equation (4.5) to obtain the value of l_2 in terms of φ_2 gives

$$l_1(\varphi_2) = \sqrt{d^2 + e^2 - 2de\cos(\varphi_2)}. \tag{4.7}$$

Equation (4.6) can be rewritten to obtain φ_3 in terms of l_1 as

$$\varphi_3(l_1) = \arccos\left(\frac{l_1^2 + d^2 - e^2}{2l_1 d}\right). \tag{4.8}$$

Similarly, $\triangle BCD$ with angle φ_4 produces

$$b^2 = l_1^2 + c^2 - 2l_1 c\cos(\varphi_4), \tag{4.9}$$

which can be rewritten as

$$\varphi_4(l_1) = \arccos\left(\frac{l_1^2 + c^2 - b^2}{2l_1 c}\right). \tag{4.10}$$

Substituting Equations (4.8) and (4.10) in Equation (4.2) gives the output angle in terms of l_1 as

$$\theta(l_1) = \arccos\left(\frac{l_1^2 + d^2 - e^2}{2l_1 d}\right) + \arccos\left(\frac{l_1^2 + c^2 - b^2}{2l_1 c}\right). \tag{4.11}$$

The value of θ with respect to s can be obtained by recursively substituting the terms in Equations (4.4) and (4.7). Using a similar process, s can be computed in terms of θ.

The following computes the joint angular velocity $\dot{\theta}$ in terms of the actuator velocity $\dot{s}$ and position s. Using Equations (4.1) and (4.3) along with their time derivatives can be combined to form

$$\dot{\varphi_2}(s, \dot{s}, \varphi_2) = -\frac{s}{ae\sin(2\pi - \phi_C - \varphi_2)}\dot{s}, \tag{4.12}$$

where $\varphi_2(s)$ can be obtained from Equation (4.4).

Time derivative of Equation (4.5) is used to obtain $\dot{l_1}$ as

$$\dot{l_1}(\varphi_2, \dot{\varphi_2}) = \left(\frac{de\sin(\varphi_2)}{l_1}\right)\dot{\varphi_2}, \tag{4.13}$$

where $\varphi_2(s)$ can be obtained from Equation (4.4).

Similarly, $\dot{\varphi_3}$ and $\dot{\varphi_4}$ can be obtained from time derivates of Equations (4.6) and (4.9) as

$$\dot{\varphi_3}\left(l_1, \dot{l_1}, \varphi_3\right) = \left(\frac{d\cos(\varphi_3) - l_1}{l_1 d\sin(\varphi_3)}\right)\dot{l_1} \tag{4.14}$$

and

$$\dot{\varphi_4}\left(l_1, \dot{l_1}, \varphi_4\right) = \left(\frac{c\cos(\varphi_4) - l_1}{l_1 c\sin(\varphi_4)}\right)\dot{l_1} \tag{4.15}$$

respectively, where Equation (4.8) gives the value of φ_3 and Equation (4.10) that of φ_4.

Substituting Equations (4.14) and (4.15) in time derivative of Equation (4.2) and rearranging gives

$$\dot{\theta}\left(l_1, \dot{l_1}, \varphi_3, \varphi_4\right) = \left(\frac{d\cos(\varphi_3) - l_1}{l_1 d\sin(\varphi_3)} + \frac{c\cos(\varphi_4) - l_1}{l_1 c\sin(\varphi_4)}\right)\dot{l_1}, \tag{4.16}$$

which can be computed by recursively substituting values from Equations (4.12) and (4.13).

Joint to Actuator Space Mapping

Similar methods can be used to derive the equations for s in terms of θ which is given by recursively substituting

$$\varphi_1(l_2) = 2\pi - \phi_C - \arccos\left(\frac{c^2 - l_2^2 - d^2}{2ld}\right) - \arccos\left(\frac{b^2 - l_2^2 - e^2}{2le}\right) \tag{4.17}$$

and

$$l_2(\theta) = \sqrt{c^2 + d^2 - 2cd\cos(\theta)} \tag{4.18}$$

into

$$s(\varphi_1) = \sqrt{a^2 + e^2 - 2ae\cos(\varphi_1)}. \tag{4.19}$$

The actuator velocity $\dot{s}$ in terms of $\dot{\theta}$ can be written as

$$\dot{s}(\varphi_1) = \left(\frac{ae\sin(\varphi_1)}{s}\right)\dot{\varphi_1}, \tag{4.20}$$

where

$$\dot{s}(l_2, \dot{l_2}, \varphi_{21}, \varphi_{22}) = \left(\frac{l_2 - e\cos(\varphi_{21})}{l_2 e\cos\varphi_{21}} + \frac{l_2 - d\cos(\varphi_{22})}{l_2 d\cos\varphi_{22}}\right)\dot{l_2} \tag{4.21}$$

and

$$\dot{l_2}\left(l_2, \theta, \dot{\theta}\right) = \left(\frac{cd\sin(\theta)}{l_2}\right)\dot{\theta}. \tag{4.22}$$

The value of l_2 is given by Equation (4.18). The angles φ_{21} ($\angle BEC$) and φ_{22} ($\angle CED$) are given by

$$\varphi_{21}(l_2) = \arccos\left(\frac{l_2^2 + e^2 - b^2}{2l_2 e}\right) \tag{4.23}$$

and

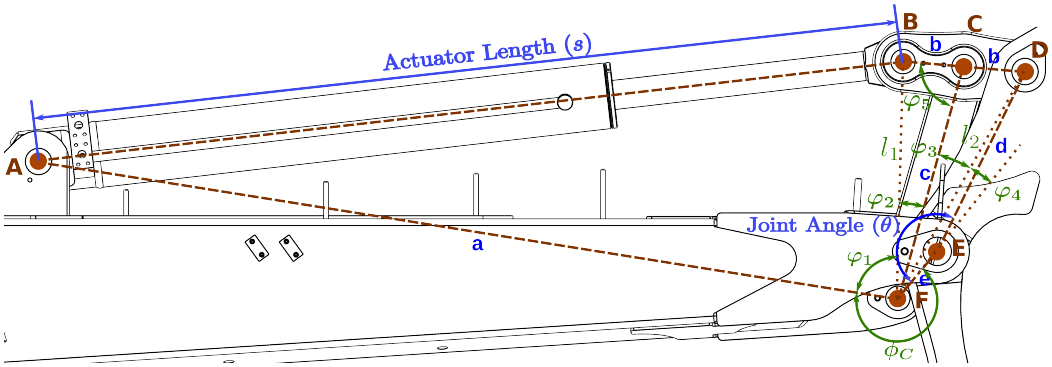
$$\varphi_{22}(l_2) = \arccos\left(\frac{l_2^2 + d^2 - b^2}{2l_2 d}\right) \tag{4.24}$$

respectively.

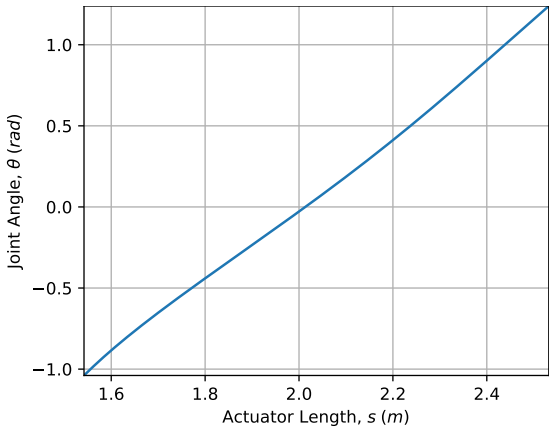
The kinematic equations for position (Equation (4.11)) and velocity (Equation (4.16)) are depicted in Figures 4.7b and 4.7c respectively. The plots show the non-linearities introduced due to the parallel kinematic mechanisms.

4.4.2 Dipper Joint Kinematics

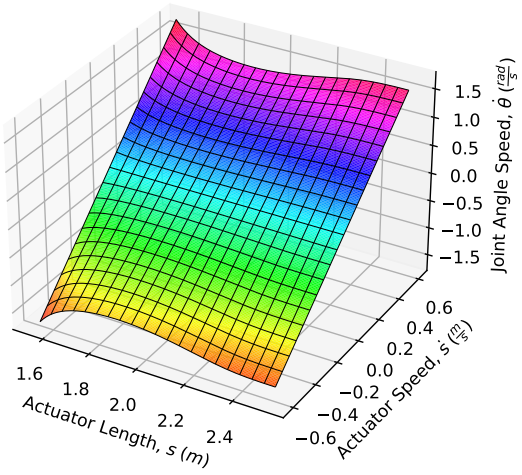
The parallel kinematics of the Dipper joint, which is shown in Figure 4.8a, cannot be separated to any known mechanisms as is the case with Shovel joint. The mechanism consists of the pivots A, B, C, D, E and F of which B, C and D are moving pivots. The input is the actuator length s, which is the length of the segment $\overline{AB}$, and the output is the joint angle θ, formed by $\angle DEF$. The linear encoder is attached directly to the actuator and hence measures s directly. a, b, c, d and e are the constant lengths of the segments $\overline{AF}$, $\overline{BC}$, $\overline{CF}$, $\overline{DE}$ and $\overline{EF}$ respectively. The segment $\overline{CD}$ also has the constant length b. The lengths of segments $\overline{BF}$ and $\overline{DF}$ are varying and are denoted by l_1 and l_2



(a) Dipper joint kinematic design.



(b) Dipper joint position kinematics.



(c) Dipper joint velocity kinematics.

Figure 4.8: Kinematics of the Dipper joint along with solutions for the parallel kinematics (Image source: Babu et al. 2024).

respectively. The angles $\angle AFB$, $\angle BFC$, $\angle CFD$, $\angle DFE$ and $\angle BCF$ are denoted by φ_1, φ_2, φ_3, φ_4 and φ_5 respectively. The constant reflex angle formed by $\angle EFA$ is denoted as ϕ_C.

Joint to Actuator Space Mapping

The relationship of s in terms of θ is derived below. Adding all the angles at pivot F gives

$$\varphi_1 + \varphi_2 + \varphi_3 + \varphi_4 + \phi_C = 2\pi. \tag{4.25}$$

Using the law of cosines for the triangles corresponding to the angles θ, $\angle DCF$, φ_5, φ_4, φ_3, φ_2 and φ_1 gives the equations

$$l_2^2 = e^2 + d^2 - 2ed\cos(\theta), \tag{4.26}$$

$$l_2^2 = b^2 + c^2 - 2bc\cos(\pi - \varphi_5), \tag{4.27}$$

$$l_1^2 = b^2 + c^2 - 2bc\cos(\varphi_5), \tag{4.28}$$

$$d^2 = l_2^2 + e^2 - 2l_2e\cos(\varphi_4), \tag{4.29}$$

$$b^2 = l_2^2 + c^2 - 2l_2 c\cos(\varphi_3), \tag{4.30}$$

$$b_2^2 = l_1^2 + c^2 - 2l_1 c\cos(\varphi_2), \tag{4.31}$$

and

$$s^2 = a^2 + l_1^2 - 2al_1\cos(\varphi_1). \tag{4.32}$$

Equation (4.26) can be rewritten as

$$l_2(\theta) = \sqrt{e^2 + d^2 - 2ed\cos(\theta)}. \tag{4.33}$$

Equations (4.27) and (4.28) can be combined to eliminate φ_5 and write l_1 in terms of l_2 which is given by

$$l_1(l_2) = \sqrt{2b^2 + 2c^2 - l_2^2}. \tag{4.34}$$

φ_4 and φ_3 can be written in terms of l_2 from Equations (4.29) and (4.30) giving

$$\varphi_4(l_2) = \arccos\left(\frac{l_2^2 + e^2 - d^2}{2l_2 e}\right) \tag{4.35}$$

and

$$\varphi_3(l_2) = \arccos\left(\frac{l_2^2 + c^2 - b^2}{2l_2 c}\right). \tag{4.36}$$

Similarly, φ_2 can be written in terms of l_1 as

$$\varphi_2(l_1) = \arccos\left(\frac{l_1^2 + c^2 - b^2}{2l_1 c}\right). \tag{4.37}$$

Substituting Equations (4.35) to (4.37) in Equation (4.37) and rearranging gives φ_1 as

$$\begin{aligned}\varphi_1(l_1, l_2) = 2\pi - \phi_C - \arccos\left(\frac{l_1^2 + c^2 - b^2}{2l_1 c}\right) \\ - \arccos\left(\frac{l_2^2 + c^2 - b^2}{2l_2 c}\right) - \arccos\left(\frac{l_2^2 + e^2 - d^2}{2l_2 e}\right).\end{aligned} \tag{4.38}$$

From Equation (4.32), s can be written in terms of φ_1 and l_1 as

$$s(\varphi_1, l_1) = \sqrt{a^2 + l_1^2 - 2al_1\cos(\varphi_1)}. \tag{4.39}$$

The value of s can be computed from θ by substituting Equations (4.33), (4.34) and (4.38) sequentially.

The velocity mapping from $\dot{\theta}$ to $\dot{s}$ is developed here. $\dot{\varphi_1}$ can be derived from time derivates of Equations (4.25) and (4.35), Equations (4.36) and (4.37) as

$$\begin{aligned}\dot{\varphi_1}\left(l_1, l_2, \varphi_2, \varphi_3, \varphi_4, \dot{l_1}, \dot{l_2}\right) = &\left(\frac{l_1 - c\cos(\varphi_2)}{l_1 c\sin(\varphi_2)}\right)\dot{l_1} + \left(\frac{l_2 - c\cos(\varphi_3)}{l_2 c\sin(\varphi_3)}\right)\dot{l_2} \\ &+ \left(\frac{l_2 - e\cos(\varphi_4)}{l_2 e\sin(\varphi_4)}\right)\dot{l_2},\end{aligned} \tag{4.40}$$

where the values of l_1, l_2, φ_2, φ_3 and φ_4 is computed using Equations (4.33) to (4.37) respectively. From the time derivative of Equation (4.34), and rearranging gives $\dot{l_1}$ in terms of $\dot{l_2}$ as

$$\dot{l_1}(l_1, l_2, \dot{l_2}) = \frac{-l_2}{l_1}\dot{l_2}. \tag{4.41}$$

Similarly, Equation (4.26) can be used to write $\dot{l_2}$ in terms of $\dot{\theta}$ as

$$\dot{l_2}(\theta, \dot{\theta}) = \left(\frac{ed\sin(\theta)}{l_2}\right)\dot{\theta}. \tag{4.42}$$

Taking time derivatives of Equation (4.32) and rearranging gives

$$\dot{s}\left(s, l_1, \dot{l_1}, \varphi_1, \dot{\varphi_1}\right) = \left(\frac{l_1 - a\cos(\varphi_1)}{s}\right)\dot{l_1} + \left(\frac{al_1\sin(\varphi_1)}{s}\right)\dot{\varphi_1}, \tag{4.43}$$

where s, l_1 and φ_1 can be computed using Equations (4.34), (4.38) and (4.39) respectively. Sequentially substituting Equations (4.40) to (4.42) into Equation (4.43) can compute the mapping from $\dot{\theta}$ to $\dot{s}$.

Actuator to Joint Space Mapping

The inverse mapping from θ to s could not be solved analytically using a similar process. An alternative is to use numerical methods which are difficult to implement on the low-level devices like the Programmable Logic Controller (PLC). Hence, this mapping is approximated using polynomials. The forward mapping in Equation (4.39) is used to generate a set of values for s with θ having an angular resolution of $0.0001°$. These values are then approximated using least-squares to a polynomial of degree 15, which provided a residual error that is sufficiently low for the developed control applications. The derivative of the polynomial is used to map from $\dot{\theta}$ to $\dot{s}$.

The results of the mappings for both position and velocity for Dipper joint are shown in Figures 4.8b and 4.8c respectively. The plots show the non-linearities which is more prevalent in the velocity mapping.

4.5 Trajectory Following using Nonlinear Model Predictive Control

In the literature, the problem of trajectory following has been addressed by a variety of different controllers. (Calzolari et al. 2017) gives an overview of the available controllers. A comprehensive survey of controllers for Autonomous Driving are presented in Paden et al. 2016. Pure Pursuit controllers are the most popular due to its simplicity. Front and rear wheel position based feedback controllers also provide path tracking with stability guarantees. Control methods for trajectory include feedback linearization and control Lyapunov based design. These controllers have some drawbacks, limiting their applicability to specific driving scenarios, the details of which can be found in Paden et al. 2016.

In comparison with these controllers, MPC based controllers provides certain inherent advantages like constraint formulation, multiple inputs and outputs, reference tracking, etc. Increase in computational power and progress in optimal control algorithms have made Nonlinear MPC (NMPC) (Gros

et al. 2016) popular as well. Here we use one of the advanced optimal control solvers for NMPC using the multiple shooting trajectory optimization method, called the Gauss-Newton Multiple Shooting (GNMS) (Giftthaler et al. 2017). The primary advantage of the algorithm is that it provides a generic planning and control solution for nonlinear systems, even with non-holonomic constraints.

The controller uses MPC structure with a kinematic model of the system and is solved using advanced multiple shooting based trajectory optimization. The kinematic model, solver algorithm and the NMPC problem formulation are given below.

The system model as described in Kong et al. 2015 is a nonlinear continuous equation called the Kinematic Bicycle Model, where the vehicle model is simplified to a two-wheel bicycle like model with one steerable wheel. The equations describing the motion of the vehicle in inertial frame (X, Y) is given by

$$\begin{aligned} \dot{x} &= v\cos(\psi+\beta), \\ \dot{y} &= v\sin(\psi+\beta), \\ \dot{\psi} &= \frac{v}{L_r} sin(\beta), \\ \dot{v} &= a, \\ \beta &= tan^{-1}\left(\frac{L_r}{L_f+L_r}\tan\delta\right), \end{aligned} \tag{4.44}$$

where x and y are the coordinates of the CoM in the inertial frame. ψ and $\dot{\psi}$ are the yaw angle and the yaw rate, respectively. L_f and L_r are the distance from CoM to the front and rear axles, respectively. v is the current velocity of the CoM. β is the angle of v with respect to the longitudinal axis of the vehicle. a is the acceleration of the vehicle along v. δ is the steering angle of the front wheels. δ and a are the control inputs for the model. The model is depicted in Figure 4.9. For this platform, the control input is the velocity and hence a is integrated and given as input for the controller.

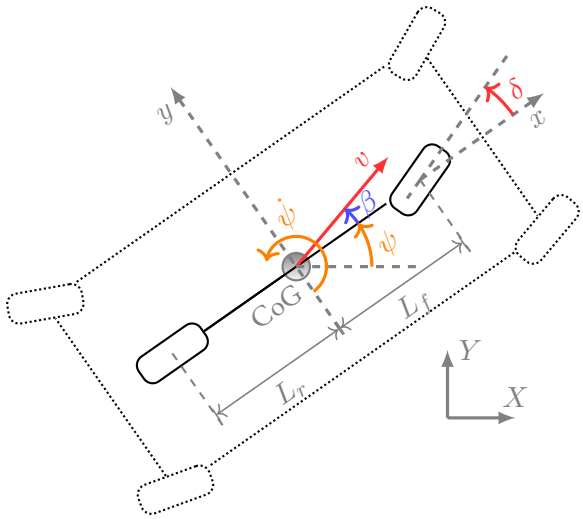


Figure 4.9: Depiction of the kinematic bicycle model which is used as the steering model for trajectory following of ARTER. The kinematic model is then mapped to the steering joints of the robot using the driving module of the mid-level controllers.

The iterative Linear Quadratic Regulator (iLQR), described in W. Li and Todorov 2004, extends the classical LQR to nonlinear systems. The iLQR starts with an initial guess and computes the LQR for the linearized system. This LQR is then used to refine the guess and get closer to the optimal solution. iLQR method is further extended and generalized by (Giftthaler et al. 2017) to include different multiple shooting variants, forming a family of iterative Gauss-Newton Shooting Methods called GNMS. The closed-loop variant of GNMS is known as the Closed Loop Gauss-Newton Multiple Shooting (iLQR-GNMS(M)) which improves convergence, contraction rates and runtimes compared to classical iLQR.

In this work, we use the Control Toolbox (Giftthaler et al. 2018) which is an open-source C++ library focusing on modeling and control of robotic systems. This library provides a very efficient implementation of the iLQR-GNMS(M) algorithm with Riccati solvers which is extended to be used as NMPC.

The trajectory tracking is formulated as a NMPC problem given by

$$\underset{\mathbf{x}(.),\mathbf{u}(.)}{\text{minimize}} \sum_{k=k_0+1}^{k_0+N} (\widetilde{\mathbf{x}}(k)-\mathbf{x}(k))^T\mathbf{Q}(\widetilde{\mathbf{x}}(k)-\mathbf{x}(k)) + (\widetilde{\mathbf{u}}(k)-\mathbf{u}(k))^T\mathbf{R}(\widetilde{\mathbf{u}}(k)-\mathbf{u}(k)), \tag{4.45a}$$

subject to

$$\dot{\mathbf{x}}(k) - f(\mathbf{x}(k),\mathbf{u}(k)) = 0, \tag{4.45b}$$

$$\mathbf{x}^- \leq \mathbf{x}(k) \leq \mathbf{x}^+, \tag{4.45c}$$

$$\mathbf{u}^- \leq \mathbf{u}(k) \leq \mathbf{u}^+, \tag{4.45d}$$

where $\mathbf{x} = \begin{bmatrix} x & y & \phi & v \end{bmatrix}^T$ are the system states and $\mathbf{u} = \begin{bmatrix} a & \delta \end{bmatrix}^T$ are the control inputs, as defined in Equation (4.44). $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{u}}$ are the reference state and control trajectories, respectively. k is the index of the sample with a constant sampling time. k_0 is the index of the sample corresponding to the current time. N is the number of samples for the fixed time horizon. $\mathbf{Q}$ and $\mathbf{R}$ are the diagonal matrices for the state and input cost weights, respectively. $f(\mathbf{x}(k),\mathbf{u}(k))$ is the system model described by Equation (4.44). $\mathbf{x}^-$ and $\mathbf{x}^+$ are the lower and upper limits of states, respectively. Similarly, $\mathbf{u}^-$ and $\mathbf{u}^+$ are the lower and upper limits for the inputs.

Equation (4.45a) gives the objective function which finds the set of states and inputs that minimizes the quadratic cost function consisting of state and input errors. The input reference is set to zero to minimize any nonzero input. Equation (4.45b) is the system dynamics constraint which is linearized for computation of Riccati backward sweep and the nonlinear model is used for simulating the shooting intervals. Equations (4.45c) and (4.45d) are the box constraints for state and input, respectively.

The controller was subjected to a simulation test, as captured by the screenshots in Figure 4.10[1], and the results were evaluated. In this setup, the localization was assumed to be ideal and free of errors. A visual comparison of the actual and reference trajectories is shown in Figure 4.11, which demonstrates a high degree of overlap. The distance and yaw errors during movement are plotted in Figure 4.12 along with the reference commands that were generated. It is evident from the figure that higher error values

[1]The video of the RViz visualization of trajectory following in simulation is available at https://drive.google.com/file/d/1jIVhOcgucI83TYC2Lb-xjzWBuZoCf44R/view?usp=sharing showing the movement of the robot along with the reference trajectory and the trajectory generated by the controller.

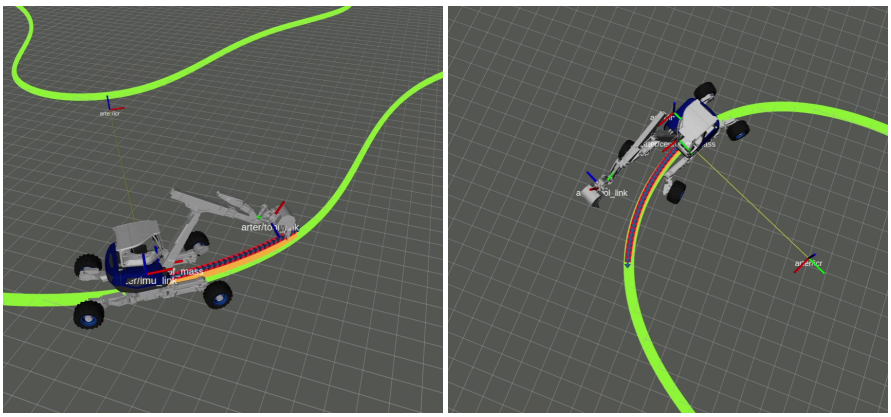


Figure 4.10: Screenshot of trajectory follower visualization in RViz. The green track is the reference trajectory, the orange track the segment given as input to the MPC and the red with blue arrows are the output from the MPC.

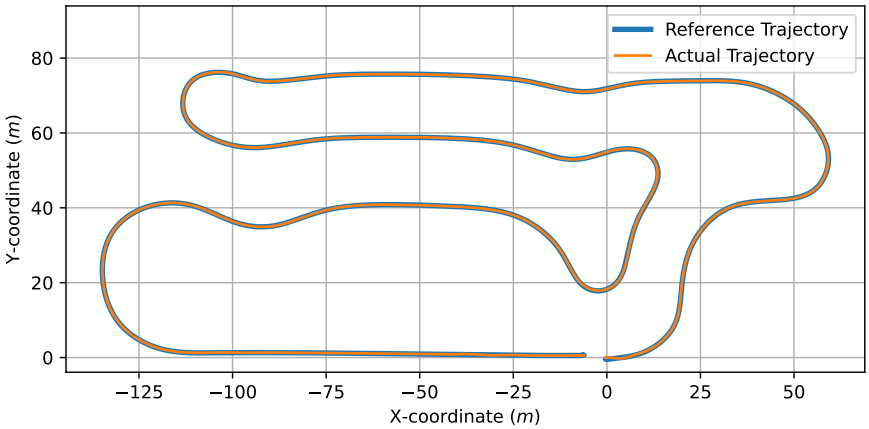


Figure 4.11: Comparison of the actual and reference trajectories of the trajectory follower in Simulation.

are exhibited during the initial 20 s, attributable to the disparity between the robot's initial pose and the trajectory's start position. Subsequent to the convergence of the trajectory, the maximum distance error is approximately 0.2 m, and the maximum yaw error is 0.2 rad.

The same controller was evaluated with lidar-based mapping and localization for autonomous driving applications in Babu et al. 2019. Further details regarding the controller and its performance can be found in that source.

4.6 Application Scenarios

The primary objective of developing ARTER is to facilitate its deployment in decontamination scenarios that are hazardous to humans. In this context the robot has been utilized in three different projects and the robot is being actively developed for use in a number of application scenarios. The control framework presented in this chapter has been evaluated in several scenarios, two of which are described in this section.

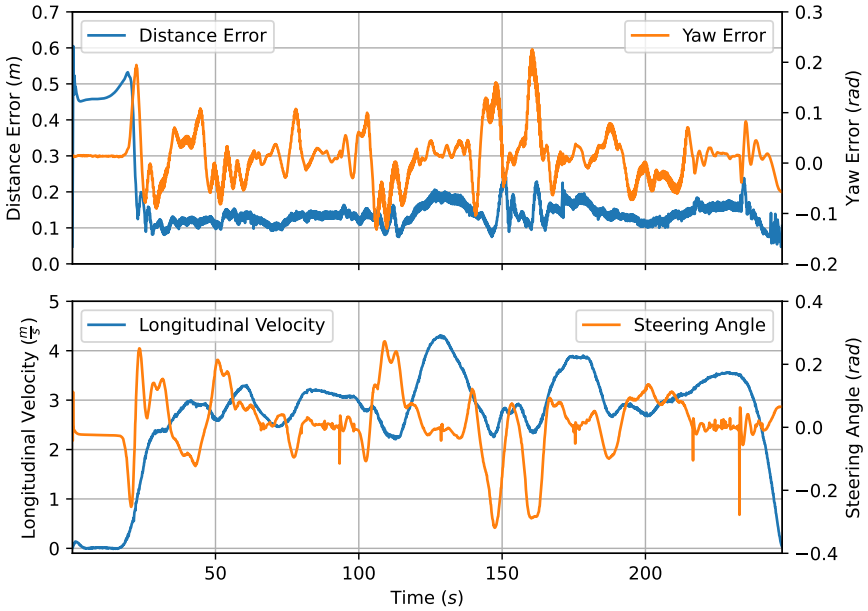


Figure 4.12: Performance of trajectory follower with distance error and angular error plotted on the top plot. The bottom plot shows the reference commands for longitudinal velocity and steering angle.

4.6.1 Remote Controlled Operation

Figure 4.13: ARTER is used in this scenario to recover barrels from environments that are hazardous for humans. This is achieved using ARTER-MCS and remote-control, where the primary operator feedback is based on video streams.

In this scenario, the robot is operated remotely using a joystick to recover partially damaged barrels[2]. This use case is shown in Figure 4.13 with the robot in operation (left) and the remote-control center (right). The user is provided with live streams of the navigating area from the cameras mounted on different positions of the robot. Additionally, the operator is supported by a range of assistive technologies including mapping, localization, joint velocity control, end-effector control, end-effector forces estimation, tip-over stability evaluation and collision warning. The employment of these tools enabled the operator to successfully complete the task with minimal difficulty.

[2] The video showing the usage of the remote control for recovering a barrel is available at https://drive.google.com/file/d/14mz_0F3JN5OxtR5Vx0U0lOWYzkn5jQx2/view?usp=sharing.

4.6.2 Autonomous Soil-Sampling Scenario

The assessment of soil contamination is essential for determining the level of risk to humans. To this end, a soil-sampling scenario was designed to collect soil samples from specific locations. These samples can then be analyzed for the presence of contaminants, thereby creating a map of the chemical composition of the soil.

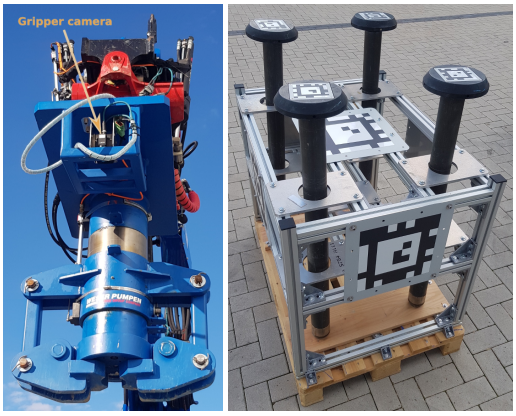


Figure 4.14: Custom developed gripper (left) and four lances on a stand (right) for autonomous soil-sampling. The stand is attached with large markers which are detectable from large distances and the individual lances are attached with smaller markers for detection using sampling gripper camera (Image source: Babu et al. 2022).

In this scenario, ARTER autonomously collects soil samples[3] using a specially developed sampling gripper and lance (Figure 4.14). The lance is a cylindrical, hollow shaft with replaceable inlines that can be swapped out after each use. The lance head serves as the grasping component, and an AprilTag marker (Wang and Olson 2016) is attached for detection and tracking through image processing. The gripper is designed to grasp the lance's head, exhibiting inherent tolerance for inaccuracies in lance positioning. Additionally, sensors, including an RGB camera, a 3D ToF camera and proximity sensors, are attached to the gripper for accurate visual servoing.

The complete sequence of tasks for soil-sampling consists of several steps. Initially, the robot navigates from the initial position to the lance-stand pose, the position of which is approximately known a priori. The precise position of the lance-stand is then detected by using the markers, and the corresponding lance is grasped and removed from the stand. The robot then moves to the desired location for soil sampling, where the lance is inserted into the ground. Subsequent to retracting the lance, the robot proceeds to the deposit location, where it places the lance into the collection bin. The sequence is repeated until all the desired samples have been collected. The high-level coordination of the tasks are controlled using behavior trees.

The process of grasping the lance entails a series of complex steps (shown in Figure 4.15), necessitating precise control of the end effector. The end-effector is initially positioned in an approximate pose over the lance stand, from where the lance marker becomes visible (Figure 4.15a). The pose estimation of this marker is then utilized to align the longitudinal plane of the manipulator with the designated lance (Figure 4.15b). Afterwards, the manipulator is moved linearly until the gripper aligns with the lance, maintaining a predefined vertical offset (Figure 4.15c). Subsequently, the end-effector is opened

[3]The semi-autonomous execution of the soil-sampling sequences are shown in https://drive.google.com/file/d/1ZOd9WjTteZRs2-QbyK97aiQRSGFA7HpZ/view?usp=drive_link.

(a) Start pose (b) Aligned (c) Approach (d) Docked (e) Partial grasp (f) Lift (g) Grasped

Figure 4.15: Task sequences during autonomous lance grasping performed using ARTER with the lances placed on the stand. (Image source: Babu et al. 2022).

and moved linearly to the final grasp position (Figure 4.15d), with the lance marker being tracked throughout. As the gripper approaches the target, the lance marker becomes occluded, resulting in a situation without active visual feedback. Upon docking, the gripper is partially closed to accommodate tolerance for positioning inaccuracies and estimation errors (Figure 4.15e). Subsequently, the lance is withdrawn from the stand, and upon achieving full withdrawal (Figure 4.15f), the gripper is closed fully to complete the grasp (Figure 4.15g).

4.7 Summary and Discussions

The walking excavator robot, ARTER, possesses diverse locomotion capabilities that make it effective in challenging environments. The software that handles the movement of the system, ARTER-MCS, is designed to control the robot in teleoperation as well as in autonomous modes. The system's architecture is structured into distinct layers, namely the lower, middle and higher levels. The lower level is responsible for executing control operations at the joint level, including parallel kinematic computations. The middle-level receives commands and converts them into corresponding joint reference commands. This layer covers several additional functionalities such as driving, steering, trajectory following, manipulator control, several assistance functions and several safety functions. Command generation for this layer is initiated by the remote control operator or by the autonomy modules located in the higher-level layer. The higher level receives sensor data and certain commands or configurations for the autonomy modules and generates the necessary commands for the middle layer. The functionalities in this layer include SLAM, traversability mapping, manipulator motion planning and navigation.

The joints of the walking excavator consists of complex parallel kinematics which needs to be modelled such that they can be controlled effectively. The modelling of the Shovel and Dipper joints are developed to compute both the forward and inverse kinematics.

A notable module within the middle layer of ARTER-MCS is the trajectory follower, which is implemented using a generic model-based NMPC method. This module provides a foundation for the extension of the model to encompass more complex systems models. The solution provided here utilizes a kinematic bicycle model and is tested and evaluated in simulation.

The main contributions in this chapter are (i) the conceptual design of the structure of the control software, (ii) modelling and solutions of the parallel kinematics, (iii) trajectory following using GNMS, and (iv) evaluation of ARTER-MCS in a realistic scenario. The assistive function modules related to terrain adaption and stepping are detailed in Chapters 5 and 6 respectively. The ARTER-MCS was

developed as part of the ROBDEKON[4] project. The implementation of the manipulation and mapping components of ARTER-MCS, and the execution of the use case involved contributions from several researchers working in the project. The ARTER robot has been part of three projects to date, and is still in the process of being developed.

[4] The project ROBDEKON is funded by the Federal Ministry of Education and Research (Grant number. *13N14675*)

Terrain Adaption for a Walking Excavator Robot using Deep Reinforcement Learning

The controller developed in Section 3.5 for SherpaTT is reconceptualized, generalized and implemented using DRL for ARTER. The design of the terrain adaption controller with different variants are detailed in this chapter, along with the training and evaluation results. Partial results of the presented work in this chapter have been published in

(i) Babu, A., P. Willenbrock, J. Tiemann, F. Bernhard, and D. Kuehn (2024). "ARTER: a walking excavator robot". In: *Biologically Inspired Series-Parallel Hybrid Robots*. Ed. by S. Kumar, A. Mueller, and F. Kirchner. 1. Auflage. Vol. 514. Elsevier Science, pp. 235–261. ISBN: 978-0-323-88482-2. URL: https://shop.elsevier.com/books/biologically-inspired-series-parallel-hybrid-robots/kumar/978-0-323-88482-2

(ii) Woock, P. and A. Babu (2022). "Autonome Robotersysteme in der Altlastensanierung". *Handbuch Altlastensanierung und Flächenmanagement*. Handbuch Altlastensanierung und Flächenmanagement 93. Aktualisierung, 3. Aufl.5111. Ed. by V. Franzius, M. Altenbockum, and T. Gerhold

(iii) Babu, A., L. C. Danter, P. Willenbrock, S. Natarajan, D. Kuehn, and F. Kirchner (2022). *at - Automatisierungstechnik* 70:10, pp. 876–887. DOI: doi:10.1515/auto-2022-0056. URL: https://doi.org/10.1515/auto-2022-0056

(iv) Babu, A. and F. Kirchner (2021). "Terrain Adaption Controller for a Walking Excavator Robot using Deep Reinforcement Learning". In: *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 64–70. DOI: 10.1109/ICAR53236.2021.9659399

The structure of the chapter is as follows: After introduction of the controller in Section 5.1, the setup for training and evaluation is given in Section 5.2. Subsequently, the design of the controller is elaborated in Section 5.3, followed by the results. Couple of improvements to the controller are also explained in Sections 5.5 and 5.6. The chapter finishes with the summary and discussions.

5.1 Terrain Adaption Controller

A terrain adaption controller ensures that the wheels maintain contact with the ground on uneven and varying terrain. There are multiple solutions to this problem. The optimal solution depends on the factors like the tip-over stability of the posture and the underlying terrain. There is an extensive amount of work in literature that addresses this problem, mainly for planetary rovers or rough terrain

robots. The solution can be broadly classified into controllers based on reactive control, planning and trajectory optimization.

Solutions developed in Iagnemma et al. 2003, Besseron et al. 2008, Tarokh et al. 2013 and Cordes et al. 2017 are dedicated reactive controllers which ensure ground contact and optimize the posture-based stability. Hutter et al. 2017 automated the ground adaption of a walking excavator by building custom hydraulic valves and using contact force optimization. This solution does not optimize for posture stability. Reactive control solutions do not normally use valuable information like the terrain height-map.

On the contrary, sampling-based solutions like Hauser et al. 2008a, Brunner et al. 2012 and Reid et al. 2020 use the terrain information either by sampling or forming grids. But these solutions are generally computationally expensive and generate non-smooth trajectories, which needs smoothening. Moreover, these do not provide a runtime controller.

Trajectory optimization-based methods generate smooth trajectories and can handle high degrees of freedom. Jelavic et al. 2021a combines trajectory optimization with sampling-based algorithms for motion planning of a walking excavator robot. But this needs explicit mathematical modeling of the robot and the terrain.

The main contributions of this chapter are the design of controllers and the development of the simulation-based setup to learn the controllers. The work also contributes to the study of the effects of terrain compression on convergence and performance. The learned controller is effectively a combination of a local planner and a runtime controller. The controller and the learning setup are easily adaptable for other robots by developing a simulation model of the robot and the basic controllers. The objective O-3b is addressed here.

5.2 Learning Setup

The robot should be able to interact in an environment with realistic terrain to learn the controller. It also needs a set of controllers which provide the basic functionalities similar to the actual hardware. The components of this setup are detailed here.

Figure 5.1 shows the overview of the controller setup. The Simulation block contains the components of the simulation, based on PyBullet (Coumans 2015; Coumans and Bai 2016) engine. The simulation runs at 240 Hz. The Unified Robotics Description Format (URDF) of the robot is the input to the simulation with simplified geometries to speed up the simulation. Apart from simulating the robot, the simulation engine computes the distance between the links and the ground.

The ARTER-MCS provides a set of controllers which control the different functionalities of the robot. The MCS uses Robot Operating System (ROS) framework. The module Trajectory Follower generates the steering commands such that the robot can follow a commanded trajectory. Pure-pursuit path tracking is used for following the trajectory, and it generates the steering command for a virtual vehicle which is based on a simple kinematic bicycle model. The Steering Controller takes the steering command and computes the command positions for the steering joints in each leg. This is done by computing the Instantaneous Center of Rotation (ICR) of the virtual vehicle and then computing the steering angles for each wheel so that the wheel axis intersects the ICR. The simulation uses this command to control the joint. In this work, the longitudinal velocity of the robot is kept constant at $0.5\,\mathrm{m/s}$. Active Suspension Controller is responsible for controlling the height of the wheels. In this

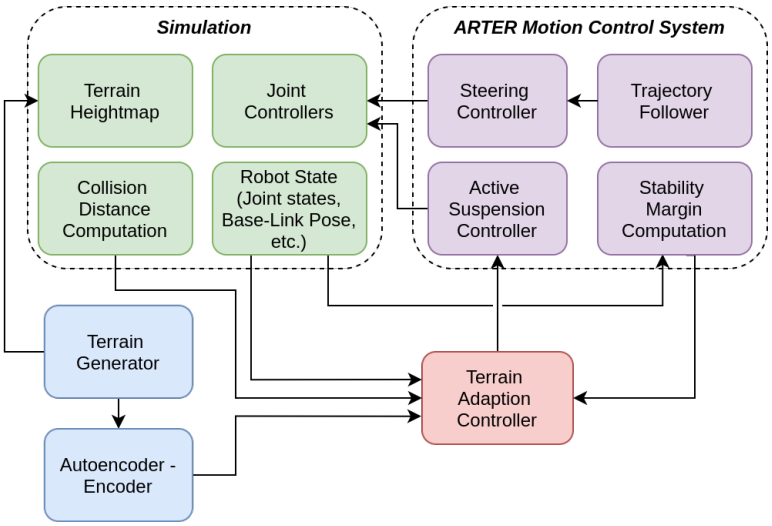


Figure 5.1: Overview of the terrain adaption controller setup with interconnections between simulation, controller and ARTER-MCS.

work, they filter the commands from the terrain adaption controller and forward them to the simulation. Stability Margin Computation module computes the stability of the current pose based on the contact points. The control loop runs at 10 Hz.

The sample terrains are generated by the Terrain Generation module which is given as height-field to the simulation with a grid resolution of $0.2\,\mathrm{m} \times 0.2\,\mathrm{m}$. In the simulation, the terrain is updated as grid elements with a size of $8\,\mathrm{m} \times 8\,\mathrm{m}$ each. New terrain grids are added and removed based on the distance from the current robot position. This dynamic terrain handling speeds up the simulation. The terrain information is also compressed and given as observation to the terrain adaption controller. This module is implemented as a neural network and trained using deep reinforcement learning. Figure 5.3 shows a screenshot from ROS 3D visualization tool RViz showing the robot, the local terrain and stability margins.

5.2.1 Stabilizer Joint Kinematics

(a) Front legs (b) Rear legs

Figure 5.2: Kinematics of ARTER legs showing the joints and the parallel kinematics.

The vehicle has different suspension modules for the front and the back legs. In the front legs, the wheel is connected to the chassis through the joints swivel, stabilizer and steering as shown in Figure 5.2a. The swivel and stabilizer joints move the wheel laterally and vertically relative to the chassis,

respectively. The steering joint orients the wheels. The stabilizer joint for the front legs consists of a four-bar linkage which makes sure that the pitch of the wheels remains independent of the stabilizer angle. In the simulation, the mimic joint emulates the parallel kinematic.

For the rear legs, the stabilizer is connected to the chassis, followed by swivel and steering joints, as shown in Figure 5.2b. The kinematics of the stabilizer also differ from those of the front legs, as it lacks the parallel linkage, and the wheels pitch with changes in the stabilizer angle. For the rear legs, the parallel kinematics of the swivel and steering are connected such that any change in the swivel joint will not change the orientation of the steering relative to the chassis.

5.2.2 Terrain Generation

The purpose of terrain generation is to generate smooth and continuous terrain. Perlin noise described in Perlin 1985 is the most commonly used algorithm for terrain generation in animation and games. But this method tends to create directional artifacts. To overcome this limitation the Simplex noise algorithm (Perlin 2002) was developed. Since the Simplex algorithm is patented, OpenSimplex noise is an alternative.

OpenSimplex noise for 2D can be modified using the frequency term, ω, and the amplitude term, A. The frequency term scales the XY footprint of the features, and the amplitude term scales the height. Apart from this, a seed S randomize the features. The terrain being generated is a combination of different OpenSimplex noises given by

$$\mathcal{T}(x, y) = \sum_{l=1}^{L} A_l \mathcal{O}(\omega_l x, \omega_l y, S_l), \tag{5.1}$$

where x and y are the X and Y coordinates respectively, $\mathcal{O}$ is the OpenSimplex2D noise function, l is the index of the noise layer. L is the number of layers. For training and evaluation, $L = 2$, ω for the layers are 0.03 and 0.05 respectively, and A for the layers are 7.0 and 2.0 respectively. The seed parameter S is generated randomly every time the environment is reset. These values generate a terrain which is traversable for the robot, yet challenging for the controller.

5.2.3 Stability Margin

Choosing the appropriate stability margin for walking or mobile robots is critical for their safety and performance. Garcia in Garcia et al. 2002 compares and classifies several such static and dynamic stability margins. NESM, introduced in Hirose et al. 1998, is one of the stability margins which give optimal values in case of uneven terrain and when ignoring inertial effects and manipulation forces. NESM is an extension of the ESM, defined in Messuri 1985. ESM is the minimum energy necessary to tip over the robot about the support polygon edge. NESM is ESM normalized by the robot weight, given by

$$h_{ne} = \frac{1}{W} \min_i (\|R_i\|(1 - \cos\alpha)\cos\beta), \tag{5.2}$$

where W is the weight of the robot, i is the index of edges on the support polygon, R_i is the line connecting the CoM and closest point on the edge, α is the angle between line R_i and the vertical plane passing through the edge. β is the angle between the line formed by rotating R_i by α, and the vertical axis. In this work, the NESM is computed assuming that the wheels are in contact with the

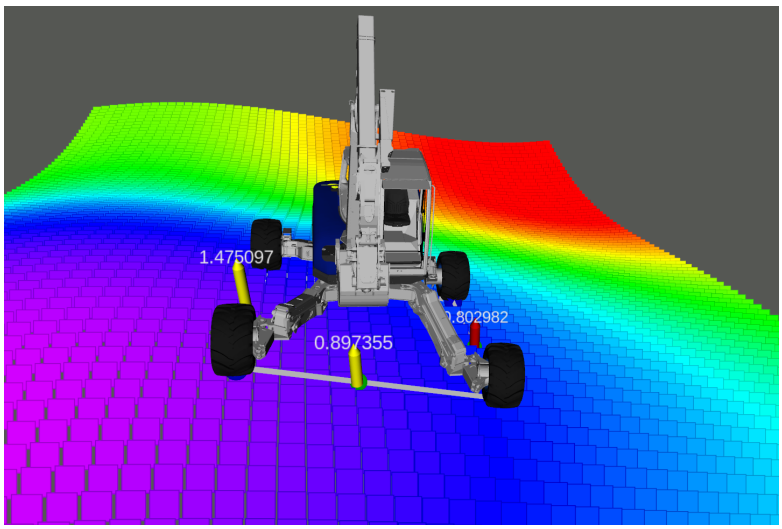


Figure 5.3: Screenshot from showing the robot on an artificially generated terrain with varying slopes. The stability margin corresponding to each edge of the support polygon is also shown. The red arrow shows the NESM.

ground. Hence, this value does not contain any direct information regarding the terrain. A more detailed description and depictions of NESM is provided in Appendix D.

5.3 Terrain Adaption Controller Design

This section explains the terrain representation, observation and action spaces, rewards, implementation details and the controller types. The primary objectives of the controller are to (i) maintain the tip-over stability of the robot, (ii) to avoid chassis collision with ground, and (iii) to maintain contact of the wheels with ground. The secondary objectives are to (i) minimize the movements of the joints, and (ii) to maintain a user preferred roll and pitch. These objectives are encoded as rewards for the reinforcement learning.

5.3.1 Terrain Representation

Knowledge about the terrain around the robot is important for adapting to the terrain. Providing the height-map directly as observation is not feasible as it will result in high number of observations, affecting the learning time and convergence. A grid size of 64×32 will result in 2048 additional observations. Due to the continuous structure assumption of the terrain, it is possible to compress the height-map to reduce the number of observations.

One of the methods best suited for this purpose is autoencoder as described by Rumelhart et al. 1986 and Ranzato et al. 2007. Neural network-based autoencoders use the encoder network with three hidden layers of 512, 128 and 32 sizes. The input layer is of 2048 size. The output layer can have sizes 2, 4, 8, or 16. Rectified Linear Unit (ReLU) is the activation function. The decoder network is the same as the encoder network except in reversed order, and the output layer has a hyperbolic tangent (Tanh) activation function.

5.3.2 Observations, Actions and Rewards

The main design factors for deep reinforcement learning controllers are the observations and rewards. The following explains these components in detail.

The complete set of observation space consists of (i) positions q_j and (ii) velocities $\dot{q}_j$ of the stabilizer joints, (iii) roll ψ and (iv) pitch θ of the chassis, (v) chassis distance to ground, (vi) wheel distances for all the wheels to ground and (vii) the encoded height-map h_n. Chassis and wheel distances are extracted from the current robot pose and the terrain height-map.

The actions are the normalized joint velocities, $\dot{q}_{c,j}$, in the range $[-1, 1]$, for the stabilizer joints $j = \{0, \ldots, 3\}$. The actions given by the controller are filtered before being applied to the joints in the simulation. A first-order filter is used for this and this smoothens the command and introduces a delay. Filtering helps bring the system closer to reality, accounting for factors such as communication and actuation delays, which are not present in simulation.

The proper shaping of the reward function is critical for the convergence of the reinforcement learning algorithm. Dense rewards are better suited for this problem as there are multiple simultaneous objectives. The reward functions are designed based on intuition as well as by trial and error.

The total reward for the ground adaption controller consists of individual reward terms which are formulated in such a way that they either increase or decrease exponentially depending on the factor. The parametrized exponential function gives the flexibility to change the profile of the reward functions easily. The rewards that increase exponentially are given by the function

$$\mathcal{R}_+(f, f_{min}, f_{max}, p) = e^{p(\mathcal{N}(f, f_{min}, f_{max})-1)} \tag{5.3}$$

and the function for exponential decrease is

$$\mathcal{R}_-(f, f_{min}, f_{max}, p) = e^{-p\mathcal{N}(f, f_{min}, f_{max})}, \tag{5.4}$$

where e is the exponential function and p the scaling factor. The normalized reward factor is computed by function $\mathcal{N}(f, f_{min}, f_{max})$ where f is the input factor, f_{min} and f_{max} are the minimum and maximum for normalization respectively. Since the inputs of equations Equations (5.3) and (5.4) are normalized, the outputs of are also within the range $[0, 1]$. The parameters f_{min}, f_{max} and p determine the shape of individual reward terms. The design of individual reward terms are as follows.

There are multiple postures that ensure ground contact, and stability margin is one of the primary factors which helps choose the better posture. The reward term for NESM stability margin is given by

$$r_{ne} = \mathcal{R}_+(h_{ne}, 0.2, 0.7, 4), \tag{5.5}$$

where h_{ne} is defined in Equation (5.2). This term gives the maximum reward 1 if the NESM is equal or higher than 0.7. The reward reduces exponentially if it goes below this value.

Rewarding ground clearance avoids the collision between the chassis and the ground. Ground clearance reward is given by

$$r_{gc} = \mathcal{R}_+(d_{gc}, 0.2, 0.5, 4), \tag{5.6}$$

where d_{gc} is the minimum distance from chassis to the ground. This term provides the highest reward for maintaining a ground clearance of at least $0.5\,\mathrm{m}$.

The main objective of the controller is to ensure continuous contact between wheels and the ground. The wheel contacts are rewarded by

$$r_{wc} = \mathcal{R}_{-}\Big(\max_{w}(d_{wc,w}), 0.0, 0.5, 4\Big), \tag{5.7}$$

where $\max_w(d_{wc,w})$ gives the maximum distance to ground for wheels with indices $w = \{0, \ldots, 3\}$.

It is desirable to achieve the rest of the objectives using as little movement of the joints as possible. The reward term for commanded velocities is given by

$$r_{cv} = \mathcal{R}_{-}(\max(|\dot{q}_{cv,j}|), 0.0, 0.5, 4), \tag{5.8}$$

where $\max(|\dot{q}_{cv,j}|)$ gives the maximum absolute commanded velocity for joints with indices $j = \{0, \ldots, 3\}$. The reward increases as movement decreases.

The robot can maintain reference roll and pitch angles by rewarding this behavior. It is an optional reward as this gives a more natural-looking motion. Ensuring that the sensors are oriented in the desired direction may also be important in some cases. The reward term for roll angle, ψ, is given by

$$r_{\psi} = \mathcal{R}_{-}(|\psi|, 0.0, 1.0, 4) \tag{5.9}$$

and for pitch angle, θ, is given by

$$r_{\theta} = \mathcal{R}_{-}(|\theta|, 0.0, 1.0, 4). \tag{5.10}$$

The total reward is the weighted sum of the reward terms defined from Equation (5.5) to Equation (5.10) and is given by

$$r = k_{ne}r_{ne} + k_{gc}r_{gc} + k_{wc}r_{wc} + k_{cv}r_{cv} + k_{\psi}r_{\psi} + k_{\theta}r_{\theta}, \tag{5.11}$$

where the values of weights are $k_{ne} = 0.2$, $k_{gc} = 0.2$, $k_{wc} = 0.2$, $k_{cv} = 0.2$, $k_{\psi} = 0.1$ and $k_{\theta} = 0.1$. The weights add up to 1.

The total reward is a positive value, and a reward of -1 is assigned only in the event of a robot tip-over and subsequent resetting of the simulation environment. The reward function profile for the different reward components are plotted in Figure 5.4.

5.3.3 Controller Types

Here we define three different groups of controllers. The first one is the Terrain-Controller (TC) which uses all the observations, except the chassis and wheel distances. The second group Base-Controller (BC) contains all the observations except the compressed terrain information. The third group Base-Terrain-Controller (BTC) which contains all the observations. The TC and BTC controllers have different types, based on the latent space size of the autoencoder. In this work, controllers TC2, TC4, TC8 and TC16 corresponding to the latent space sizes 2, 4, 8 and 16 respectively are tested. BTC also has similar subtypes BTC2, BTC4, BTC8 and BTC16.

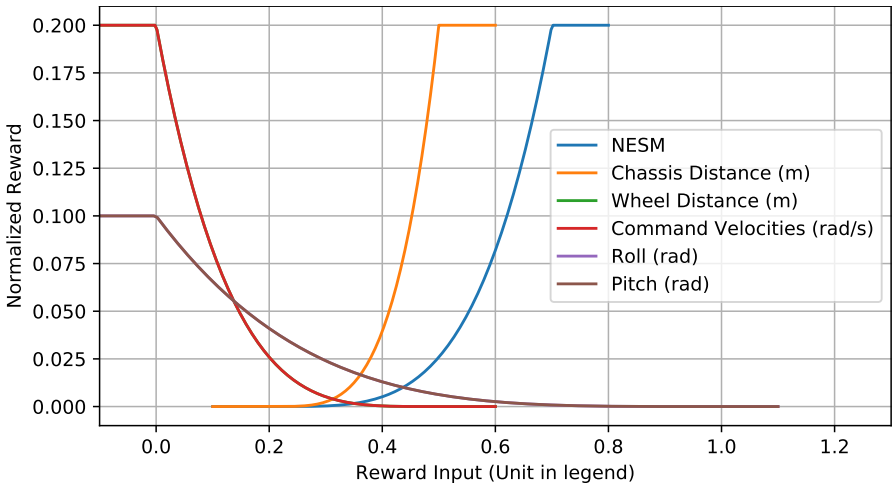


Figure 5.4: Different inputs for the rewards and the generated weighted normalized rewards after passing through the rewards function. The reward for Wheel Distance overlaps with that of Command Velocities and hence not visible. Similarly, reward plot for Roll overlaps with that of Pitch.

5.3.4 Implementation

SAC is the used reinforcement learning algorithm since it has proven to be successful in locomotion applications as shown in Haarnoja et al. 2018a and Haarnoja et al. 2018b. The key idea of SAC is the incorporation of entropy regularization which rewards the randomness in policy, encouraging the policy network to explore more. The modified objective function J as a function of the policy π is given by

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))], \tag{5.12}$$

where s_t and a_t are the states and actions respectively, ρ_π is the marginals of the trajectory distribution induced by the policy. $\mathbb{E}$ is the expectation, r is the reward at each time step, $\mathcal{H}$ is the entropy term and α is the temperature parameter which gives the relative weight of the entropy term.

It also makes use of clipped double-Q learning introduced in Fujimoto et al. 2018 to avoid overestimation of Q-function. SAC uses three different networks for learning the state-value function, the soft Q-function and, the policy function.

The implementation in Stable-Baselines3 (Raffin et al. 2019) is used which is based on PyTorch (Paszke et al. 2019). Table 5.1 gives the list of parameters and their values.

The network is a Multilayer Perceptron (MLP) with two hidden layers of sizes 400 and 300, using the ReLU activation function. The temperature parameter α is automatically optimized.

5.4 Comparison of Controllers

The results are presented here for the terrain representation, learning convergence and the controller performance.

Table 5.1: SAC Parameters for Terrain Adaption.

Learning rate (η)	7.3×10^{-4}
Discount factor (γ)	0.98
Soft update coefficient (τ)	0.02
Temperature parameter (α)	auto
Train frequency	256
Gradient steps	64
Initial value of log standard deviation	-3
Generalized State Dependent Exploration (gSDE)	active
Number of critics	2

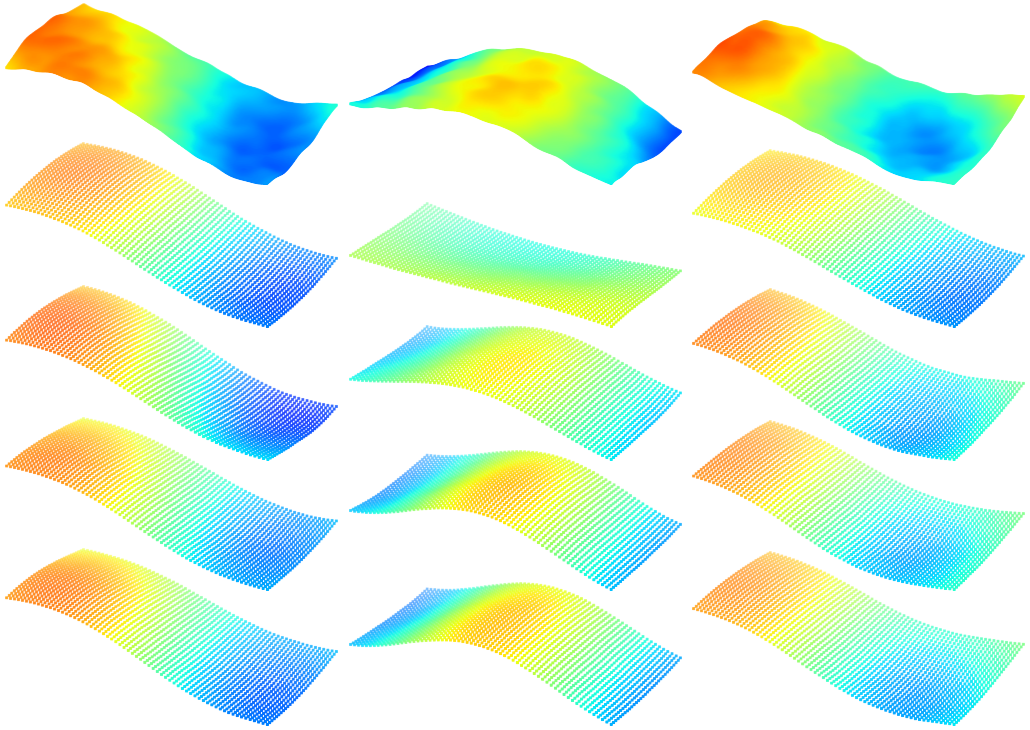


Figure 5.5: Terrain representation using autoencoders with different latent space sizes. The columns are three different terrain samples. First row is the originally generated terrain. The rest of the rows are the terrains encoded and decoded with a latent space size of 2, 4, 8 and 16 respectively.

The autoencoders were trained in PyTorch (Paszke et al. 2019) for 30 000 randomly generated samples. Figure 5.5 shows a visual comparison of original terrain sample and the ones reconstructed using autoencoders. It can be seen that autoencoder is able to represent the main features of different types of terrain well. Figure 5.6 shows the average RMSE of height for reconstructed terrains for 500 samples. As expected, the error reduces with increase in the latent space size. The difference in errors for latent space sizes 4, 8 and 16 is minimal.

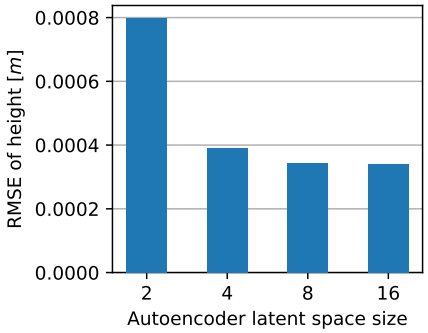


Figure 5.6: RMSE of height for terrain encoded and decoded using auto-encoder for varying encoder output sizes.

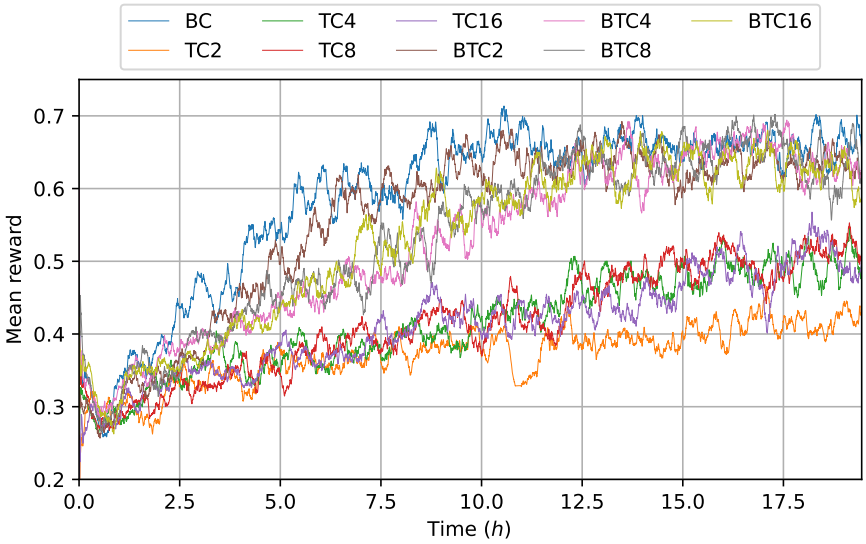


Figure 5.7: Moving average of rewards generated during the training of different terrain adaption controllers.

The controllers are each trained[1] for 20 h of simulated time as most of the controllers seem to converge within this time as shown in Figure 5.7. The BC and BTC controllers all converge within 12 h. The TC controllers take longer to converge.

The learned controllers are evaluated for 1 h each in simulation and the resulting average rewards are plotted in Figure 5.8. The BC and BTC controllers perform better than the TC controllers. This difference is due to the contact distances given as observation which avoids the inaccuracies in the terrain compression. The BC controller performs slightly better than the BTC controllers since the BC controller has fewer observations and does not have the sometimes conflicting information given by the terrain and the contact distances. The performance of the TC controllers reflects the inaccuracies in terrain reconstruction.

[1] The video showing the sample motion of the robot during the training process and the evaluation of the BC controller can be found at https://drive.google.com/file/d/13mz3Vx9q_INdas4IPqa8k5ZX4TaF2hlF/view?usp=sharing. It also shows the automatically generated terrain along with the stability margins.

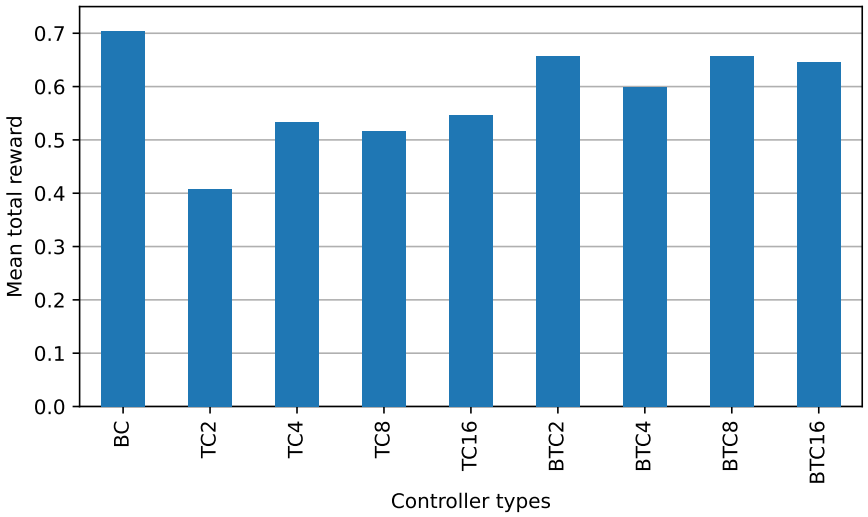


Figure 5.8: Mean rewards generated during the evaluation of different terrain adaption controllers.

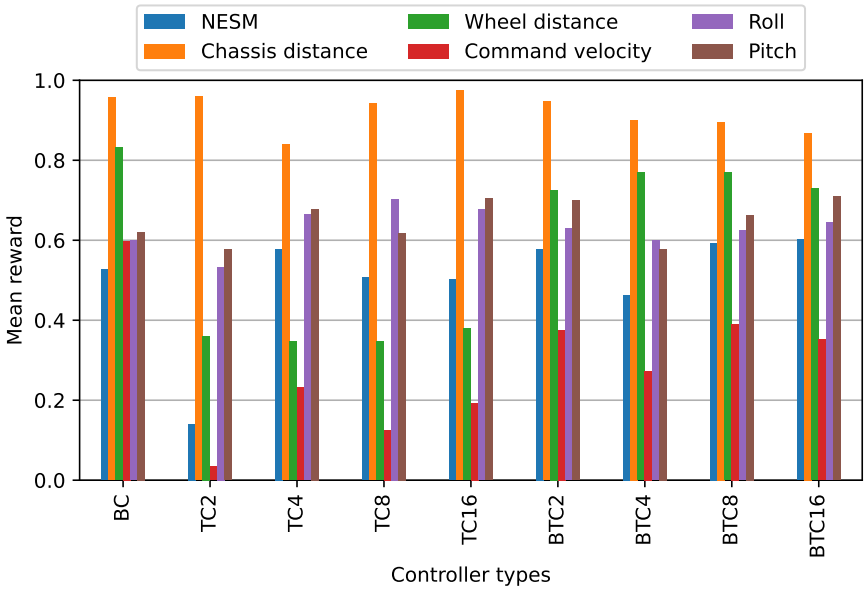


Figure 5.9: Breakdown of the mean rewards generated during the evaluation of different terrain adaption controllers.

Figure 5.9 shows the average of the individual rewards during evaluation. The rewards for chassis distance, roll and the pitch do not differ much between the controllers. The stability margin reward is similar, except for TC2 for which it is low. This difference is due to the high inaccuracies in terrain compression. The wheel distance reward is lower for TC controllers. BC and BTC controllers can better maintain ground contact since the wheel distances are part of the observations. The command velocity rewards seem to be the most contributing factor to the difference in performances. A lack of relevant observations for the TC controllers or conflicting information for the BTC controllers makes the controller to move the joints more to compensate.

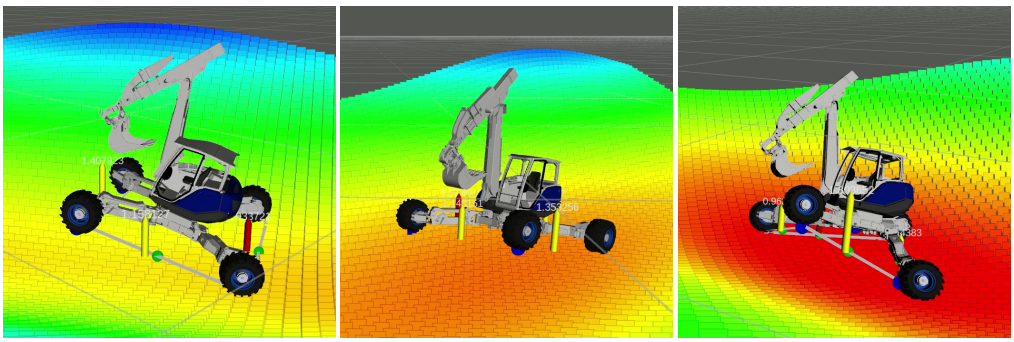


Figure 5.10: Screenshots during training of terrain adaption controller.

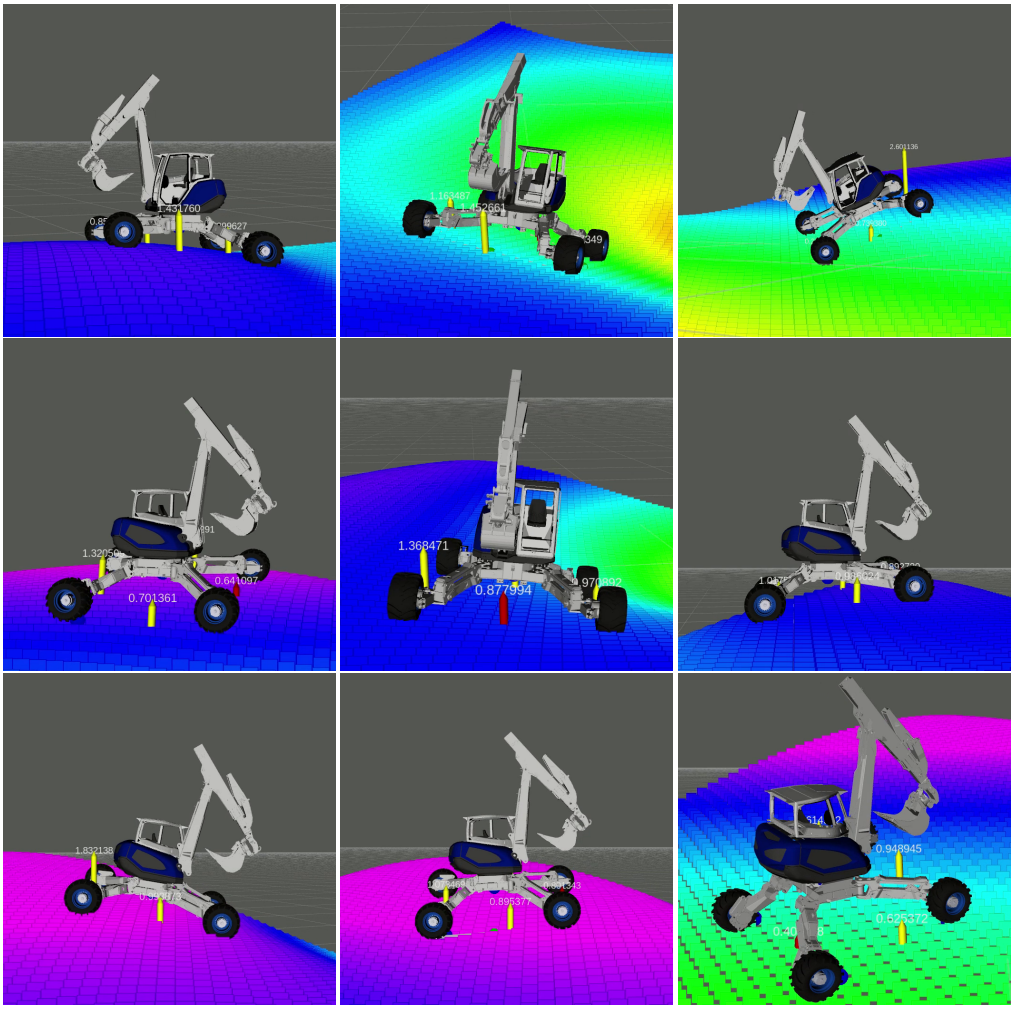


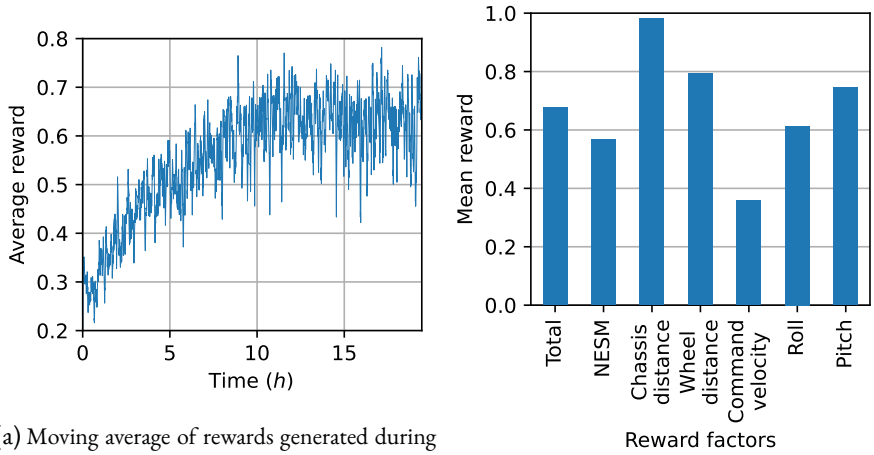
Figure 5.11: Screenshots after training of terrain adaption controller.

In comparison to the GAP in Cordes et al. 2017, the controller in this work achieves several additional objectives. Both controllers can maintain good wheel-ground contacts. GAP can accurately maintain reference roll and pitch angles as well. The controller in this work is also able to optimize stability and avoid chassis collision without any explicit mathematical modeling of the kinematics. A quantita-

tive comparison of the performance with other state-of-the-art controllers is beyond the scope of this work. Screenshots of the setup during training and after training are shown in Figures 5.10 and 5.11 respectively.

5.5 Terrain Adaption with Contact Detection

This work improves upon the previously described controllers by enabling faster convergence in learning and designing the controller state space to more closely align with the states available on the actual system, requiring minimal preprocessing. The contact detection gives a normalized value representing the probability of having a wheel in contact with the ground. This value on the actual robot can be generated using the actuator pressure sensor values. This is different to the previous controller, where the extracted distance to ground for each wheel is used. The encoded terrain with a latent space dimension of 4 is also included in the observation.



(a) Moving average of rewards generated during the training of the terrain adaption controller in simulation.

(b) Breakdown of rewards during evaluation.

Figure 5.12: Training and evaluation results for controller with contact detection.

The progress of rewards during learning is shown in Figure 5.12a where the learning converges after approximately 10 h of simulated time. The resulting learned controller was evaluated for 1 h of simulated time and the average rewards are shown in Figure 5.12b. The plot shows the total reward as well as the breakdown of individual rewards corresponding to the objectives mentioned above. The total reward is the weighted sum of the individual rewards. The relative performance of each reward factor can be controlled by manipulating these weights, which in turn can be tuned for specific application and operator requirements.

5.6 Terrain Encoding with β-Variational Autoencoder

The terrain representation is a crucial aspect for controllers developed in this chapter, as it enables the adaptation of robot suspension to varying terrain conditions. Both the accuracy of the representation and the size of the latent space play a significant role in training success and overall controller performance. Therefore, this aspect is further analyzed in the following sections. Different architectures for autoencoders are evaluated to compare their performances. Apart from the Linear Autoencoder

(LAE) in Section 5.3.1, two other architectures are evaluated: Convolutional Autoencoder (CAE) (Ranjan et al. 2017) and β-Variational Autoencoder (β-VAE). The details of the architectures are shown in Appendix E. A comprehensive survey of autoencoders is provided in P. Li et al. 2023.

The main difference in CAE in comparison to LAE is the use of Convolutional Neural Network (CNN) instead of MLP. LAEs convert the input height-map into a one-dimensional vector and hence losses the two-dimension features of the image. On the contrary CAEs keeps the two-dimensional structure intact. CAEs are primarily used with images for denoising, dimensionality reduction, etc. The encoder part of CAE extracts features by performing convolution operation on the input data and compresses this. The convolutional layer applies different filters and the pooling layer reduces the dimension. Upsampling and dimensionality expansion is done by the decoder part using transposed convolution layers. CAEs are evaluated to check if the use of convolutional layers will improve the reconstruction accuracy.

Variational Autoencoders (VAEs) (Kingma 2013) introduce a probabilistic latent space to represent the compressed information. The mapping of the input data to preformed to a probabilistic distribution enables the VAE to learn a more continuous and structured latent space. The latent space is represented by the mean and the variance. Additionally, the loss term during training contains a Kullback-Leibler (KL) divergence term to ensure that the latent space distribution is close to a prior distribution and hence helps in regularization.

An adjustable hyperparameter β is added in β-VAE (Burgess et al. 2018) to control the weightage between the reconstruction error and the KL divergence in the loss function. The β scaling factor when higher than 1 regularizes strongly for and results in a more disentangled latent space. When $\beta = 1$ the β-VAE is the VAE. Lower values of β encourages better reconstruction at the cost of disentanglement of latent space.

The disentangling capabilities of β-VAE is improved in Burgess et al. 2018 by controlling the encoding capacity of the latent space. This is achieved by introducing a training process where the encoding capacity is gradually increased by increasing the maximum permitted average KL divergence from zero. This promotes robust learning and maintains the capability of the β-VAE to generate latent space components that are qualitatively different and makes distinctive contributions during generation. The network using this process is here called as Disentangled β-Variational Autoencoder (D-β-VAE).

Several autoencoders formed by using LAE, CAE and D-β-VAE with latent spaces 2, 4, 8 and 16 are trained and evaluated. The results for reconstruction loss and maximum distance error are shown in Figure 5.13. There is only negligible difference between the performances of the different autoencoder architectures. The error is higher for latent space with the latent space dimension of 2 and the error is lower and consistent for latent space dimensions 4, 8 and 16. The reconstruction loss for all the networks with latent spaces higher than 2 is below 1.5×10^{-4} m and the maximum distance error is below 0.08 m.

The compactness and quality of the information stored in the latent spaces could influence the training of the controller. Lower dimension with distinctive features for latent spaces could potentially improve the DRL convergence as it will reduce the state space and also reduces the need for exploration. The latent spaces of CAE is interpolated and displayed with a color map in Figure 5.14. As expected the information in the latent spaces are spread across all the values and not easily interpretable. For comparison, the latent spaces learned by D-β-VAE is shown in Figure 5.15. The D-β-VAE learns the latent spaces with distinctive and interpretable features. The first row corresponding to the first latent space represents the general curvature of the height map about the center. The second and the third

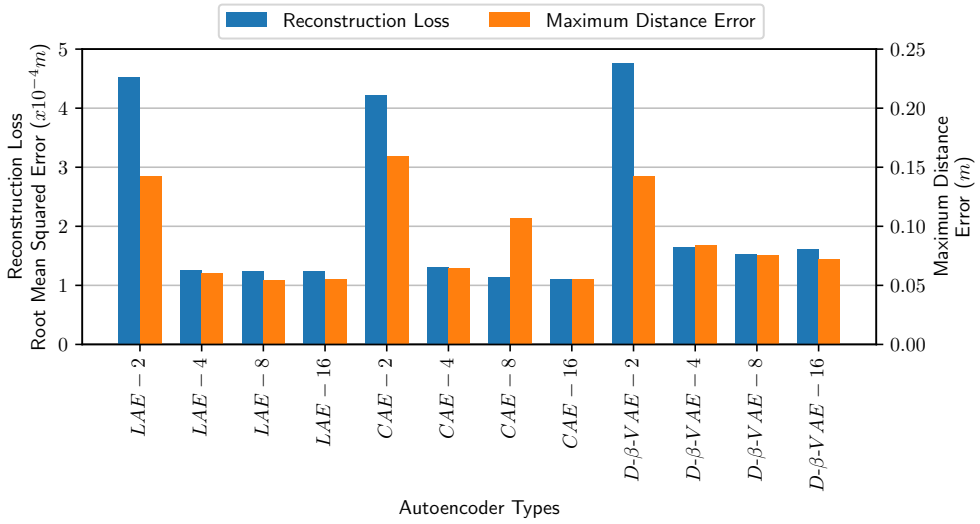


Figure 5.13: Terrain reconstruction loss for different types of autoencoders.

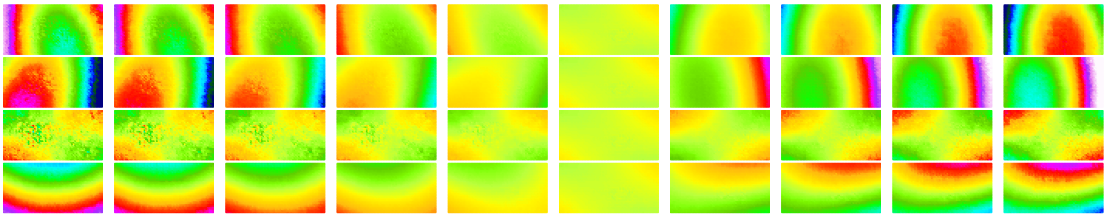


Figure 5.14: CAE-4 latent space interpolation showing the information distributed among all the latent spaces. Most of the information is concentrated in the first three rows which corresponds to the curvature, roll and pitch respectively.

latent spaces correspond to the slope along the vertical and horizontal axes. The fourth latent space does not have any information and hence unused. This also shows why the all the autoencoders with latent spaces higher than three performs equally well.

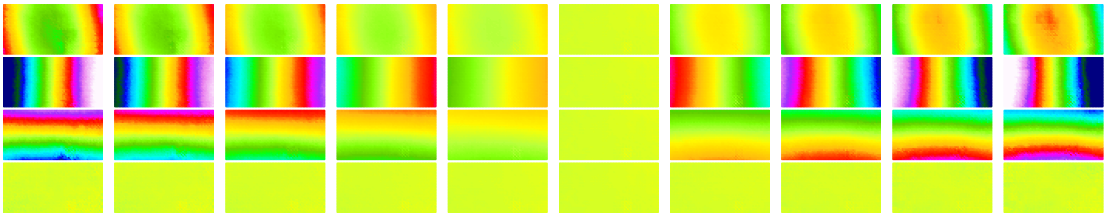


Figure 5.15: D-β-VAE-4 latent space interpolation showing that the information is compacted in the initial latent spaces with the last one having practically no information.

5.7 Summary and Discussions

Terrain adaptation controllers have been demonstrated to facilitate the automation of the active suspension of a walking excavator robot in uneven terrain. This study demonstrates the efficacy of deep

reinforcement learning in the training of the controller. The controllers take into account the tip-over stability of the robot, ground clearance for the chassis, and ground contact for the wheels. The incorporation of compressed terrain information is a feature of certain controllers. To compress the height-maps, they were transformed into a latent space of different sizes using autoencoders.

It has been observed that the controller that utilizes chassis and wheel distances as observations, in the absence of terrain information, attains optimal performance. However, incorporating additional terrain information into this controller results in a marginal decline in performance. Controllers that utilize compressed terrain as an observation, in the absence of chassis and wheel distances, demonstrate functionality, though with diminished performance. This discrepancy is primarily attributable to the incorporation of wheel distance and command velocity rewards. It is noteworthy that controllers relying exclusively on terrain information also demonstrate good performance. The utilization of these controllers offers the benefit of eliminating the need for extracted information, such as the distance of the wheel from the ground, in this context.

A number of extensions to the controller were also developed. One such extension involves the utilization of contact detection instead of contact distance, thereby yielding a controller that exhibits good performance. This controller offers the advantage that it is easier to transfer to a real system where the contact can be detected directly. Another enhancement is the use of disentangled β-VAE to find the necessary dimension of the autoencoder latent spaces that can effectively represent the terrain.

This chapter presents the development of a generalized controller that is independent of the kinematics of the legged-wheeled systems, and capable of simultaneously achieving multiple objectives. The controller design, which is independent of the kinematics of the system, can be transferred to systems with adaptable suspensions to achieve similar results. This development is in alignment with the objective specified in O-3b.

Stepping for a Walking Excavator Robot using Hierarchical Deep Reinforcement Learning

One of the most complex locomotive movements for a walking excavator is stepping, which requires the actuation of numerous joints in multiple sequences, in a synchronous manner to successfully traverse an obstacle. This research employs a hierarchical reinforcement learning approach to guide the design of the stepping controller, leveraging domain knowledge. The complex control task is divided into multiple, simple subtasks, each of which can be trained and validated in isolation. Subsequent integration of these subtasks into a hierarchical design facilitates the construction of the desired functionality. Furthermore, the controller is guided during training by using invalid action masking. In certain instances, the sequence in which the subtasks must be executed is predetermined, thereby reducing the training burden to the selection of the corresponding goals. Following training, the motion sequences generated by the controller are observed to bear a strong resemblance to those produced by an expert operator. The design of the controller, along with the results of training and evaluation in simulation, are presented in this chapter. This addresses the objective O-4. Partial results of the presented work in this chapter is under review in

(i) Babu, A. and F. Kirchner (2025). "Stepping Locomotion for a Walking Excavator Robot Using Hierarchical Reinforcement Learning and Action Masking". In: *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)*. IEEE, Hangzhou, CHINA (Accepted)

An introduction to stepping locomotion is provided in Section 6.1 including stepping for hybrid locomotion systems, related work and eventually the motivation for development of a new controller. Afterwards the preliminaries regarding Guided Reinforcement Learning (GRL) and several ways of achieving it are discussed in Section 6.2. The learning setup and modeling of the learning environment are detailed in Section 6.4. The controller design is elaborated in Section 6.5, followed by the implementation details in Section 6.6 and evaluation results in Section 6.7. Finally, the chapter is summarized in Section 6.8.

6.1 Stepping Locomotion

Stepping locomotion is defined as a mode of movement whereby an agent propels itself forward by exerting force on the ground with its limbs. This mode of locomotion is associated with the coordinated movement of the limbs, and typically involves the repetition of a sequence of movements. A defining feature of stepping locomotion is the temporary loss of contact between the limbs and the ground. This is necessary to step over obstacles and traverse challenging terrain.

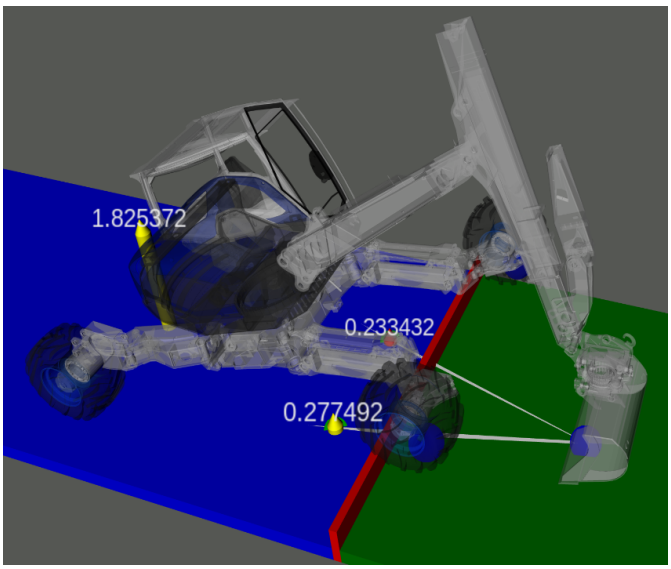


Figure 6.1: Screenshot of the visualization of the learning environment where the part in red is the obstacle.

The characteristics of stepping include repetition cycles, stability, balance, terrain adaptability and efficiency. Typical legged walking involves the repetition of stance and swing phases. During stance phase, the limb is in contact with the environment and applying forces; in swing phase, the limb is not in contact and is moved to the position of the next intended contact point. Stability is paramount to prevent falls and to account for environmental uncertainty. Balancing is defined as the active maintenance of stability by adapting movements in real time. Adaptation to motion sequences is necessary when traversing uneven terrain or changes in substrate. Energy efficiency is also a key factor in defining the stepping locomotion pattern.

In robotics, the stepping locomotion is mimicked using legs or appendages of robots. Robots capable of such movement are termed "walking robots" and are versatile and adaptable to uneven and difficult terrains, compared to wheeled or tracked robots. The prevalence of bipedal and quadrupedal robots is quite high. Robots with a single leg, as well as those with more than four legs, have also been developed. The application of walking robots is diverse, including search and rescue missions, exploration of challenging terrains, and more. The control of walking robots is complex and robustness depends on the accuracy of limb placement and the quality of environmental sensing. In comparison to wheeled or tracked robots, walking robots demonstrate lower energy efficiency.

6.1.1 Stepping for Hybrid Locomotion

Hybrid locomotion robots can combine different modes of locomotion for enhanced mobility. In some cases, this includes stepping locomotion as well. So far, this thesis has dealt with controllers that used legged wheel systems to adapt to uneven terrain and for force distribution. However, the establishment and release of contact with the environment was not addressed. In hybrid locomotion involving stepping, limbs actively establish and release contacts to overcome obstacles or ascend steps. The incorporation of stepping capabilities in hybrid locomotion robots, akin to their counterparts in walking robots, has been demonstrated to enhance traversability and navigation capabilities.

Walking excavator robot like ARTER has hybrid locomotion capabilities including stepping locomotion. The use of stepping locomotion makes this robot very versatile in terms of traversability in uneven and challenging terrain. Typically, wheeled-legged systems can perform stepping by shifting the CoG within a planned support polygon before shifting one of the wheeled legs for stepping. ARTER has a strong manipulator with a large workspace making it suitable for using it as a leg for locomotion. This combination of wheeled legs and normal leg like structures available for locomotion makes the system a unique platform. ARTER has the capability to use its manipulator for locomotion especially for climbing and stepping locomotion modes. The wheels of the excavator are normally expected to be in continuous contact with the ground. For stepping the manipulator is used to temporarily lift the wheels and step over the obstacles.

There are several challenges involved in developing control solutions for such as system. One of the main challenges is the large number of DoFs which makes most of the currently available solutions ineffective. Another challenge is the kinematics of the limbs which are different for the front and rear legs as well as the manipulator. This kinematics combined with the different contact constraints for legs and wheels makes controlling the robot cumbersome. Contact constraints change based on which limbs are in contact with the ground. A solution to assist the humans by autonomously stepping over obstacles during remote control or autonomous operation is necessary to enable broader use across various applications. The objective is to develop controller for stepping over an obstacle with motion sequences similar to that of an expert operator.

6.1.2 Related Work

Researchers have developed several solutions for stepping in hybrid locomotion robots. The solutions are highly dependent on the type of the system and type of locomotion available.

The wheeled-legged robot Momaro in Schwarz et al. 2016 has developed a control solution where it is able to climb steps, step into a vehicle and step over obstacles. This approach utilizes a control strategy that shifts the body—and consequently the CoG—to maintain stability, even when a leg is lifted off the ground. The leg is then lifted and placed at a desired location before shifting the CoG such that the next leg can be lifted. The main limitation of such a solution is that it is not transferrable to systems which cannot shift its CoG. An advancement to this controller is proposed in Klamt and Behnke 2017 where a planner is developed which can plan both driving and stepping locomotion. The stepping part is done in two stages where abstract maneuvers are planned first and then expanded to detailed motion sequences. Klamt and Behnke 2018 plans for larger areas by planning different levels of abstraction, coarse planning for longer distances and fine resolution planning for shorter distances.

The quadrupedal wheeled-legged robot CENTAURO in Laurenzi et al. 2018 uses Linear Model Predictive Control (LMPC) framework for generating walking gait. A similar robot Pholus in J. Sun et al. 2020 tackles the problem using hierarchical structure with layers for hybrid foot placement planning, CoM trajectory planning and WBC. Gait graphs are extended for hybrid foot placement which is then used to generate CoM trajectories using MPC. These CoM trajectories are then given as input for WBC which in turn uses QP to generate joint trajectories. In Bjelonic et al. 2020 a more dynamic quadrupedal ANYmal with wheels online trajectory optimization framework is used for generating hybrid walking-driving locomotion. This solution separates the trajectory planning of wheel and base trajectories, making it real-time capable, which are tracked using hierarchical WBC. A similar solution in Medeiros et al. 2020 uses trajectory optimization using terrain map and stability constraints, formulates it as a NLP, the solution which is tracked by hierarchical WBC. Another solution for the same

robot in Bjelonic et al. 2021 uses MPC for simultaneous optimization of wheel and base trajectories. The kinodynamic model of the robot is optimized online without the need for motion heuristics. Yet another motion optimization framework is proposed in Hosseini et al. 2023 incorporating dynamic jumping behaviors as well. This framework is based on MPC which uses time-varying rigid body dynamic model of the robot.

There are several solutions specific to walking excavator robots. Jelavic and Hutter 2019 provides a solution where a trajectory optimization framework is developed with focus on walking excavators. The plan considers the whole body of the robot, wheeled and non-wheeled limbs, as well as include contact scheduling. This work was extended in Jelavic et al. 2020, where TO is used to generate references for end-effectors and joints. The references are then tracked using whole body hierarchical optimization. Jelavic et al. 2021a combines sampling and optimization to cope with challenging terrain. The optimization part also considers non-holonomic rolling constraints. The solutions work effectively in a variety of terrains. Jelavic et al. 2023 provides a similar solution where the hybrid motion planning uses sampling based solutions for approximate planning and trajectory optimization for solution refinement. The solution is evaluated using HEAP and ANYmal robots.

There are also some learning based solution that has been developed for hybrid locomotion. Cui et al. 2021 formulates it as an adaptive optimal control problem for a wheel-legged robot in the absence of an accurate dynamic model. Here DRL and adaptive dynamic programming is combined to derive a learning-based solution. Sometimes motion priors are used to learn the skills as shown in Vollenweider et al. 2023 where the adversarial motion prior-based RL is used to train the locomotion controllers of a wheeled-legged robot in order to avoid tedious reward function tuning.

6.1.3 Motivation

The existing solutions that are applicable for the stepping of a walking excavator is either a combination of sampling and trajectory optimization or using learning based methods. The main drawback of the methods that use explicit mathematical formulation is that it becomes very difficult and non-intuitive to reformulate the problem for influencing the intended behavior. Generated motion from learning based methods like reinforcement learning can be influenced using the reward function which is not expressive enough for a complex behavior like stepping. Our attempts at using a single agent reinforcement learning for the stepping task resulted in very little progress and generated movements that were not realistic.

The primary contribution of our approach lies in the methodology employed for developing the controller, which facilitates the integration of domain expertise, thereby shaping the desired behavior. This is accomplished by decomposing the complex task into a hierarchy of smaller sub-tasks, each with simple reward functions, rules, limits and constraints. Furthermore, invalid actions are explicitly masked to steer the training process. In comparison to existing solutions, this approach offers distinct advantages in terms of simplicity in development and the capacity for intuitive integration of domain knowledge, while yielding comparable results to that of available in literature. The controller is trained for three distinct types of stepping, thereby demonstrating its versatility. To the best of our knowledge, this is the first approach that combines HRL with Invalid Action Masking (IAM) for stepping locomotion in a hybrid locomotion robot.

6.2 Guided Reinforcement Learning

GRL based methods can provide a solution to the challenges faced by learning based methods by integrating domain knowledge in the various levels of the DRL pipeline. An extensive survey, taxonomy and evaluation of the GRL methods can be found in Eßer et al. 2023. The DRL pipeline consists of problem representation, learning strategy, task structuring and sim-to-real methods, each of which provides possibilities for injecting additional knowledge into the system. Apart from the most common methods for GRL such as state representation and reward design, there are some not so common methods such as HRL and IAM.

6.2.1 Hierarchical Reinforcement Learning

HRL is a form of DRL where complex tasks are decomposed into smaller sub-tasks (or skills) which are easier to solve. This set of high-level tasks and low-level tasks form a hierarchy of policies and actions. The actions of high-level tasks act as goals for the low-level tasks. In environments with complex and long-term dependencies, this hierarchical structure provides more efficient learning. HRL is one of the methods of GRL by injecting additional knowledge during the task structuring step. Comprehensive surveys of different HRL methods, taxonomy, etc. are given in Barto and Mahadevan 2003; Pateria et al. 2021. One of the earlier related work, detailed in Kirchner 1998, employs hierarchical Q-learning for the locomotion of a six-legged robot.

Several behaviors in animals seem to suggest the utilization of hierarchical structure to learn and make decisions. Studies (Eckstein and Collins 2019; Neftci and Averbeck 2019; Rasmussen and Eliasmith 2014; Ribas Fernandes et al. 2011) have been conducted to understand the link between HRL computational models of learning and behavioral neuroscience. The interaction between neural structures in brain for planning and decision-making (prefrontal cortex) and those supporting action selection and reinforcement learning (basal ganglia) shows evidence (Botvinick et al. 2011) for HRL. Behavioral studies (Donnarumma et al. 2021) have demonstrated hierarchical behavior where rodents use higher-level information like landmarks to structure the search and perform actions. Similarly, animals also show the ability to learn skills separately and then combine them together to achieve a long-term goal (Gobet et al. 2001).

Hierarchy, temporal abstraction and state-action abstraction are the main properties of HRL. The tasks are arranged in a hierarchy with different levels where the high-level task gives input to the low level tasks. Such tree type structure makes the overall complex task more manageable. Temporal abstraction (Precup and Sutton 2000) is also one of the properties of HRL where different tasks have different time scales. The higher-level tasks operate for more time-steps taking macro-actions while the lowest-level tasks execute primitive actions at each time-step. This allows the policies to work on less granular timescale, reducing the number of decisions to be made. Options framework (Bacon et al. 2017; Stolle and Precup 2002) is a popular method for implementing HRL by defining temporarily extended course of action including initiation condition, policy and termination condition. State and action abstraction is property of higher level task to focus on the states and actions which are relevant for achieving long-term goals.

There are several advantages to using HRL. The most attractive one is sample efficiency which is achieved by breaking down the tasks whereby the learning gets simplified, reducing the necessary amount of time and data. The low-level tasks can be used by multiple high-level tasks improving the

reusability of the individual policies. HRL also scales well to large and complex environments where classic DRL will be infeasible due to the curse of dimensionality.

HRL has been widely used in locomotion of robotic systems. One such controller for bipedal walking is DeepLoco (Peng et al. 2017) where two-level hierarchical control framework is used. Initially, the low-level controllers are learned which achieve robust walking gaits to achieve stepping targets and walking styles. The high-level controllers are then learned which generate the desired step targets for the low-level controllers based on terrain maps. Another solution for legged locomotion in T. Li et al. 2020 divides the goal-directed locomotion into two parts: learning primitive skills and using planning to sequence these skills. A solution which is trained in simulation and then transferred to real system is designed in Tan et al. 2021. Here a hierarchical learning framework is introduced where the high-level policy adjusts the low-level trajectory generators to adapt to the terrain. In a recent work Lee et al. 2024, navigation and locomotion of autonomous wheeled-legged robot is solved using a hierarchical RL framework.

6.2.2 Invalid Action Masking

During the training of a DRL agent, in some cases the DRL algorithm spends time in trying out actions which are not allowed or does not make sense, given a certain state, resulting in low training efficiency. In robotics, examples of this include actions during joint limits, moving into obstacles, self-collision of manipulators, etc. Several transformations (Kanervisto et al. 2020) of the action space are possible to improve the training efficiency: Removing invalid actions, discretize continuous actions, and convert multi-discrete actions into discrete. Removing invalid actions in continuous spaces often involved removing actions that will not help the agent progress in the task. In cases where the continuous actions are too large and hence affects exploration, it is discretized for better efficiency. Similarly, multi-discrete actions are converted into single discrete action to avoid the action space to explode combinatorially.

There are several ways of handling such invalid actions as described in Huang and Ontañón 2022. Two of the most popular methods are Invalid Action Penalty and IAM. In case of Invalid Action Penalty, a negative reward is given for taking invalid actions which in turn will train the agents to use only valid actions. IAM is the technique of masking out actions in a full action space which are invalid, depending on the observations. The masked out actions are then used to directly increase the probability of taking valid actions. IAM has been successfully used in several complex real-time strategy and multiplayer battle arena games (Berner et al. 2019; Vinyals et al. 2019; Ye et al. 2020).

The implementation of IAM depends on the type of DRL algorithm. In value-based algorithms, the highest values of the estimated action-value function $Q(s, a)$ is selected. When an action is masked, the Q-value is set to a very low value ensuring that the action is never selected. For policy gradient algorithms, the probability of invalid actions are set to 0. Since the actions are selected according to the probability distribution of the model, these actions will never be selected. Softmax function shifts the logits to probability domain and if the logits for invalid action is $-\inf$, the sampling probability of these actions become 0. Gao 2018 and Zahavy et al. 2018 used IAM in Deep Q-Network (DQN) to speedup training. A trainable action-mask is used by Y.-C. Wu et al. 2020 for improving training efficiency of a model-based DRL algorithm. Action masking for one of the most popular DRL algorithm, Proximal Policy Optimization (PPO), is implemented in Tang et al. 2020.

In this chapter we will develop controllers which extends the use of IAM to guide the DRL algorithm. Typically, IAM is used to avoid actions that are inherently invalid because they have no effect or result

in states from where the robot cannot recover. It is also possible to guide the exploration in the correct direction by masking all the actions that does not guide the agent in the intended trajectory. Such guidance involves input from experts and hence results in better training efficiency and robustness, also reducing the need for complex reward engineering. The algorithm for applying masks with PPO is detailed below.

Masked-PPO

PPO (Schulman et al. 2017) is a state-of-the-art DRL algorithm which is widely used in training agents in a variety of applications including board games (Patankar et al. 2024; Yang et al. 2023), video games (Shao et al. 2019; C. Yu et al. 2022) and robotics (Shahid et al. 2022). The main objective behind development of the algorithm is to create a balance between sample efficiency, implementation effort and hyperparameter tuning effort. This is achieved by ensuring that the change in policy at each step is within a limit while computing an update at every time-step. PPO is an on-policy algorithm, which means that the algorithm uses a particular policy to collect experiences, which is in turn used to improve the policy. PPO2 is an updated Graphics Processing Unit (GPU)-enabled version which can use multiprocessing in vectorized environment. The key concepts that are the basis for PPO are proximal updates, clipped surrogate objective and advantage estimation. The following describes the CLIP version of the PPO algorithm.

Policy gradient methods need an estimate of the gradient of the policy for performing stochastic gradient ascent optimization. The gradient is typically estimated using the gradient estimator

$$\hat{g} = \hat{\mathbb{E}}_t \Big[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \Big], \tag{6.1}$$

where $\hat{\mathbb{E}}$ is the expectation, which is the weighted sum of outcomes multiplied by their probabilities. π_θ, which is parametrized by θ, is the stochastic policy which outputs an action a_t for a given state s_t. A_t is the advantage function. The $\hat{}$ indicates that the values are estimates.

The objective function that is used by automatic differentiation algorithms has the same gradient as the policy gradient estimator and is given by

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \Big[\log \pi_\theta(a_t|s_t) \hat{A}_t \Big]. \tag{6.2}$$

The objective of PPO as proposed in Schulman et al. 2017 is given by

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \Big[\min \Big(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \Big) \Big], \tag{6.3}$$

where $r_t(\theta)$ is the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ between the old policy and new policy. ϵ is the parameter to limit the clipping ratio. The first term inside the min is the conservative policy iteration term (L^{CPI}) or the unclipped part. The second term modifies the objective by clipping the probability ratio such that having r_t outside the limits $[1-\epsilon, 1+\epsilon]$ does not give any incentive during optimization. This ensures that new policy stays close to the current policy.

Computing estimates of the advantage function ($\hat{A}_t$) requires learning of the value function as well. In PPO the same network is used for policy and value functions, augmented by entropy bonus to ensure exploration. The combined loss function as given by

$$L^{CLIP+VF}(\theta) = \hat{\mathbb{E}}_t\left[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right], \tag{6.4}$$

where c_1 and c_2 is the coefficients for the value-function loss and the entropy bonus respectively. $L^{VF} = \left(V_\theta(s_t) - V_t^{targ}\right)^2$ is the squared-error value loss and S is an entropy bonus.

PPO algorithm is extended in Tang et al. 2020 for discrete action environment by adding masks for the actions. Action mask is used to remove the invalid actions for a particular state to be removed from consideration by the learning algorithm. The environment provides a mask indicating which actions are invalid, and then ignores the invalid actions. Masking with PPO is implemented by using only valid actions during the collection of the trajectory, and during stochastic descent while computing Equation (6.4).

The output of the policy is a softmax layer which provides probabilities of all the discrete actions in the environment. The outputs before the softmax layer produces the unnormalized scores called the logits. Since the invalid actions are already ignored by the algorithm, the probabilities of the valid actions needs to be re-normalized. This is carried out by computing softmax outputs for only valid actions. The output of the policy with masks (Huang and Ontañón 2022) applied is defined by

$$\pi_\theta^M(\cdot|s_t) = softmax(mask(l(s_t))), \tag{6.5}$$

where l is the logits outputted by the original policy π_θ, $softmax$ for a vector $\mathbf{z} = (z_1, \cdots, z_K)$ defined by $softmax(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ and the $mask$ function is defined by

$$mask(l(s_t))_i = \begin{cases} l_i(s_t) & \text{if } a_i \text{ is valid in state } s_t \\ M & \text{otherwise} \end{cases}, \tag{6.6}$$

where M is a very large negative number, resulting a very small probability for invalid actions.

6.2.3 Guided Reinforcement Learning with Hierarchical Reinforcement Learning and Invalid Action Masking

Learning to step for system like ARTER has multitude of inherent challenges. The main challenge is the high number of DoFs which results in large action space which is difficult to explore even for the state-of-the-art DRL solutions. Normally in not so complex systems, reward functions are designed to guide the algorithm in the correct direction. In the case of stepping this is complicated due to the long sequence of actions with different objectives and varying constraints. A combined reward function representing all the needs is difficult. Similarly, the observation space is also huge due to the complexity of the system and environment which reduces the sampling efficiency. Many actions required by the system involve repeated sequences. In the case of direct reinforcement learning, these sequences must be learned repeatedly to achieve success. This will drastically reduce the sampling efficiency and may even prevent the system from learning a viable solution due to the long sequences.

One of the most popular approach available in literature is to scale up simulation which runs in parallel until a solution is reached. An alternative is to do imitation learning from human operator who will perform the task and the DRL algorithm solution is guided by this input.

The solution proposed here to overcome the above challenges is to guide the DRL by giving additional expert guidance in several aspects including structural design of the controller, reward function design, masking invalid actions and proposing a controller execution sequence. Converting the DRL to GRL is done by extensively using HRL and IAM. The objective is to make the problem tractable and to strike a balance between design effort and sampling efficiency.

The main purpose of the HRL is to break the complex long horizon stepping to smaller dedicated tasks each with simpler and tractable objectives. As the observations for each controller contains only states that are relevant for the specific task, it is smaller in comparison to the original complete state. Similarly, since the tasks are simpler compared to the original stepping task, the rewards for the tasks are simpler and hence easier to design.

IAM is used in two different ways. The first way is by masking invalid actions such as moving into joint limits as well as other obvious actions that are not valid or takes the robot to undesired states. The second way is to guide the higher level controllers by pre-defining the sequence of choosing the lower-level controllers. This simplifies the learning effort of the higher-level controller to selecting a goal for the particular controller rather than learning which controller to use as well as the goal for each controller. This is not strictly required, but improves the sampling efficiency and guides the learning process.

The combined usage of HRL and IAM for guiding the learning of the controllers provide several advantages. The main advantages are reduced sampling efficiency, lower effort in reward engineering and reduced need for computing infrastructure. Apart from this, the simpler controllers and guiding produces controllers that are each robust for a particular task. The lower-level controllers can also be reused for learning more higher-level controller which serve different purposes.

6.3 Formalism of Hierarchical Reinforcement Learning

A DRL agent has the objective of interacting with an environment and find a policy which maximizes the cumulative reward. The environment is considered to a Markov Decision Process (MDP) which is defined by the tuple $< S, A, P, r >$ where S is the state-space and $s_t \in S$ is the state at time step t. A is the set of actions and $a_t \in A$ is the action taken time t. $P(s_{t+1}|s_t, a_t)$ is the transition probability of reaching the state s_{t+1} after taking the action a at state s. $r(s_t, a_t)$ is the reward as a function of state and action.

The main objective of a DRL algorithm is to find an optimal policy π^* such that

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ Q^{\pi}(s_t, a_t), \forall s_t \in S, \forall a_t \in A, \tag{6.7}$$

where $Q^{\pi}(s_t, a_t)$ is the Q-value while following the policy π, and it is defined by

$$Q^{\pi}(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{a \sim \pi(s)} \left[\sum_{i=1}^{T} \gamma^i r(s_{t+i}, a_{t+i} | s_t, a_t) \right], \tag{6.8}$$

where T is the total number of time-steps in an episode and γ is the discount factor.

In order to learn the optimal policy, the DRL algorithm has to explore the states and action spaces and collect trajectories. The trajectories are then used to learn an optimal policy. The trajectories could be finite or infinite, depending on the environment. Exploration is challenging if the state and actions spaces are large or if the episodes have a long time horizon. Standard DRL algorithms will not perform well or even fail in such cases.

HRL decomposes the main task into smaller subtasks which are then assembled together to form a hierarchy of controllers. The control policies are then learned using DRL. In the hierarchy, the lowest-level controllers generate primitive actions, which are directly executed in the environment. The higher level policies learn to choose the appropriate subtasks in the lower hierarchy and provide the appropriate goals for these subtasks. All the subtasks learn to perform their own specific tasks using internal rewards. There are several methods to perform HRL. Pateria et al. 2021 and Hutsebaut-Buysse et al. 2022 provides comprehensive surveys of different HRL methods. The mathematical formalism of HRL as defined in Pateria et al. 2021 is described below.

Every subtask has different components including policy, reward, goals, initiation condition and termination condition. Let a subtask be denoted by ω^l where l is the level in the hierarchy, $l = 1$ denotes the lowest level outputting primitive actions. π_{ω^l} is the policy of the subtask which maps states of the subtask to the primitive actions or goals for the lower tasks. r_{ω^l} is the subtask specific rewards and g_{ω^l} is the goal associated with a specific subtask. This goal is a state $s \in S$ and rewards might be defined based on these goals. Additionally, the subtasks will have the execution components initiation condition, I_{ω^l}, and termination condition β_{ω^l}. Initiation and termination conditions are the set of states or set of conditions in which execution of the controller is started and terminated respectively.

HRL uses as its basis Semi-Markov Decision Process (SMDP) which is similar to MDP but additionally involves explicitly time for which a subtask is executed. The transition function of the SMDP is defined as

$$P\big(s_{t+c_{\omega_t}}, c_{\omega_t}|s_t, \omega_t\big) = P\big(s_{t+c_{\omega_t}}|s_t, \omega_t, c_{\omega_t}\big)P(c_{\omega_t}|s_t, \omega_t), \tag{6.9}$$

where c_{ω_t} denotes the number of time-steps for which ω_t is executed, starting from state s_t.

The expected cumulative reward obtained by following the policy π_{ω_t} of the subtask π_{ω_t} for c_{ω_t} time-steps is given by

$$R(s_t, \omega_t) = \mathbb{E}_{a\sim\pi_{\omega_t}(s)}\left[\sum_{i=0}^{c_{\omega_t}-1} \gamma^i r(s_{t+i}, a_{t+i})|s_t, a_t = \pi_{\omega_t}(s_t)\right]. \tag{6.10}$$

The Q-value can now be calculated by

$$Q(s_t, \omega_t) = R(s_t, \omega_t) + \sum_{s_{t+c_{\omega_t}}, c_{\omega_t}} \gamma^{c_{\omega_t}} P\big(s_{t+c_{\omega_t}}, c_{\omega_t}|s_t, \omega_t\big) \max_{\omega_{t+c_{\omega_t}}} Q\big(s_{t+c_{\omega_t}}, \omega_{t+c_{\omega_t}}\big), \quad \forall s \in S, \forall \omega \in \Omega \tag{6.11}$$

An optimal policy is the one that maximizes the above Q-value which depends on the cumulative reward Equation (6.10) and the transition function Equation (6.9) which in turn is determined by the policy π_{ω_t}. Learning in HRL hence needs to learn multiple policies at different levels of hierarchy. HRL consists of two principal parts: the subtask space and the hierarchical policy.

The subtask space (Ω_H) is the superset of all the subtasks.

$$\Omega_H = \{\Omega_{\omega^1}, \Omega_{\omega^2}, \Omega_{\omega^3}, \cdots, \Omega_{\omega^\Gamma}\},$$

where ω^l is the subtask at level l of hierarchy and Ω_{ω^l} is the set of subtasks under subtask ω^l. For the lowest level where the primitive actions are executed $l = 1$ and for the highest level $l = \Gamma$.

The hierarchical policy (π_H) is the policy of the complete hierarchy resulting from the recursive choice of subtasks and actions, starting from ω^Γ to the primitive actions by ω^1. The policy at the highest level π_Γ will choose a subtask $\omega^{\Gamma-1} \in \Omega_{\Gamma-1}$. The policy of subtask $\omega^{\Gamma-1}$ runs the policy $\pi_{\omega^{\Gamma-1}}$ until termination condition $\beta_{\omega^{\Gamma-1}}$. These runs recursively until the lowest level a primitive action $a = \pi_{\omega^1}(s)$ is taken. The hierarchical policy is this recursive process where a primitive action is taken defined by $a = \pi_H(s)$.

The HRL problem is to find the optimal hierarchical policy π_H^* and the optimal subtask space Ω_H^*

$$\Omega_H^*, \pi_H^* = \underset{\Omega_H}{\operatorname{argmax}} \underset{\pi_H|\Omega_H}{\operatorname{argmax}} Q^H(s,a), \forall s \in S, \forall a \in A, \tag{6.12}$$

where the Q-value for a hierarchy is defined by

$$Q_H(s_t, a_t) = \mathbb{E}_{a\sim\pi_H|\Omega_H}\left[\sum_{i=0}^{c_{\omega_t}-1} \gamma^i r(s_{t+i}, a_{t+i}) | s_t, a_t = \pi_{\omega_t}(s_t)\right]. \tag{6.13}$$

In Equation (6.12) there are two parts of the problem that needs to be solved: subtask discovery and learning hierarchical policy. Subtask discovery is the process of defining the different subtasks which can be found automatically or defined manually using domain knowledge. Learning hierarchical policy is the DRL for each of the subtasks which can be done simultaneously or using a bottom-to-top approach. In the stepping developed controller, the subtask discovery is done manually using domain knowledge, and the training is done in a bottom-to-top approach.

6.4 Setup and Modelling

The learning of a complex task such as stepping needs an elaborate setup and modeling of the environment for formulating the problem and training the controller. Apart from representing the internal robot states, the terrain and the interaction with the terrain need to be modelled. These states are depicted in Figures 6.2 to 6.4.

The internal states are the joint positions represented by $q_i \, \forall i \in \{0, \cdots, 6\}$ and $q_i \, \forall i \in \{ FL, FR, RL, RR\}$ for the manipulator joints and the stabilizer joints of front-left, front-right, rear-left and rear-right legs respectively. $\dot{q}_i$ are the joint velocities. Tool Base Point (TBP), which is attached of the Roto joint, is used as reference for kinematic operations of the manipulator and is defined with respect to BCS. The cylindrical coordinates, height (h_T), radius (ρ_T) and angle (ϕ_T), are used to define TBP pose.

As shown in the Figure 6.5, three different types of stepping terrain types - obstacle, step and gap - are used as the environments. Hereafter, the obstacle terrain is used as the basis for explaining the setup, design and results. It consists of a flat plane with a thin obstacle of varying height as depicted in Figure 6.4. The flat surface is placed on the XY-plane of the World Coordinate System (WCS) with

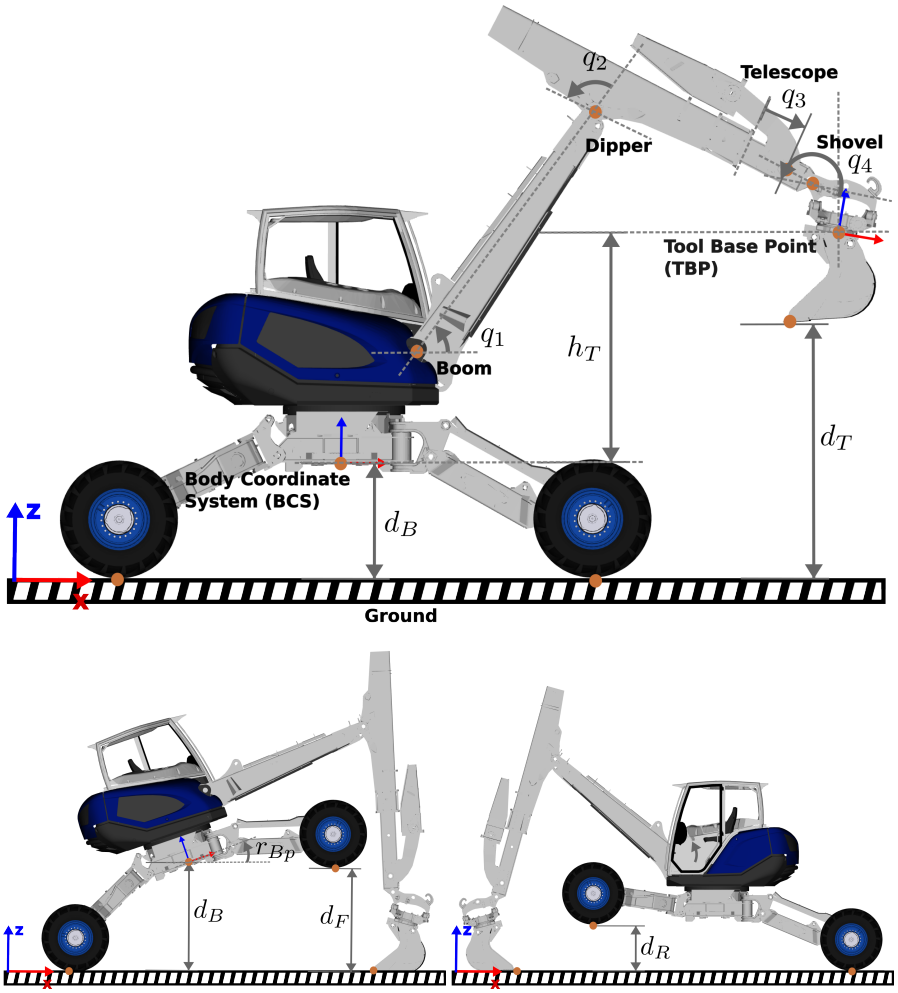


Figure 6.2: Side views of the learning setup for stepping.

the progress of the motion from Start Point (SP) to Goal Point (GP) moving along the X-axis of the WCS. The distance of the center of the obstacle from SP is given by η_O. The width of the obstacle Λ_w is a constant set at $0.3\,\mathrm{m}$. The height of the obstacle Λ_h is uniformly sampled within the range $0.1\,\mathrm{m}$ to $0.5\,\mathrm{m}$. The progress of the robot motion from SP to GP is represented by η_B. η_O and η_B are normalized with respect to the distance between SP and GP.

To model the interaction of the robot with the terrain, the distances of different components of the robot with the flat surface of the terrain is also represented. d_B and d_T are the minimum distances from the ground to robot base and tool respectively. d_F and d_R are the average distances of front wheels and rear wheels respectively to the ground.

In certain cases, the continuous goals are discretized, represented by the function $\mathcal{D}(\cdot)$. HRL controllers take in commands as goals which are then either directly or indirectly part of their observations space. In indirect cases, the commands and actual values are used to compute the error values which are then part of the observation space. During training the goals are sometimes randomly selected within a range with uniform sampling probability which is given by $U(l_l, l_u)$, where l_l and l_u

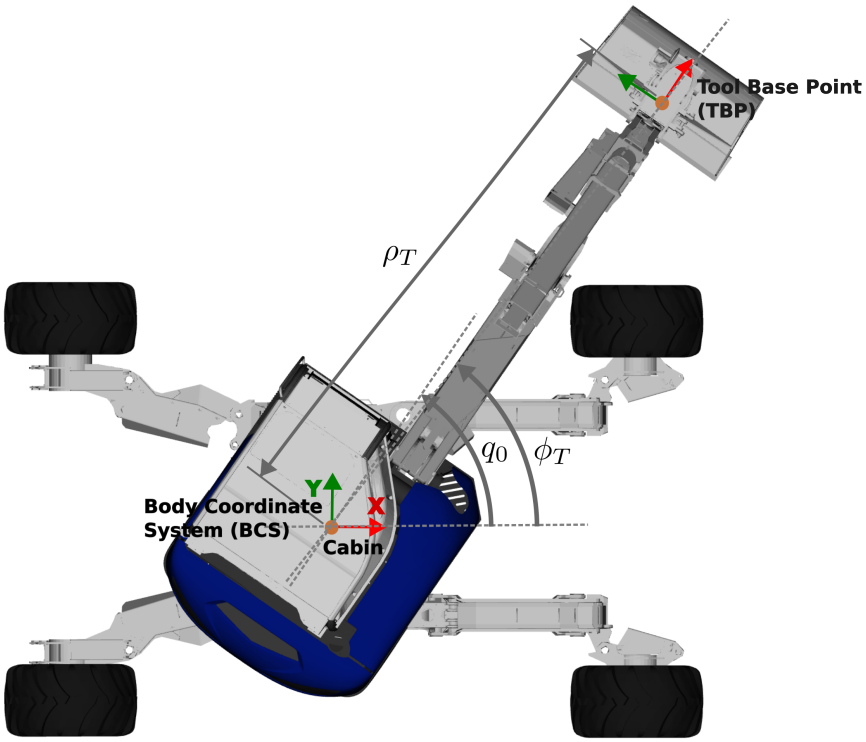


Figure 6.3: Top view of the learning setup for stepping.

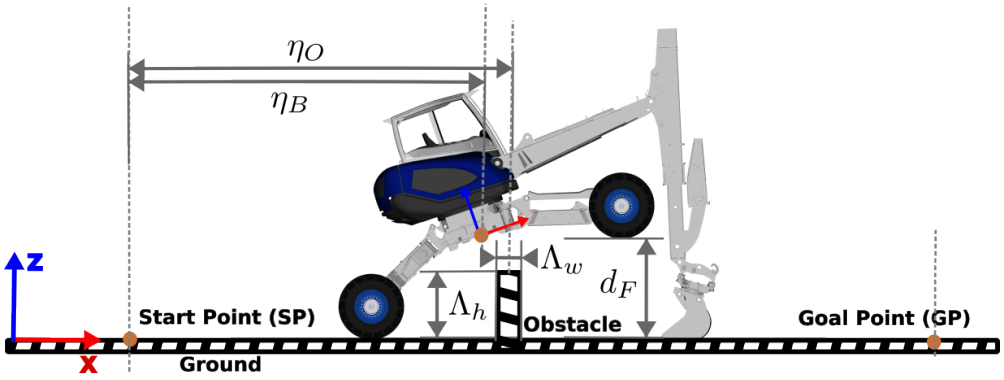
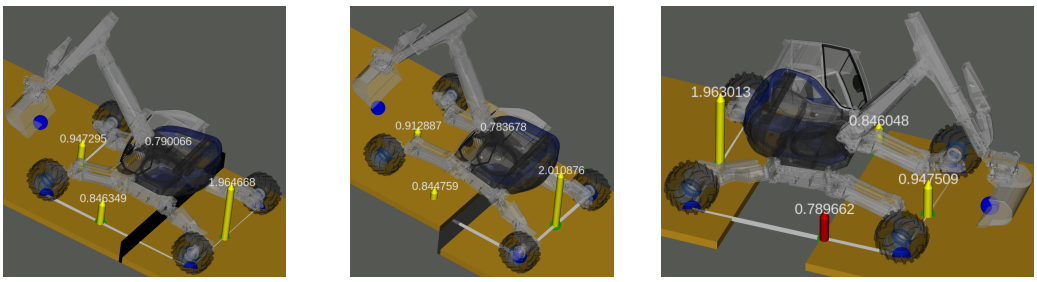


Figure 6.4: Terrain setup for stepping.

are the lower and upper limits respectively. The front stabilizer joints are commanded together with the same command $\dot{q}_F^c$. Similarly, $\dot{q}_R^c$ is the command for the rear joints. The commands and errors are represented with right superscripts c and e respectively.

The rewards are an important part of DRL, and their design will directly influence the performance of the controller. To make designing of the reward function easier, a generic function is implemented given by

$$\mathcal{R}(r, n, w, l_l, l_u) = w\mathcal{C}(\mathcal{N}(\mathcal{C}(r, l_l, l_u), l_l, l_u)^n, 0.0, 1.0), \tag{6.14}$$



(a) Obstacle (b) Step (c) Gap

Figure 6.5: Three different types of terrain environment with which the same controller was trained.

where r is reward factor, n is the power-factor, w is the weighing factor, l_l is the lower limit, and l_u is the upper limit. $\mathcal{C}$ is the clipping function, and $\mathcal{N}$ is the normalization function. If necessary, multiple reward functions are summed up to generate the total reward.

6.5 Hierarchical Controller Design

The objective of the stepping controller is to step over an obstacle without colliding. This needs a sequence of complex steps executed by the robot with each step having different requirements and constraints. Figure 6.6 depicts the hierarchical control structure that is developed for stepping. The setup has three blocks: HRL controllers, auxiliary controllers and joint controllers. The joint controller takes the velocity commands and controls the actuator to maintain the commanded velocity. The auxiliary controller block above this makes use of existing controllers to simplify the tasks for higher level controllers. The block above the auxiliary controllers is the HRL controllers which are trained using DRL.

The auxiliary controllers are manipulator Cartesian controller, drive control and trajectory follower. The manipulator Cartesian controller takes in Cartesian linear velocity commands for TBP in BCS and convert this into joint velocity commands which are sent to the robot joints. Weighted Damped Least-Square (WDLS) (Doty et al. 1993) is used to transform the velocity from the Cartesian space

$$\dot{X} = \begin{bmatrix} \dot{p}_x & \dot{p}_y & \dot{p}_z & \dot{r}_x & \dot{r}_y & \dot{r}_z \end{bmatrix}^T,$$

to the joint space

$$\dot{q} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 & \dot{d}_1 & \dot{\theta}_3 & \dot{\theta}_4 & \dot{\theta}_5 & \dot{\theta}_6 \end{bmatrix}^T.$$

The transformation is given by

$$\dot{q} = J_{\#}\dot{X}, \tag{6.15}$$

where the WDLS Jacobian inverse, $J_{\#}$, is given by

$$J_{\#} = M_q V_b \mathcal{I}_{dls}(D_b) U_b^T M_x,$$

where I_{dls} is the damped least-squared pseudo inverse function; $B = M_x J(q) M_q$ is the weighted Jacobian, where $J(q)$ is the Jacobian of the manipulator; $B = U_b D_b V_b^T$ is the Singular Value Decomposition (SVD) decomposition of B; M_q and M_x are the joint-space and task-space weighting matrices respectively.

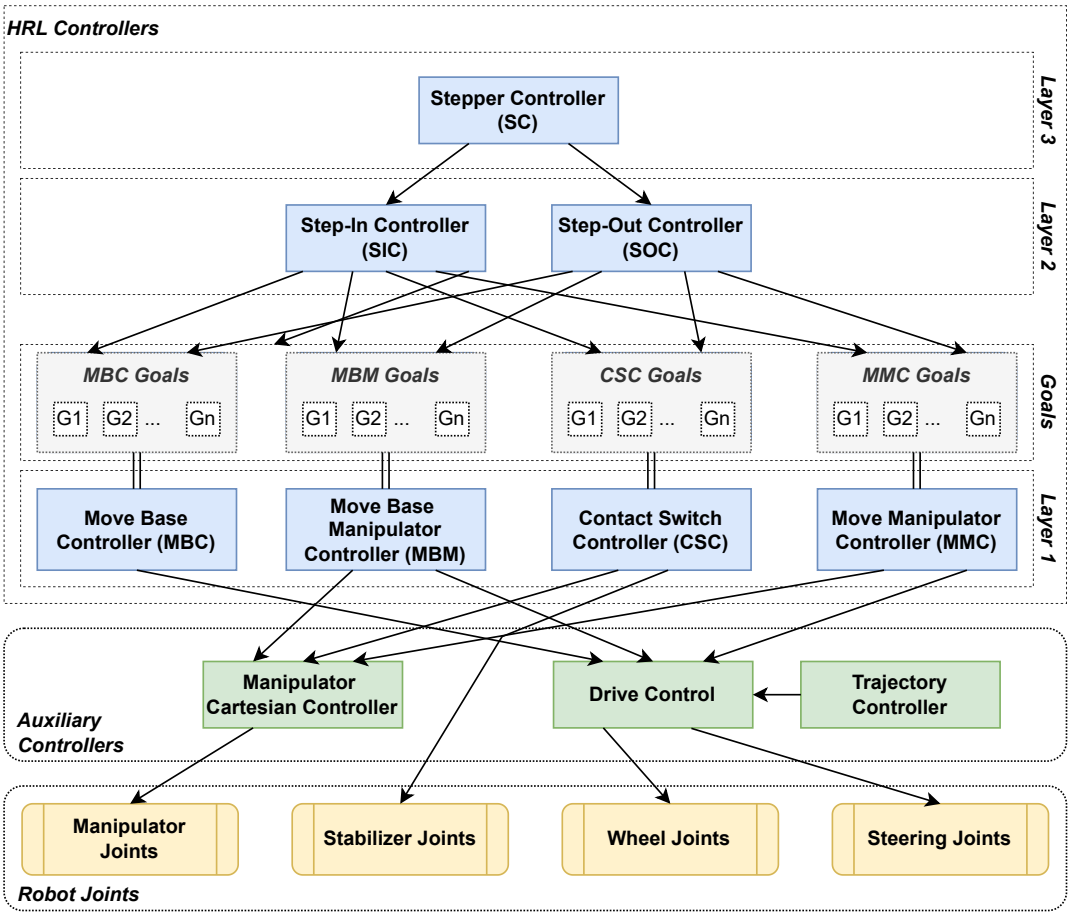


Figure 6.6: Hierarchical structure of the controllers with three layers of HRL controller with one, two and four subtasks each. The interaction between the second and third layer controllers are through discretized goals. Additionally, auxiliary and joint controller are also provided to execute the commands from the HRL controllers.

The rotation of the TBP is not relevant for stepping and hence is ignored. This is reflected in the weighting matrix of the task-space given by

$$M_x = diag(1.0, 1.0, 1.0, 0.0, 0.0, 0.0).$$

The motion of shovel, tilt and roto joints are ignored, thereby assigning weights in joint space to be

$$M_q = diag(1.0, 1.0, 1.0, 0.5, 0.0, 0.0, 0.0).$$

The weight corresponding to telescope joint is reduced to reduce its use compared to boom and dipper which are redundant joints.

The second auxiliary controller is the drive controller which transforms the incoming drive commands to wheel steering positions and wheel velocities. The drive consists of longitudinal velocity command ($\dot{v}^c_{Bl}$) and the angular velocity command ($\dot{\theta}^c_{Bl}$) for the robot. The angular velocity command is provided by trajectory follower, which follows a given trajectory for robot base. The longitudinal ve-

locity command is part of the actions of the HRL controllers Move Base and Manipulator Controller (MBM) and Move Base Controller (MBC). The third auxiliary controller is a trajectory follower which tries to follow a particular trajectory for the robot. For stepping the trajectory is always assumed to be straight lines.

In the input for the auxiliary controllers and joint commands come from the HRL controllers. As per the formalism of the HRL controller in Section 6.3, the primitive actions of the HRL are the inputs for either the auxiliary controller or direct joint commands. The subtasks of the first layer of HRL consists of four controllers: MBC, MBM, Contact Switch Controller (CSC) and Move Manipulator Controller (MMC). The output space of the lowest level is the primitive action space as given by

$$\Omega_{\omega^1} = \{\dot{\theta}_1^c, \dot{\theta}_2^c, \dot{\theta}_3^c, \dot{d}_1^c, \dot{\theta}_F^c, \dot{\theta}_R^c, \dot{v}_{Bl}^c, \dot{\rho}_T^c, \dot{h}_T^c\}.$$

These controllers output the primitive actions. The predefined discrete goals for the first layer are given by the controllers in the second layer. Step-In Controller (SIC) and Step-Out Controller (SOC) are part of second layer such that

$$\Omega_{\omega^2} = \{\Omega_{MBC}, \Omega_{MBM}, \Omega_{CSC}, \Omega_{MMC}\}.$$

Controllers in Ω_{ω^2} selects and give goals to Ω_{ω^1}. The top controller at third layer is the stepper controller such that

$$\Omega_{\omega^\Gamma} = \{\Omega_{SIC}, \Omega_{SOC}\}.$$

The combined subtask space of the HRL controller is

$$\Omega_H = \{\Omega_{MBC}, \Omega_{MBM}, \Omega_{CSC}, \Omega_{MMC}, \Omega_{SIC}, \Omega_{SOC}, \Omega_{SC}\}.$$

In Ω_H, all controllers except MBC and Stepper Controller (SC) are trained using DRL. SC and MBC are simple controllers, which do not need any DRL and hence implemented programmatically. SC does not have any goals, it just initiates either the SIC and SOC depending on the initiation conditions $I_{SIC} : \eta_B < \eta_O$ and $I_{SOC} : \eta_B \geq \eta_O$ respectively. MBM gives a constant value for $\dot{v}_{Bl}^c = 0.5$ until the commanded translation is reached.

The HRL controller is manually designed using domain knowledge and the controllers are trained for first layer followed by the second layer, hence a bottom-to-top approach. The training of each controller needs separate simulation environments as well as different DRL algorithms with different parameters. The following subsections give the details of HRL controllers which are trained using DRL and the corresponding training results.

6.5.1 Contact Switch Controller

The objective of CSC is to establish contact or maintain a predefined distance of the limb end-effectors and the robot body with respect to the ground. The end-effectors for the limbs are the wheels and manipulator tool. The robot starts at a random initial pose and tries to reach a pose that satisfies the reference distances (goals), which are indirectly included in the observation states. Masked-PPO algorithm is used to train the controller as the actions are discrete and has action mask. The parameters used for training are given in Table 6.6. The simulation environment setup for the training was similar to the environment except the obstacle height which is set to $h_O = 0.0$. The design of CSC is summarized in Table 6.1.

Table 6.1: Summary of the controller design for CSC.

Goals (g_{CSC})	$d_B^c \sim U(0.5, 1.0)$ $d_F^c \sim \bar{U}(-1.0, +1.0)$ $d_R^c \sim \bar{U}(-1.0, +1.0)$ $d_T^c \sim \bar{U}(-1.0, +2.0)$	Body and end-effector distance to ground references
Observations (S_{CSC})	$q_i \forall i \in \{FL, FR, RL, RR\}$ $\dot{q}_i \forall i \in \{FL, FR, RL, RR\}$ h_T, ρ_T, ϕ_T $d_B^e, d_F^e, d_R^e, d_T^e$ h_{ne} r_{Bp}	Actual stabilizer joint positions Actual stabilizer joint velocities Actual TBP cylindrical coordinates Actual end-effector distances to ground NESM value Pitch rotation of robot body
Actions (A_{CSC})	$\mathcal{D}\left(\dot{\theta}_F^c\right) \rightarrow \{-0.4, 0.0, 0.4\}$ $\mathcal{D}\left(\dot{\theta}_R^c\right) \rightarrow \{-0.4, 0.0, 0.4\}$ $\mathcal{D}\left(\dot{h}_T^r\right) \rightarrow \{-0.4, 0.0, 0.4\}$	Discretized command velocity of front stabilizers Discretized command velocity of rear stabilizers Discretized cylindrical height velocity command of TBP
Reward (r_{CSC})	$\mathcal{R}(\|d_B^e\|, 2.0, -0.2, 0.0, 1.0)+$ $\mathcal{R}(\|d_F^e\|, 2.0, -0.2, 0.0, 1.0)+$ $\mathcal{R}(\|d_T^e\|, 2.0, -0.2, 0.0, 4.0)+$ $\mathcal{R}(\|d_B^e\|, 2.0, -0.2, 0.0, 1.5)+$ $\mathcal{R}(\|h_{ne}\|, 1.0, -0.1, 0.0, 0.3)+$ $\mathcal{R}(\|r_{Bp}\|, 1.0, -0.1, 0.0, 0.2)$	Body and end-effector distance to ground errors Stability margin Robot body pitch angle error
Successful termination (β_{CSC}^s)	$(\|d_B^e\| < 0.1) \wedge$ $(\|d_F^e\| < 0.1) \wedge$ $(\|d_R^e\| < 0.1) \wedge$ $(\|r_{Bp}\| < 0.1)$	Distance to ground errors and pitch error in range
Failed termination (β_{CSC}^f)	$t_\omega > 70.0$	Controller episode time limit
Action masks	$\mathcal{M}_L(\dot{q}_F^c, \theta_F, -0.83, 0.29)$ $\mathcal{M}_L(\dot{q}_R^c, \theta_R, -0.77, 0.27)$ $\mathcal{M}_L\left(\dot{h}_T^c, h_T, -2.0, 3.0\right)$	Limit masks for all actions

The reference distances to the ground for the body, front-wheels, rear-wheels and manipulator tool are denoted by d_B^c, d_F^c, d_R^c and d_T^c respectively. The reference distances are the goals which are provided to the controller. d_B^c is sampled uniformly from a predefined range. The rest of the goals are sampled using $\bar{U}$ which is defined by

$$\bar{U}(l_l, l_u) = max(U(l_l, l_u), 0.0) \tag{6.16}$$

to make sure that the non-negative reference values are set to 0.0, which represents contact with the ground. This makes sure that there are similar chances of generating contact and non-contact references. There is also a validity check which makes sure that the goals are reachable. The difference

between the reference and the actual values are the errors denoted by d_B^e, d_F^e, d_R^e and d_T^e respectively. The error values are part of the observation.

Other than the error values, the observations consist of stabilizer joint states, cylindrical TBP position, NESM margin and body pitch. The joint states and TBP positions are relevant to know the current state of the robot that are relevant for the controller. The stability margin h_{ne} and the body pitch angles are part of the reward function.

The actions for CSC are the stabilizer joint velocity commands and the vertical velocity of the TBP. In general the cylindrical velocity command (h_T^c, ρ_T^c, ϕ_T^c) are converted to joint commands using the auxiliary controllers by using Equation (6.15)

$$\dot{q^c} = J_{\#}\chi\Big(\begin{bmatrix} h_T^c & \rho_T^c & \phi_T^c \end{bmatrix}^T\Big), \tag{6.17}$$

where χ is the transformation from cylindrical velocities to Cartesian velocities.

The main terms in the reward function are the error values which ensures that the goals are reached. The other two factors make sure that good stability is maintained as well as the robot does not have too high body pitch angle. All the factors are transformed using the function defined in Equation (6.14) and added up to form the total reward.

The controller is initiated upon receiving a new goal and can terminate either upon success or failure. Successful termination occurs when all reference errors fall within the specified tolerances, and the body pitch angle is also within a defined threshold. Conversely, the episode termination is considered a failure if the controller's episode duration (t_ω) is exceeded without reaching the goal.

The use of discrete actions makes it possible to include actions masks. In CSC action masks are used only for reaching limits. If a joint or coordinate reaches a limit, the action which takes it further into the limit is masked out. The mask limit function is given by

$$\mathcal{M}_L(A_v, b, l_l, l_u) = \begin{cases} \{x \in A_v \mid x \geq 0.0\}, & \text{if } b < l_l \\ \{x \in A_v \mid x \leq 0.0\}, & \text{if } b > l_u \\ A_v, & \text{otherwise} \end{cases}, \tag{6.18}$$

where A_v is the set of all actions with velocity commands, b is the corresponding positions, l_l and l_u are the lower and upper limits of b.

The discounted return during the training of π_{CSC} is shown in Figure 6.7. The training was performed for 2000 episodes and converged by around 600 episodes.

6.5.2 Move Base and Manipulator Controller

One of the main features of ARTER is its ability to use the manipulator for locomotion. To overcome obstacles, it is necessary to move the robot body using both wheels and manipulator simultaneously. MBM achieves this by the synchronized movement of the manipulator and the wheels to ensure that the robot body moves in the intended direction and no unnecessary internal stress is developed within the mechanical elements of the vehicle. The design of MBM is summarized in Table 6.2.

The goal for MBM is the translational offset command (d_{MBM}^c) which is the distance that the robot body should travel using both wheels and manipulator. During training, the goal is uniformly sampled

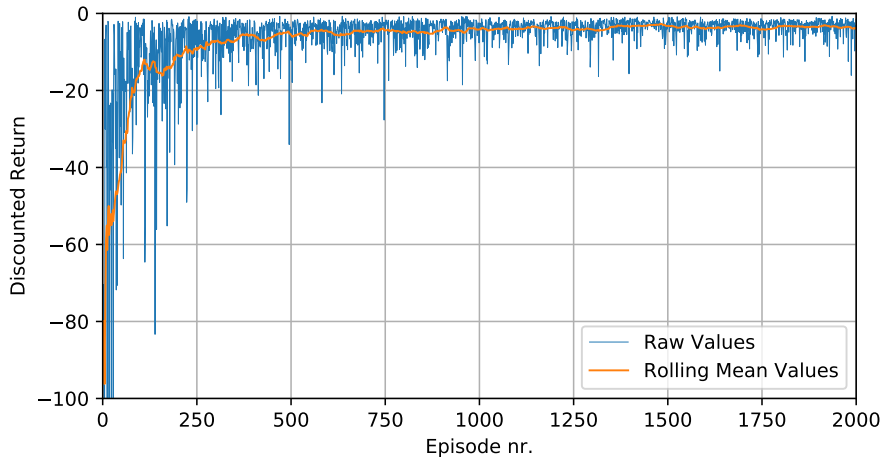


Figure 6.7: Discounted return including the raw values as well as the rolling mean values during training of CSC.

Table 6.2: Summary of the controller design for MBM.

Goals (g_{MBM})	$d^c_{MBM} \sim U(0.5, 3.0)$	MBM translational offset command
Observations (S_{MBM})	d^e_{MBM}	MBM translational offset error
	d_T	Minimum distance of tool to ground
	v_{Wl}	Horizontal velocity of the robot computed based on wheel angular velocities
	ρ_T, ϕ_T	Actual cylindrical radius and angle of TBP
	$\dot{\rho}_T$	Actual Cartesian velocity of the TBP about X-axis of the robot
	s_l	Longitudinal slip between end-effectors
Actions (A_{MBM})	$\mathcal{L}(\dot{v}^c_{Bl}, 0.0, 0.5)$	Commanded horizontal velocity of robot
	$\mathcal{L}(\dot{\rho}^c_T, -1.0, +1.0)$	Commanded radial velocity for TBP
Rewards (r_{MBM})	$\mathcal{R}(\|d^e_{MBM}\|, 2.0, -0.5, 0.0, 1.0)+$	MBM-offset error
	$\mathcal{R}(\|s_l\|, 2.0, -0.5, 0.0, 1.0)$	Longitudinal slip
Successful termination (β^s_{MBM})	$(\|d^e_{MBM}\| < 0.1)$	MBM-offset distance reached
Unsuccessful termination (β^f_{MBM})	$(t_\omega > 100.0) \vee$	Controller episode time limit
	$(d_T > 0.1)$	Tool has lost contact with ground

from a predefined range. The actual value (d_{MBM}) is determined by the distance travelled by the robot, starting from the position at the start of the episode, along the X-axis.

The observation states include the translational offset error (d^e_{MBM}) which incorporates both the reference and actual values of translational offset. The TBP cylindrical radius and angle of gives information regarding the current pose of the manipulator. The longitudinal velocities due to wheels

(v_{Wl}) and due to manipulator (v_{Tx}) are both part of the observations, along with slip. Keeping slip to zero ensure that the manipulator and wheels work synchronously. In general, slip is the relative motion between the end-effector and the actual motion of the vehicle. The longitudinal slip is defined by

$$s = \frac{v_B - v_E}{v_B}, \tag{6.19}$$

where v_B is the longitudinal velocity of the vehicle, v_E is the longitudinal velocity due to the end-effector. The longitudinal axis is the X-axis in BCS.

In case of MBM, the motion is caused by both wheels and the manipulator end-effector. Apart from the movement of both wheels and manipulator tool, the ground interaction of these end-effectors also affects the slip. This is very subjective of the terrain type and the interaction properties. Hence, this work considers only the relative slip between the two end-effectors which is given by

$$s_l = \frac{v_{Wl} - v_{Tl}}{v_{Wl}}, \tag{6.20}$$

where v_{Wl} is the expected longitudinal velocity of the vehicle based on wheel velocities, and v_{Tl} is the longitudinal velocity of the TBP, both expressed in BCS.

The actions for the controller are the horizontal velocity reference for drive control and the cylindrical radial velocity for the TBP which goes as input to the manipulator Cartesian controller. Both these values are continuous and are within predefined ranges.

The objective of reaching a translational offset without slip is reflected in the reward function which contains both the translational offset error (d^e_{MBM}) and slip ratio (s_l) as contributing factors.

Termination of the controller is successful if the translational offset is reached within a specified tolerance. On the other hand the controller fails if the episode time limit is reached or the tool looses contact with the ground.

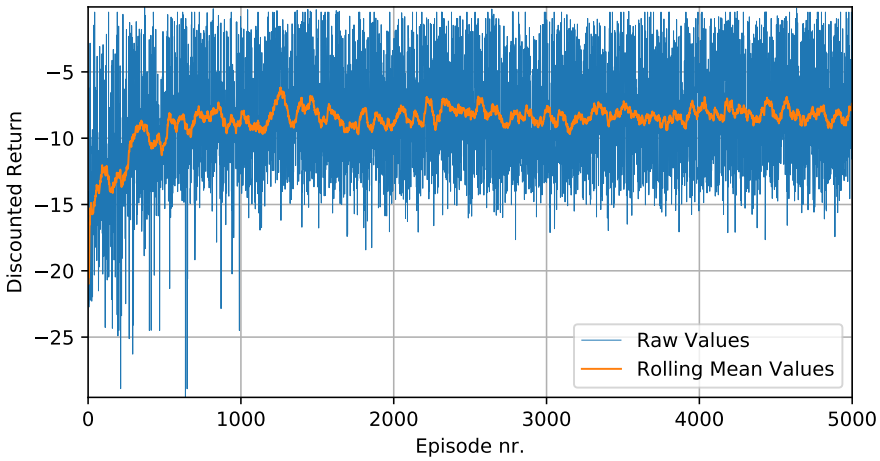


Figure 6.8: Discounted return including the raw values as well as the rolling mean values during training of MBM.

The actions are continuous and hence SAC is used as the DRL algorithm and without any action masks. The training of the controller was performed for 5000 episodes and the controller converges within 1000 episodes as shown in Figure 6.8.

6.5.3 Move Manipulator Controller

Table 6.3: Summary of the controller design for MMC.

Goals (g_{MMC})	$h_T^c \sim U(-3.0, 4.0)$ $\rho_T^c \sim U(2.0, 6.0)$ $\phi_T^c \sim U(-0.1, 3.3)$	References in cylindrical coordinates
Observations (S_{MMC})	$q_i \forall i \in \{FL, FR, RL, RR\}$ h_T, ρ_T, ϕ_T $h_T^c, \rho_T^c, \phi_T^c$ $h_T^e, \rho_T^e, \phi_T^e$ $q_i \forall i \in \{FL, FR, RL, RR\}$ d_{mc}	Actual manipulator joint positions Actual TBP cylindrical coordinates Reference TBP cylindrical coordinates Error TBP cylindrical coordinates Actual stabilizer joint positions Minimum distance of manipulator link to collision with other robot links
Actions (A_{MMC})	$\mathcal{L}(\dot{q}_0^c, -0.5, 0.5)$ $\mathcal{L}(\dot{q}_1^c, -1.0, 1.0)$ $\mathcal{L}(\dot{q}_2^c, -1.0, 1.0)$ $\mathcal{L}(\dot{q}_3^c, -1.0, 1.0)$	Velocity command for cabin joint Velocity command for boom joint Velocity command for dipper joint Velocity command for telescope joint
Rewards (r_{MMC})	$\mathcal{R}(\|h_T^e\|, 2.0, -0.1, 0.0, 10.0)$ $\mathcal{R}(\|\rho_T^e\|, 2.0, -0.1, 0.0, 10.0)$ $\mathcal{R}(\|\phi_T^e\|, 2.0, -0.1, 0.0, 4.0)$ $\mathcal{R}(d_{mc}, 2.0, -0.7, 0.0, 2.0)$	Cylindrical coordinate errors and manipulator collision
Successful termination (β_{MMC}^s)	$(\|h_T^e\| < 0.1) \wedge$ $(\|\rho_T^e\| < 0.2) \wedge$ $(\|\phi_T^e\| < 0.1)$	Errors in cylindrical coordinates within limit
Unsuccessful termination (β_{MMC}^f)	$t_\omega > 20.0$	Controller episode time limit

The main objective of the MMC is to move the manipulator from one pose to another without self collision. This necessary to move the manipulator to a pose where the other controllers can perform their tasks. Traditionally, this is performed by the manipulator motion planners. In this work, due to the kinematics of the manipulator, it is not necessary to include such complex solutions and hence MMC is developed and trained. The design of MMC is summarized in Table 6.3.

The goals for MMC are the reference cylindrical coordinates of TBP (h_T^c, ρ_T^c, ϕ_T^c), which are also sampled uniformly from a predefined range. The observation contains the actual values, reference values (goals) and error of the cylindrical coordinates of TBP. The current joint positions of the manipulator as well as the stabilizer joints of the legs are also included in the observation.

The reward function, in addition to the error in goals, also has a term to avoid collision. This term penalizes the controller if there is collision with the manipulator or if it is too close. d_{mc} is the closest distance of any link in manipulator to the rest of the robot links. The termination is considered suc-

cessful if the goal is reached within a predefined tolerance and time limit; otherwise, it is considered a failed termination.

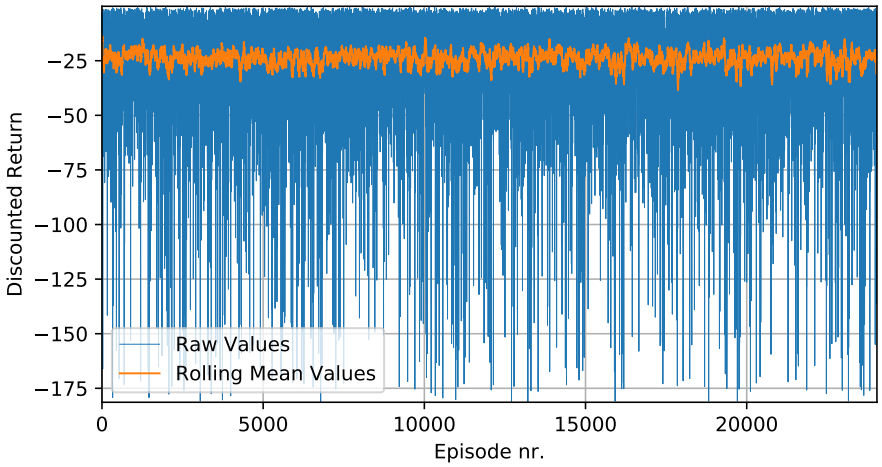


Figure 6.9: Discounted return including the raw values as well as the rolling mean values during training of MMC

The policy for MMC is pre-trained using a prior function which produces a proportional velocity command based on the error in cylindrical coordinates. This command is given as input to manipulator Cartesian controller. The pre-trained policy is then trained to obtain the final controller. The performance of the final trained controller did not vary from the prior since most of the provided prior trajectories from start to goal did not have any collision. The reward progress during the training is shown in Figure 6.9. Nevertheless, the provided prior already gave satisfactory performance necessary for the higher level controllers.

6.5.4 Step-In Controller

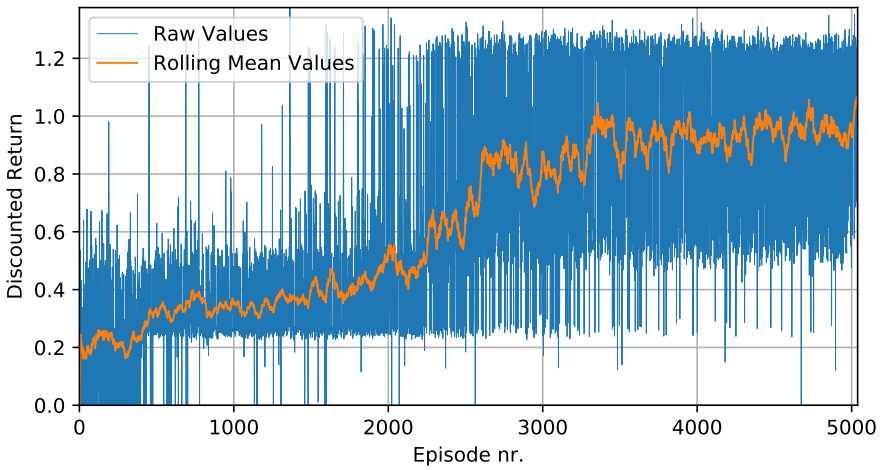


Figure 6.10: The evolution of the reward during training of SIC where multiple jump in rewards are clearly visible which shows the instances where the controller learns the goals for the subtasks. The training of the controller took around 4000 episodes.

Table 6.4: Summary of the controller design for SIC.

Observations (S_{SIC})	Λ_h, Λ_w	Obstacle height and width
	η_B, η_O	Normalized progress for robot and obstacle relative to distance from SP to GP
	n_{SEQ}	Normalized progress of sequence
	δ_B	Difference in normalized progress before and after the execution of a subtask
	$q_i \forall i \in \{FL, FR, RL, RR\}$	Actual stabilizer joint positions
	h_T, ρ_T, ϕ_T	Actual TBP cylindrical coordinates
	$d^e_B, d^e_F, d^e_R, d^e_T$	Actual end-effector distances to ground
Actions (A_{SIC})	$\mathcal{D}(g_{MBM}) \rightarrow \{NaN, G_{MBM}\}$	Discretized goals for MBM
	$\mathcal{D}(g_{MBC}) \rightarrow \{NaN, G_{MBC}\}$	Discretized goals for MBC
	$\mathcal{D}(g_{CSC}) \rightarrow \{NaN, G_{CSC}\}$	Discretized goals for CSC
	$\mathcal{D}(g_{MMC}) \rightarrow \{NaN, G_{MMC}\}$	Discretized goals for MMC
Rewards (r_{SIC})	$\mathcal{R}(\lVert\delta_B\rVert, 1.0, 1.0, 0.0, 1.0)$	Progress made on execution of each subtask
Successful termination (β^s_{SIC})	$n_B \geq n_O$	Normalized progress reached the obstacle
Unsuccessful termination (β^f_{SIC})	$(c_{RO} = 0.0) \vee$	Robot collide with obstacle
	$(n_{SEQ} = 1.0) \vee$	Sequence finished
	$(\neg \mathcal{S}(\omega_1))$	Failed subtask
Actions masks	$\mathcal{M}_{S,SIC}(n_{SEQ}) \rightarrow \{\}$	Mask for trying a predefined sequence
	$\mathcal{M}_{CSC}(\phi_T)$	Mask for eliminating invalid CSC-goals

SIC is a second layer controller which initiates the controllers in the first layer with appropriate goals in a sequence to reach the obstacle. SIC is initiated according to specific condition by SC. The objective of SIC is to reach the middle of the obstacle from the start position. The design of the controller is summarized in Table 6.4. The reward progression during training with Masked-PPO is shown in Figure 6.10.

The observations of SIC contains terrain information including obstacle height and distance of obstacle from start, which were not provided to any of the controllers in the first layer. Progress being made between start and the goal as well as the progress made by a subtask is part of the observation. The joint states of the stabilizer joints, cylindrical position of the TBP, distance of end-effectors to ground are also included in the observation to make the controller aware of the current state of the robot. Additionally, the index of the subtask sequence which is the number of subtasks already executed in the current episode is also included.

The actions are the goals to the controllers CSC, MBC, MBM and MMC in the first layer. Even if these controllers are trained with continuous goals, SIC uses 12 discretized goals per controller and NaN, which is to be used for disabling the controller. The reward is computed from the progress made by each subtask. The controller terminates successfully if the robot progresses to the obstacle. The controller fails if it collides with the obstacle which is determined by the distance between robot links and obstacle (c_{RO}). Exceeding the threshold for the index of sequence or any of the subtask failing will fail SIC. $\mathcal{S}(\omega_1)$ returns $True$ on success of subtask ω_1. IAM is used by SIC to incorporate more

domain knowledge. There are two masking functions, one to predefine the sequence and a second one to give only valid commands to CSC. The sequence is predefined by

$$\mathcal{M}_{S,SIC}(n_S) = \begin{cases} G_{MBM}, & \text{if } n_S \in \{5\} \\ G_{MBC}, & \text{if } n_S \in \{2,7\} \\ \mathcal{M}_{CSC}(G_{CSC}, \phi_T), & \text{if } n_S \in \{1,4,6\} \\ G_{MMC}, & \text{if } n_S \in \{3\} \end{cases}, \tag{6.21}$$

where $\mathcal{M}_{CSC}(G_{CSC}, \phi_T)$ masks invalid actions in CSC such that only valid goals are given, depending on the direction of the manipulator.

6.5.5 Step-Out Controller

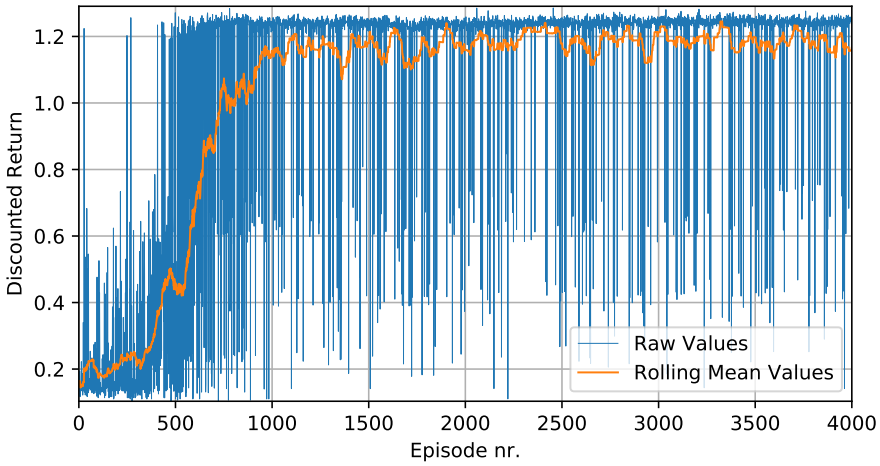


Figure 6.11: Discounted return including the raw values as well as the rolling mean values during training of SOC. During training the rewards are maximized by 1000 episodes which is faster than SIC, even though the number of steps in the sequence are the same.

SOC has an identical design except the initiation condition ($n_B \geq n_O$), termination condition ($n_B \geq 1.0$) and the predefined sequences (MBC, MMC, CSC, MBM, CSC, MMC and MBM). During training the simulation of SOC is initialized with the pose of the robot at the end of successful runs of SIC.

The objective of SOC is to move from the middle of the obstacle to the goal position of stepping. The design is identical to SOC except the initiation condition ($n_B \geq n_O$), termination condition ($n_B \geq 1.0$) and the predefined sequences. The subtask sequence is predefined to be

$$\mathcal{M}_{S,SOC}(n_S) = \begin{cases} G_{MBM}, & \text{if } n_S \in \{4\} \\ G_{MBC}, & \text{if } n_S \in \{1,7\} \\ \mathcal{M}_{CSC}(G_{CSC}, \phi_T), & \text{if } n_S \in \{3,5\} \\ G_{MMC}, & \text{if } n_S \in \{2,6\} \end{cases}. \tag{6.22}$$

Table 6.5: SAC parameters for MBM and MMC. The controllers were tuned manually and differences can be seen in learning rate, discount factor and soft update coefficient. State dependent exploration is active only for MBM.

	MBM	MMC
Policy network type	MLP	MLP
Actor network size	[400, 300]	[400, 300]
Critic network size	[400, 300]	[400, 300]
Learning rate (η)	7.3×10^{-4}	3×10^{-4}
Discount factor (γ)	0.98	0.999
Soft update coefficient (τ)	0.02	0.005
Temperature parameter (α)	Auto	Auto
Train frequency	256	256
Gradient steps	64	64
Initial value of log standard deviation	-3	-3
gSDE	Active	Inactive
Number of critics	2	2

Table 6.6: Masked-PPO parameters for CSC and SIC/SOC controllers which are tuned by hand. CSC has bigger network size and lower learning rate. There are some minor differences in the discount factor, clipping parameter and entropy coefficient.

	CSC	SIC	SOC
Policy network type	MLP	MLP	MLP
Actor network size	[256, 256]	[256]	[256]
Critic network size	[256, 256]	[256]	[256]
Network activation function	ReLU	ReLU	ReLU
Learning rate (η)	1.0×10^{-5}	5.0×10^{-4}	5.0×10^{-4}
Discount factor (γ)	0.999	0.99	0.98
GAE trade-off bias vs variance (λ_{GAE})	0.98	0.98	0.98
Clipping parameter	0.4	0.1	0.1
Entropy coefficient	0.0	0.01	0.01
Value function coefficient	0.4	0.4	0.5
Maximum value for the gradient clipping	0.5	0.5	0.5

6.6 Implementation

All the training and testing is performed in simulation which is based on Bullet (Coumans 2015). The MCS modules described in Chapter 4 uses ROS as the robot framework. The DRL algorithms SAC and masked-PPO uses implementation from Raffin et al. 2019. The training is performed using a deep learning computer with an AMD Ryzen Threadripper 3970X 32-Core processor and an Nvidia RTX A6000 GPU.

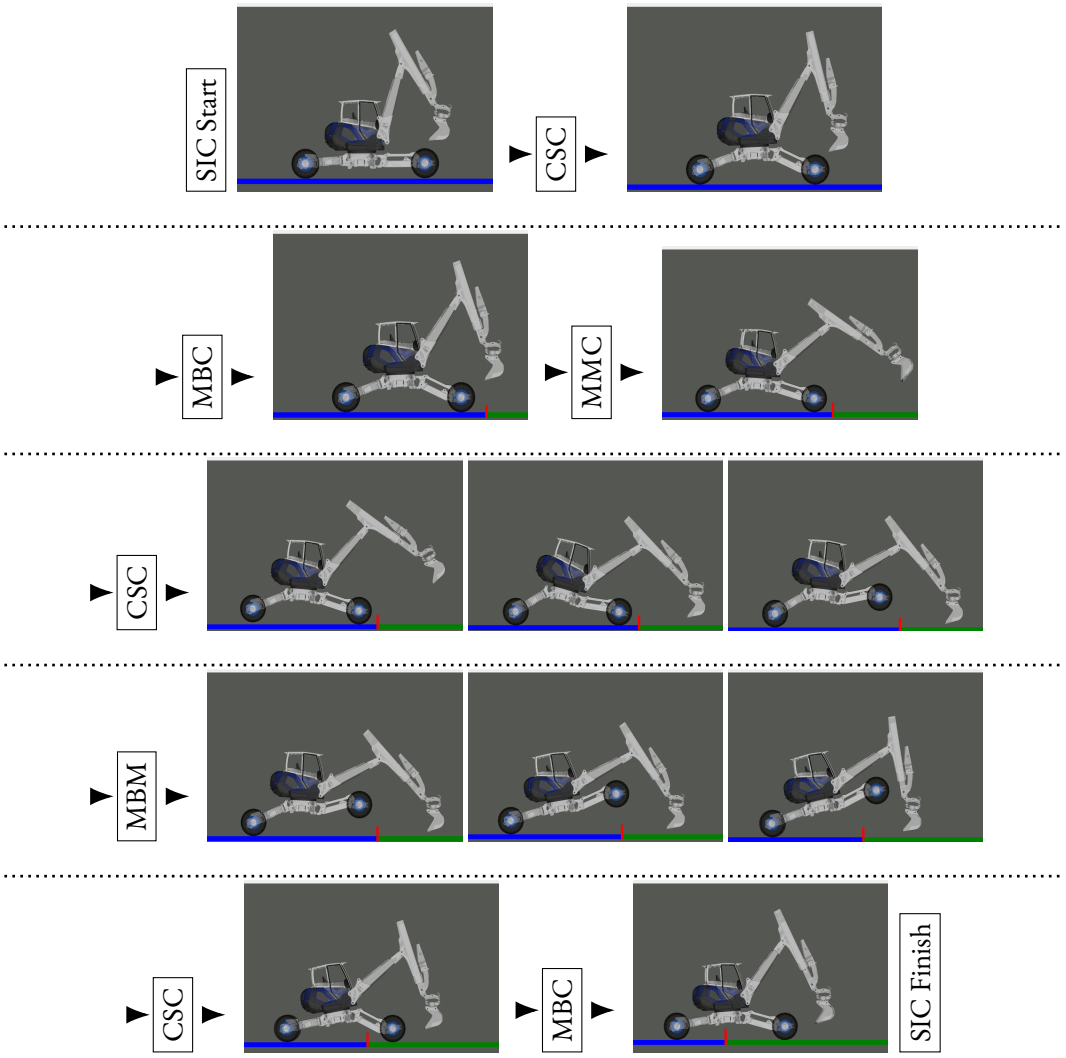


Figure 6.12: An example stepping sequences of the SIC, depicted row-wise and from left to right. The subtasks that are started are indicated between the screenshots.

6.7 Controller Evaluation

The controller is successfully trained [1] to traverse the three different types of stepping terrains. An illustrative example of a stepping sequence for the obstacle terrain is provided in Figures 6.12 and 6.13 for SIC and SOC respectively. From the initial pose, the CSC is initiated with a goal that moves both the front and rear wheels downwards, thereby increasing the clearance between the body and the ground. Subsequently, the MBC directs the robot to move until the front wheels retains a slight offset relative to the obstacle, followed by MMC which moves the manipulator outwards in order to reach the initial pose for CSC. In the next step, the CSC maneuvers the wheels upward while the TCP is moved downwards, thereby elevating the front wheels to a height that surpasses that of the obstacle, while the tool maintains contact with the ground. Subsequently, the MBM is executed in a synchronous manner with the TCP and wheels, facilitating forward motion without slippage. Once the front wheels

[1] The video of the resulting stepping motions are shown in this link: https://drive.google.com/file/d/1X0VHZDuECn-OE3vf3uMPn8UUAfVch-0F/view?usp=sharing&t=138.

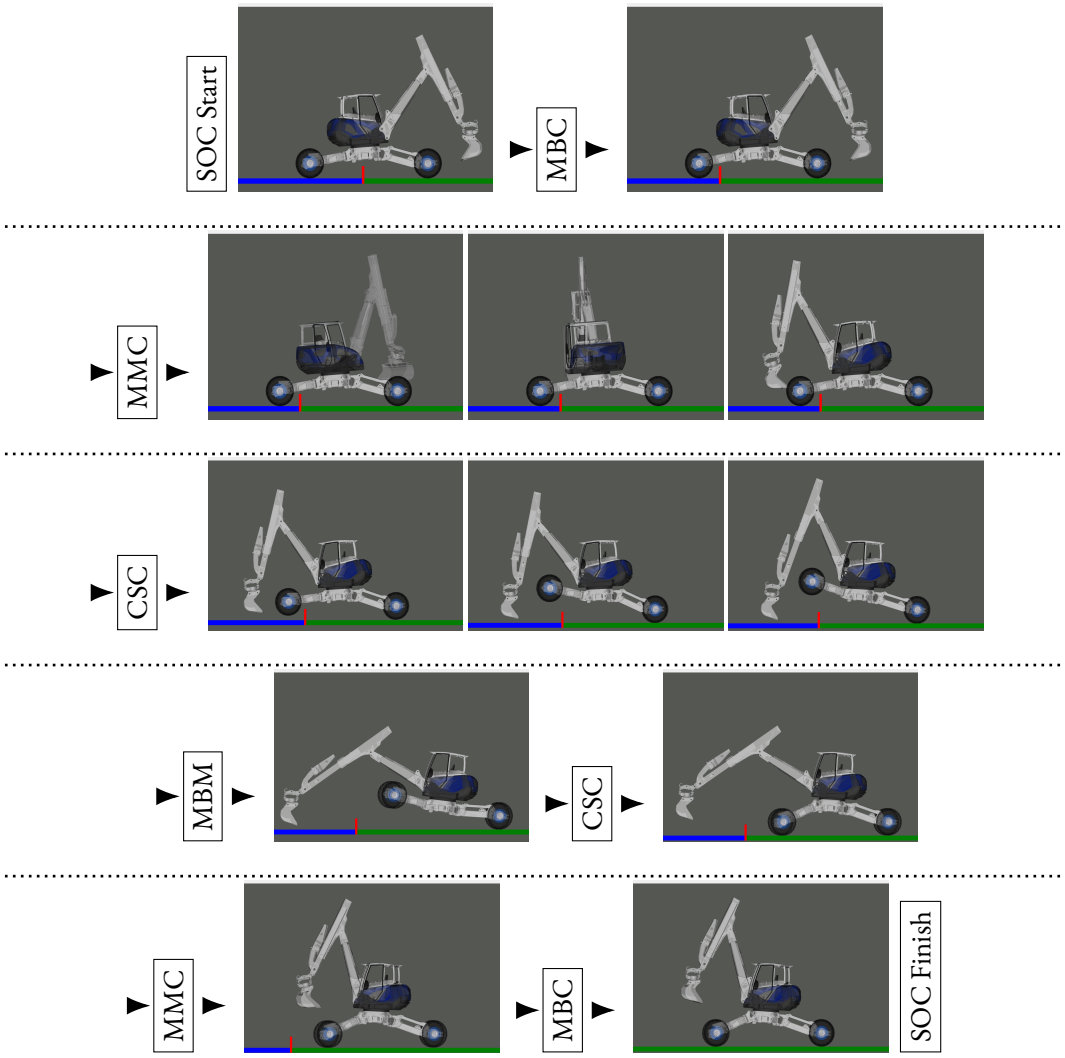


Figure 6.13: An example stepping sequences of the SOC, depicted row-wise and from left to right. The subtasks that are started are indicated between the screenshots.

have traversed the obstacle with sufficient clearance, the CSC initiates a change in contact such that the robot reverts to the drive pose. Upon completion of SIC, MBC maneuvers the robot until the body has traversed the obstacle.

The SOC follows similar steps but the main difference being change in the direction of the manipulator and executing MBM with different pose of the manipulator. Finally, SOC terminates the operation with MBC, which drives the platform until the goal is reached. The SOC begins with MBC, which propels the robot until the wheels are in proximity to the obstacle. Subsequently, the manipulator is switched to a pose in which the TCP is situated on the rear side of the robot. At this point, CSC lifts the rear legs with the tool and the front wheels are in contact with the ground. The MBM is executed until the rear wheels cross the obstacle. MMC then moves the manipulator to a drive pose. Finally, SOC terminates the operation with MBC, which drives the platform until the goal is reached.

In order to gain a deeper comprehension of the sequences that were acquired by the controller, the goal sequences for SIC and SOC that were successfully executed during the evaluation phase have

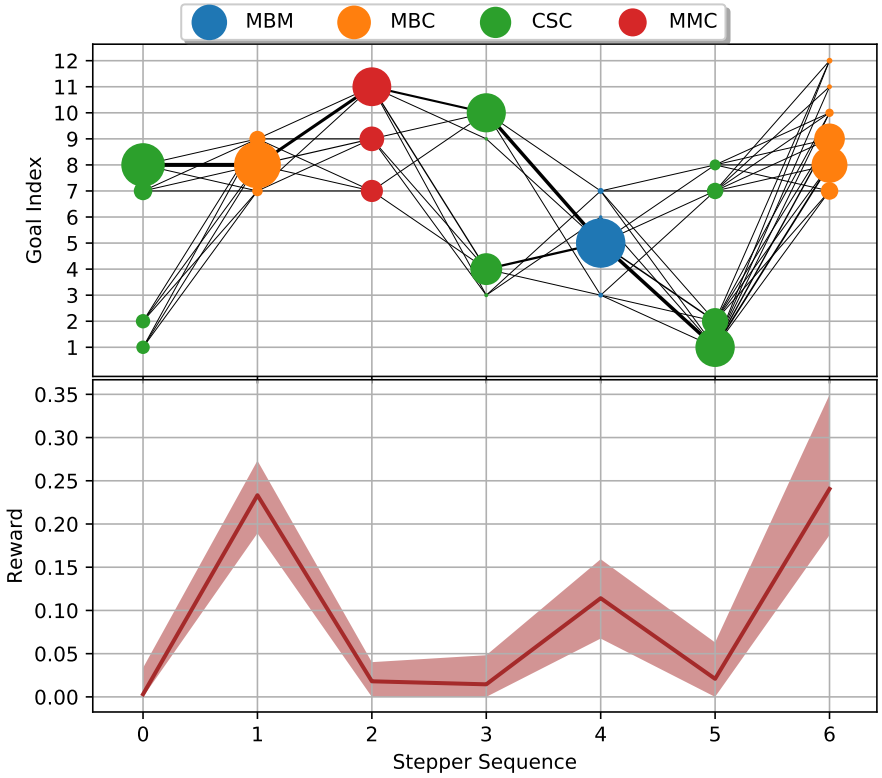


Figure 6.14: Subtask goal sequence and rewards for SIC during each step for all the successful runs during evaluation. The upper plot illustrates the goals utilized by each subtask, with the subtasks differentiated by color and the radius of the marker being proportional to the frequency of each goal. The bottom plot depicts the reward obtained for each subtask in the sequence, along with the variance.

been illustrated in Figures 6.14 and 6.15, respectively. The upper plots illustrate the goals utilized by each subtask, with the subtasks differentiated by color and the radius of the marker being proportional to the frequency of each goal. The lines connecting the markers demonstrate the succession of goals chosen. The lower plot depicts the reward obtained for each subtask in the sequence, along with the variance.

The height of the obstacle is the main external factor that varies for each episode. The goals which are mostly influenced by the varying obstacle height show more even distribution. The goals for the subtasks MMC, CSC and CSC at sequence indices 2, 3 and 5 respectively shows this effect. The obstacle height has a bigger influence on selection of goals in SIC than in SOC.

6.8 Summary and Discussions

A HRL based controller with three level of subtasks is developed to address the complex task of stepping for a walking excavator by leveraging domain knowledge. Furthermore, the controller employs IAM to mask actions that are detrimental or to rectify the selection sequence of the subtask. The training of the subtasks is distilled to the selection of an appropriate goal for a given subtask. The integration of HRL and IAM results in the development of a controller that is capable of successfully traversing three different types of obstacles within a simulated environment. The resulting motion and

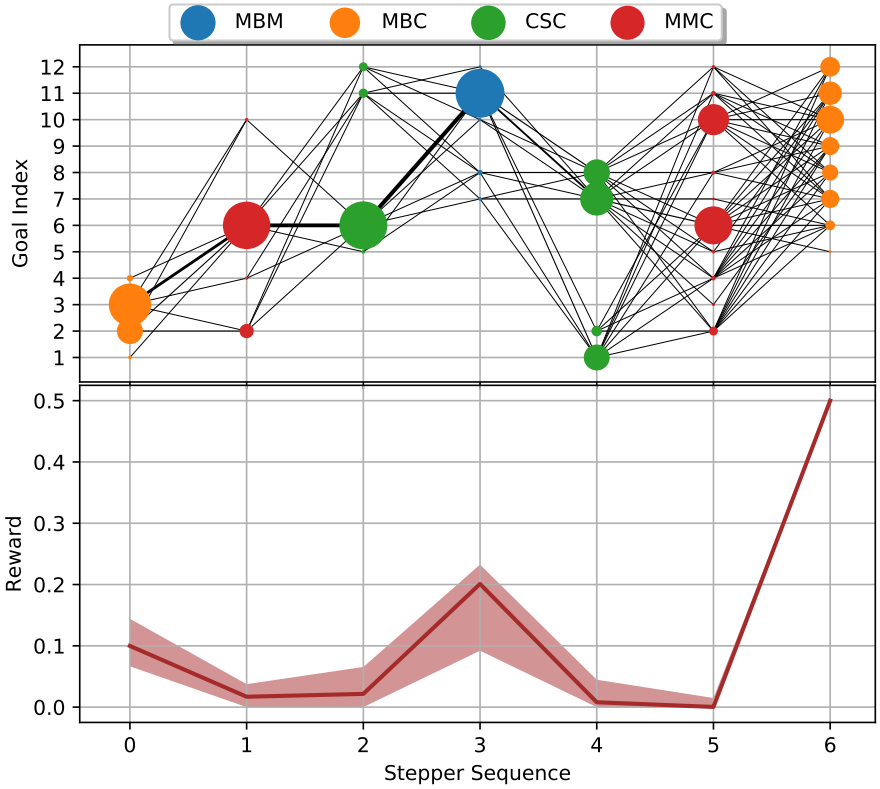


Figure 6.15: Subtask goal sequence and rewards for SOC during each step for all the successful runs during evaluation. Here the last three sequence steps have more even distribution. But selection of these goals are not influenced by the obstacle height, since the robot has already crossed the obstacle.

sequences are analogous to those that would be executed by an expert operator in real-world scenarios, thereby rendering it a more suitable candidate for use as an assistance function.

The hierarchy comprises three levels of controllers, with the lowest level comprising CSC, MBM and MMC, which are trained using RL. The controllers have distinct and particular objectives, which are shaped by the structure of the reward function. The observation space and action space are also small, which facilitates the training process and enhances its efficiency. The middle level of the hierarchy comprises two similar controllers, SIC and SOC, which utilize the controllers in the lowest level but have different predefined subtask sequences incorporated through IAM. The output of the middle layer controllers is the discretized representation of the goals for the controller in the lower level. The SC at the highest level merely initiates the middle-level controllers in accordance with the progress of the robot in reaching the goal.

The main contribution of this chapter is the design of the controller for stepping by incorporating domain knowledge during design and training using HRL and IAM. Such a design methodology could help in developing solutions that are more robust and more relatable for humans. The work in this chapter fulfills the objective O-4.

Conclusions, Contributions and Outlook

This chapter concludes the thesis by summarizing the developed control strategies for hybrid locomotion robots and highlighting their contributions to advancing autonomous mobility in complex terrains. It also outlines potential future research directions aimed at overcoming the current limitations and extending the applicability of these methods to real-world scenarios.

7.1 Conclusions

Hybrid locomotion, which entails the integration of multiple modes of locomotion within a single agent, confers a distinct advantage in environments characterized by challenging and evolving terrain. It is inevitable that robots with such capabilities will be used in the future to perform tasks that are either tedious or dangerous for humans. The control of locomotion in such systems presents a significant challenge due to the unconventional modes available to the robot in the environment. This thesis presents multiple locomotion control solutions for hybrid locomotion robots with varying morphologies.

The passive hybrid robot, Asguard, features a combination of wheeled and legged design, without the addition of any additional degrees of freedom. This results in the creation of a robot that is capable of traversing challenging terrain using simple control solutions typically employed for wheeled systems. However, the potential of the different locomotion modes remains untapped. To enhance the efficiency of locomotion and optimize the locomotive performance, cascaded position-velocity-torque controllers were developed based on the wheel states. The torque controller employs the deflection of the flexible coupling of the drive train to estimate the torque. The enhanced position control is employed to synchronize the motion of the wheels. Subsequently, various synchronization modes were evaluated, demonstrating that the specific resistance exhibited characteristics that spanned the spectrum between those observed in legged and wheeled locomotion. The most efficient configuration is achieved when the offset between the front and rear wheels is at its maximum, irrespective of the offset between the left and right wheels. The distinctive design also results in failures during point turns, where the passive joint tends to flip due to the large lateral traction. An algorithmic solution was developed which employed specific synchronization of the wheels along with other motion sequences to achieve better turning behavior.

The planetary rover, SherpaTT, has a wheel-on-leg design, which has been developed with the objective of enabling adaptation to uneven terrain. A 6-DoF force-torque sensor is affixed to the central axis of the wheel in order to quantify the interactions with the ground. A control framework, designated SherpaTT-MCS, was developed for the purpose of controlling the motion of the robot with numerous degrees of freedom. The framework incorporates a multitude of features, including the capacity to alter the footprint, autonomously adapt to terrain, regulate force levels, and perform driving and steer-

ing operations, while maintaining a limitation on the required computational resources. Moreover, distinct control modes are incorporated into the system to be deployed in accordance with the specific demands of the traversed environment. The ground adaptation controller has been developed with the objective of enabling the wheels to adapt to varying terrain, thereby maintaining contact with the ground, distributing forces between the wheels, and maintaining the roll and pitch of the body at a desired value. This is accomplished by transforming the vertical motion of the legs into control variables, which are then regulated via a feedback loop. The efficacy of the controller in each of its operational modes was evaluated in a laboratory setting. The findings indicated that the controllers were capable of adapting to fluctuations in the terrain, even in the absence of sophisticated perception sensors.

ARTER is a robot with a wheel-on-leg configuration, equipped with an additional manipulator that can be deployed as a leg. The configuration of the robot renders it particularly well-suited to deployment in extreme environments characterized by challenging terrain. The adaptation of wheels to terrain, as well as the usage of the manipulator for the purposes of increasing traction and traversing obstacles, are illustrative of the locomotive capabilities that are facilitated by such a morphology. The design concept of SherpaTT-MCS was adapted and extended to develop the ARTER-MCS for the control of motion during remote control and autonomous operations. The control framework comprises three layers, each of which implements a distinct functionality with varying degrees of complexity. These layers also provide appropriate interfaces for external control of the operations. Subsequently, the framework was demonstrated to be effective in two distinct scenarios: barrel recovery and soil sampling. In these cases, the focus was on remote control and autonomy, respectively.

In the context of driving with wheels on uneven terrain, it is expected that ARTER will demonstrate the capacity to adapt to the terrain in an autonomous manner. Due to the absence of high-fidelity contact force measurements, the utilization of terrain maps is indispensable. The terrain adaptation controller must simultaneously address multiple primary objectives, including the maintenance of stability, the avoidance of collisions and the maintenance of contact with the ground. Two additional objectives are included to minimize the joint movements and to maintain the desired roll and pitch angles, with the aim of improving the overall movements. Given the complexity of the problem, a DRL approach was employed to train the controller. In addition to the aforementioned joint states and stability margin, the observation states also included the distances of the robot chassis and wheels to the ground. Additionally, the latent representation of the terrain height map constitutes another component of the state, comprising a compressed form of the original data. The performance of controllers that differ in terms of the number of dimensions in their latent space representations and the inclusion of distance to ground states was evaluated through testing in simulation. The controller that demonstrated the optimal balance between usability and performance was the one with a latent space of sufficient dimension and, instead of utilizing contact distances, employed contact detection.

Stepping for ARTER represents a crucial locomotion mode, yet it is a highly intricate skill to master. It necessitates the coordinated movement of multiple joints in a series of sequences to successfully traverse an obstacle. The usage of DRL is challenging due to the considerable number of DoFs and the lengthy sequences, which cannot be solely achieved through normal exploration. The developed approach utilizes domain knowledge to inform and direct the design process through the utilization of HRL, and to facilitate and support the training process through the use of IAM. The HRL controller design comprises three layers of controllers, each of which is responsible for a specific, relatively simple subtask. The lowest layer comprises four controllers, which output basic commands and accept goals pertaining to the actions attempted by the controller. The second layer provides the goals, which are achieved by the lowest-level controllers. The top layer controller then uses the second-layer controllers

which are triggered based on initiation and termination conditions. IAM is used to predefine the sequence of selection of controllers in the lowest layer by the second layer, reducing the effort during training to selecting appropriate goals. The resulting controller achieved results similar to that of the state-of-the-art solutions using less design and development effort. The same controller was trained for three different terrain including gaps, steps and obstacles.

This thesis presents the development of controllers with varying control objectives and system and application requirements for hybrid locomotion robots with differing morphologies. The solutions, which range from bespoke robot solutions to learning-based solutions applicable to a class of robot, can be selected based on the morphology of the system. This work enhanced the locomotion capabilities of multiple hybrid locomotion robots by developing innovative solutions that leveraged the available multiple locomotion modes. This enhancement of locomotion capabilities has the potential to facilitate the deployment of robots in autonomous scenarios and applications that were previously deemed unfeasible.

7.2 Contributions

The thesis contributes to several aspects of locomotion control for hybrid locomotion robots, including software framework, locomotion optimization and development of new control solutions. These contributions and their corresponding objectives are listed below:

- **Locomotion Optimization for a Passive Hybrid Locomotion Robot**: Improved the efficiency and effectiveness of different motions of the Asguard passive hybrid locomotion robot.
  - **Torque Estimator:** Angular compression and extension of the flexible coupling in the wheel drive train is used to estimate the torque applied by the motor using an extended hysteresis model and a calibration process. The quadratic mean error in estimation of the torque was found to be within $5\,\%$ of the maximum torque.
  - **Cascaded Position Controller** (O-1a): The torque estimator is then used to implement a cascaded position, velocity and torque controller to improve the tracking performance of the wheels. The torque controller demonstrated an average accuracy of $93.7\,\%$. The overall controller showed a $56\,\%$ improvement in position tracking accuracy when compared with the motor-side controller.
  - **Optimization of Longitudinal Motion** (O-1b): Longitudinal movement of Asguard is studied to find the optimum positional offset between the wheels for maximum efficiency and reduced vibration. This created the possibility for front-rear offsets that are unaffected by turning motion, resulting in low specific resistance and reduced vertical vibrations. The average specific resistance with full front-rear offset is up to $90\,\%$ lower when compared to the worst-case.
  - **Improvement for Turning Motion** (O-1c): Development of a new algorithmic solution to prevent the passive joint from flipping over during a turn, the effectiveness of point turns is improved. The LOT controller successfully performed ten continuous turns under circumstances where normal point-turns failed to achieve even one full turn.

- **Active Suspension Planetary Rover Control**: Development of solutions and controllers for locomotion control of SherpaTT with adaptable legs and force-torque sensors on wheels. This contribution has been further successfully utilized in more than ten projects and three field test.
  - **Software Architecture for Control** (O-2a): Design and implementation of a control software architecture for controlling the locomotion, including steering, slip-free footprint changing and remote control.
  - **Ground Adaption Controller** (O-3a): Development of controller to autonomously maintain ground contact, distribute forces and maintain attitude on uneven ground by using the adaptability of the legs. In comparison to when the controller is not activated, the force variations were reduced by 80 % and the error in attitude angles, roll and pitch, was reduced by 95 % and 93 %, respectively.
- **Control Solutions for a Walking Excavator Robot**: Development of locomotion control solutions for the ARTER walking excavator robot which has several advanced locomotion modes including terrain adaption, stepping over obstacles, etc.
  - **Software Architecture for Autonomy and Remote Control** (O-2b): Remote control and autonomous operations were integrated into a control software framework, which was then tested in several realistic scenarios. The contribution of this thesis to this framework includes the overall design and implementation of several modules. Several modules are implemented by colleagues as part of the relevant project.
  - **Reinforcement Learning Controller for Terrain Adaption** (O-3b): Multiple reinforcement learning based controllers for uneven terrain adaptation are developed, evaluated and their performance compared. The controller design can be transferred to other robots with adaptable suspension systems without the need to extensive redesign.
  - **Hierarchical Reinforcement Learning Controller for Stepping** (O-4): Complex problem of stepping, with its many joints and long sequences, is solved by reinforcement learning, using hierarchical reinforcement learning and action masking. This design methodology, incorporating domain knowledge injection, enabled the generation of stepping controllers for three distinct types of steps, obviating the necessity for explicit modelling. This can be implemented for a new robot with minimal development effort.

7.3 Publications

The contributions in this thesis have been published in several international conferences, peer-reviewed articles and journals, book chapters and other articles. The consolidated list is given below:

- **International Conferences:**
  - i **Babu, A.** and F. Kirchner (2025). "Stepping Locomotion for a Walking Excavator Robot Using Hierarchical Reinforcement Learning and Action Masking". In: *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)*. IEEE, Hangzhou, CHINA (Accepted)

ii **Babu, A.** and F. Kirchner (2021). "Terrain Adaption Controller for a Walking Excavator Robot using Deep Reinforcement Learning". In: *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 64–70. DOI: 10.1109/ICAR53236.2021.9659399

iii **Babu, A.**, K. Y. Yurtdas, C. E. S. Koch, and M. Yüksel (2019). "Trajectory Following using Nonlinear Model Predictive Control and 3D Point-Cloud-based Localization for Autonomous Driving". In: *2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic*. IEEE

iv Cordes, F., **A. Babu**, and F. Kirchner (2017). "Static Force Distribution and Orientation Control for a Rover with an Actively Articulated Suspension System". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2017)*

v Cordes, F. and **A. Babu** (2016). "SherpaTT: A Versatile Hybrid Wheeled-Leg Rover". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2016), June*

vi Hidalgo-Carrio, J., **A. Babu**, and F. Kirchner (2014). "Static forces weighted Jacobian motion models for improved Odometry". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, pp. 169–175

vii Cordes, F., C. Oekermann, **A. Babu**, D. Kuehn, T. Stark, and F. Kirchner (2014). "An active suspension system for a planetary rover". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014), June*, pp. 17–19

viii **Babu, A.**, S. Joyeux, J. Schwendner, and F. Grimminger (2010). "Effects of Wheel Synchronization for the Hybrid Leg-Wheel Robot Asguard". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2010), August*

➢ **Journals:**

i **Babu, A.**, L. C. Danter, P. Willenbrock, S. Natarajan, D. Kuehn, and F. Kirchner (2022). *at - Automatisierungstechnik* 70:10, pp. 876–887. DOI: doi:10.1515/auto-2022-0056. URL: https://doi.org/10.1515/auto-2022-0056

ii Woock, P. and **A. Babu** (2022). "Autonome Robotersysteme in der Altlastensanierung". *Handbuch Altlastensanierung und Flächenmanagement*. Handbuch Altlastensanierung und Flächenmanagement 93. Aktualisierung, 3. Aufl.5111. Ed. by V. Franzius, M. Altenbockum, and T. Gerhold

iii Cordes, F., F. Kirchner, and **A. Babu** (2018). "Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain". *Journal of Field Robotics* 35:7, pp. 1149–1181. DOI: 10.1002/rob.21808. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21808

iv Joyeux, S., J. Schwendner, F. Kirchner, **A. Babu**, F. Grimminger, J. Machowinski, P. Paranhos, and C. Gaudig (2011). "Intelligent Mobility". *KI - Künstliche Intelligenz* 25:2, pp. 133–139. ISSN: 1610-1987. DOI: 10.1007/s13218-011-0089-8. URL: https://doi.org/10.1007/s13218-011-0089-8

➢ **Book Chapters:**

i **Babu, A.**, P. Willenbrock, J. Tiemann, F. Bernhard, and D. Kuehn (2024). "ARTER: a walking excavator robot". In: *Biologically Inspired Series-Parallel Hybrid Robots*. Ed. by S. Kumar, A.

Mueller, and F. Kirchner. 1. Auflage. Vol. 514. Elsevier Science, pp. 235–261. ISBN: 978-0-323-88482-2. URL: https://shop.elsevier.com/books/biologically-inspired-series-parallel-hybrid-robots/kumar/978-0-323-88482-2

ii Cordes, F., **A. Babu**, and T. Stark (2024). "Sherpa, a family of wheeled-leg rovers". In: *Biologically Inspired Series-Parallel Hybrid Robots*. Ed. by S. Kumar, A. Mueller, and F. Kirchner. 1. Auflage. Vol. 514. Elsevier Science, pp. 281–304. ISBN: 978-0-323-88482-2. URL: https://shop.elsevier.com/books/biologically-inspired-series-parallel-hybrid-robots/kumar/978-0-323-88482-2

➢ **Other Articles:**

i **Babu, A.** (2016). *Ground Adaption Process for Sherpa TT*. technical report. DFKI GmbH, pp. 84–91

ii Ahmed, M. and **A. Babu** (2014). "Autonomous Steering Controller for Path Following". In: *RIC Project Day Workgroups "Framework & Standardization" and "Manipulation & Control"*. Vol. 14-05. DFKI Documents ISBN: ISSN 0946-0098. DFKI GmbH. Selbstverlag, pp. 118–119

7.4 Outlook

The following highlights the potential research directions for the future, which address the inadequacies in the developed approaches and could further improve usability, performance and the domains of application.

Transferring the Learning-based Solutions to Real System: The solutions for terrain adaption and stepping locomotion for ARTER is currently trained and evaluated in simulation. Transferring this to the real system is challenging mainly due to the simulation reality gap. Further research is necessary to address this issue as the robot has complex non-linear dynamics and environment interactions which are difficult to be explicitly modelled and the parameters identified. Active research is being conducted to rectify this. Instead of fully modelling the dynamics of the system, it might also suffice to match the performance of the joint level velocity controllers to train robust controllers.

The terrain shape and properties are also key to training a robust controller. Height maps of real terrains can be sampled and using Generative Adversarial Networks (GAN) to generate realistic terrains that the robot might encounter in real scenarios. Domain randomization of the interaction properties could be yet another solution that could be explored.

Safety and Robustness of DRL Controllers: Safety and robustness of DRL are defined in Yamagata and Santos-Rodriguez 2024 on the basis of other definitions available in literature. Safety of learned policies ensures that the true objective is maintained and that the safety constraints are always respected during both learning and deployment. Additionally, there should be a possibility for humans to safely intervene at all times. Robustness primarily refers to the ability of the agent to account for all the relevant uncertainties in the environment.

In general, controllers developed for linear systems are well understood and has safety and robustness guarantees. For non-linear systems this has been ensured for numerous systems and controllers, at least within predefined limits. In the case of DRL controllers that use Deep Neural Networks (DNN), there are severe limitations with respect to safety and robustness.

Further research is necessary to develop methods to learn safe and robust controllers. Domain randomization or adversarial training could make the controllers less sensitive to uncertainties and variations in environment, thereby improving the robustness. Constrained MDP or penalties in reward function for unsafe behavior could make the training and deployment safer. Better safety guarantees are possible for model-based DRL for systems where the dynamics are accurately modelled. Human supervision using human-in-the-loop training and control gives the ability to guide the training and intervene in cases of dangerous actions.

Automatic Subtask Discovery: The stepping controller for ARTER is based on Hierarchical Deep Reinforcement Learning (HDRL) where the subtask hierarchy is designed by an expert. This is cumbersome for more complex problems. Automatic subtask discovery is the process of finding the hierarchical structure including the necessary observations, actions and rewards to achieve the high-level goals. Apart from the structure, the sub-goals that are necessary for attaining the main goal can also be automatically discovered. One of the aspects that promote the task discovery is the intrinsic motivation which drives the agent to explore and thereby learning skills relevant for different tasks.

The ability to discover subtasks automatically has several benefits. Since the main task is decomposed into a hierarchy of subtask which are less complex, the efficiency of learning improves. The subtasks can then potentially be used to achieve goals of multiple high-level tasks making it more transferable. These benefits are achieved without the need for manual design of the subtasks and the hierarchy. Several methods have been developed to address these including options discovery, state-space clustering, intrinsic motivation, bottleneck discovery, etc. Further research is necessary to make this more scalable, generalizable and tractable for multiple levels. Including guidance by humans could be more effective and the agents learn safe and robust controllers.

Simultaneous Training of Subtasks: Another critical aspect of the training in hierarchical reinforcement learning is the training of the subtasks. In this work, the learning is performed separately in a bottom-to-top approach in environments specifically designed for that particular task. The better option would be if the training of subtasks could be carried concurrently for multiple subtasks. The interaction and sharing of experiences could be used to improve the overall learning efficiency.

Nevertheless, this is challenging due to the non-stationary nature of the problem. During training the low-level policies change and hence produces noisy transitions and uncertainty for the high-level policies. This results in stability problems for the high-level policies. Addressing this in further research have the potential to reduce the effort and improve efficiency in training a hierarchy of policies.

Lifelong Learning: The solutions presented in this work are trained in simulation and has the potential to be deployed on real systems in real scenarios. Depending on the changes in the scenarios, retraining can be triggered with the new environment and deployed again. The system, even though acquires all the necessary data during deployment, lacks the ability to adapt and improve its skills over a longer period of time. Lifelong learning for robots gives it the ability to use the knowledge it acquires during deployment to continuously learn and adapt during its life span.

Research in several aspects including continuous and incremental learning, skill transfer, knowledge retention, autonomous exploration to attain new skills, etc., is necessary to attain progress in this domain. A hybrid locomotion system already has multiple modes of locomotion to adapt to complex and

varying terrain. This capability combined with the ability to perform lifelong learning could pave way for intelligent and autonomous mobile robots of the future that can tackle challenging environment.

Appendices

A Asguard Wheel Hysteresis

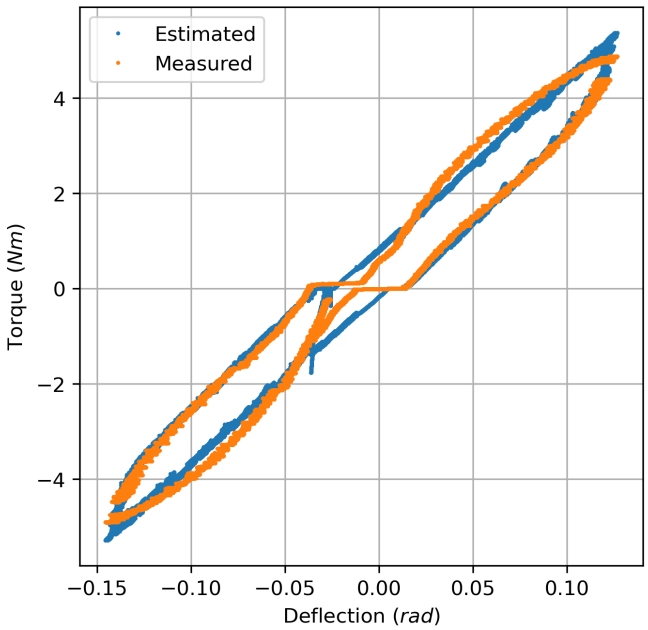


Figure A.1: Torque hysteresis reference data and fitted data for front-left wheel

The torques applied by the wheels are measured in Section 2.3.2 based on the deflection between the motor output shaft and the wheel which are connected using a flexible coupling. This deflection of this coupling though shows hysteresis which is modelled using Bouc-Wen model as described in Section 2.3.2. Additionally, the model and the details of estimation are described. The result for the rear-right wheel is also presented in Figure 2.7. The results of calibration for the wheels front-left, front-right and rear-left are shown in Figures A.1 to A.3 respectively.

Comparison of the estimation results show some differences in modelling performance between the different wheels. The rear-right wheel has the measured values which are smoother with less noise and hence model is able to better fit to the measurements. The other wheels show measurement that are more noisy and in some cases not repeatable. This results in a fitting that is very accurate, especially around the low torque and low deflection region.

Some discrepancies can be found in higher torque and deflection region where the curve for each direction are different to each other. For example, in Figure A.1 while the lower curve in the deflection range from 0.025 rad to 0.075 rad shows mostly linear properties, the curvature of the upper part shows noticeably higher. Even though the accuracy is adequate for the control purposes, better models for hysteresis are necessary to capture these dynamics more accurately.

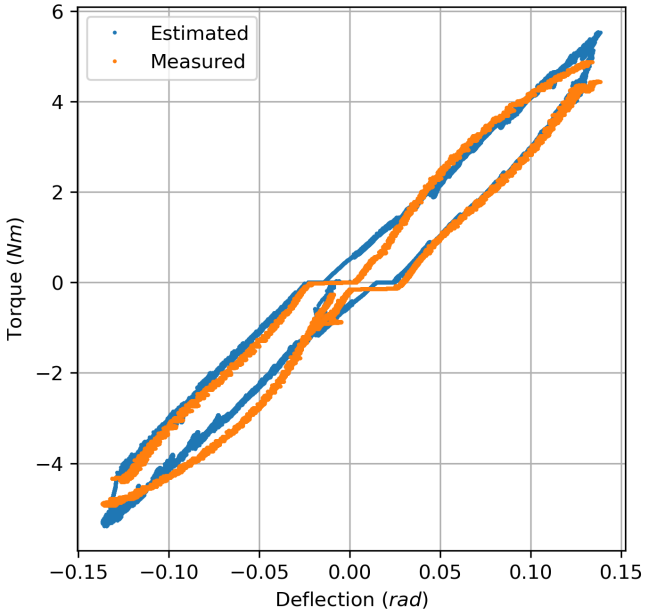


Figure A.2: Torque hysteresis reference data and fitted data for front-right wheel

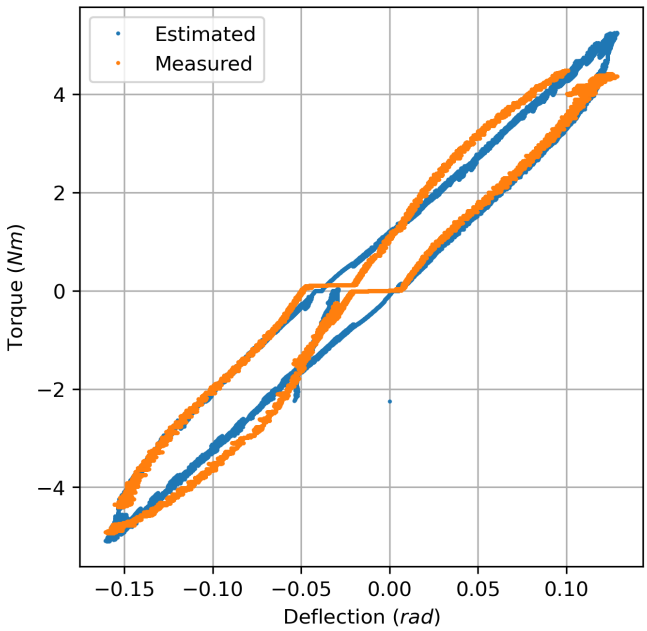


Figure A.3: Torque hysteresis reference data and fitted data for rear-left wheel

B SherpaTT Ground Adaption Process Step Response

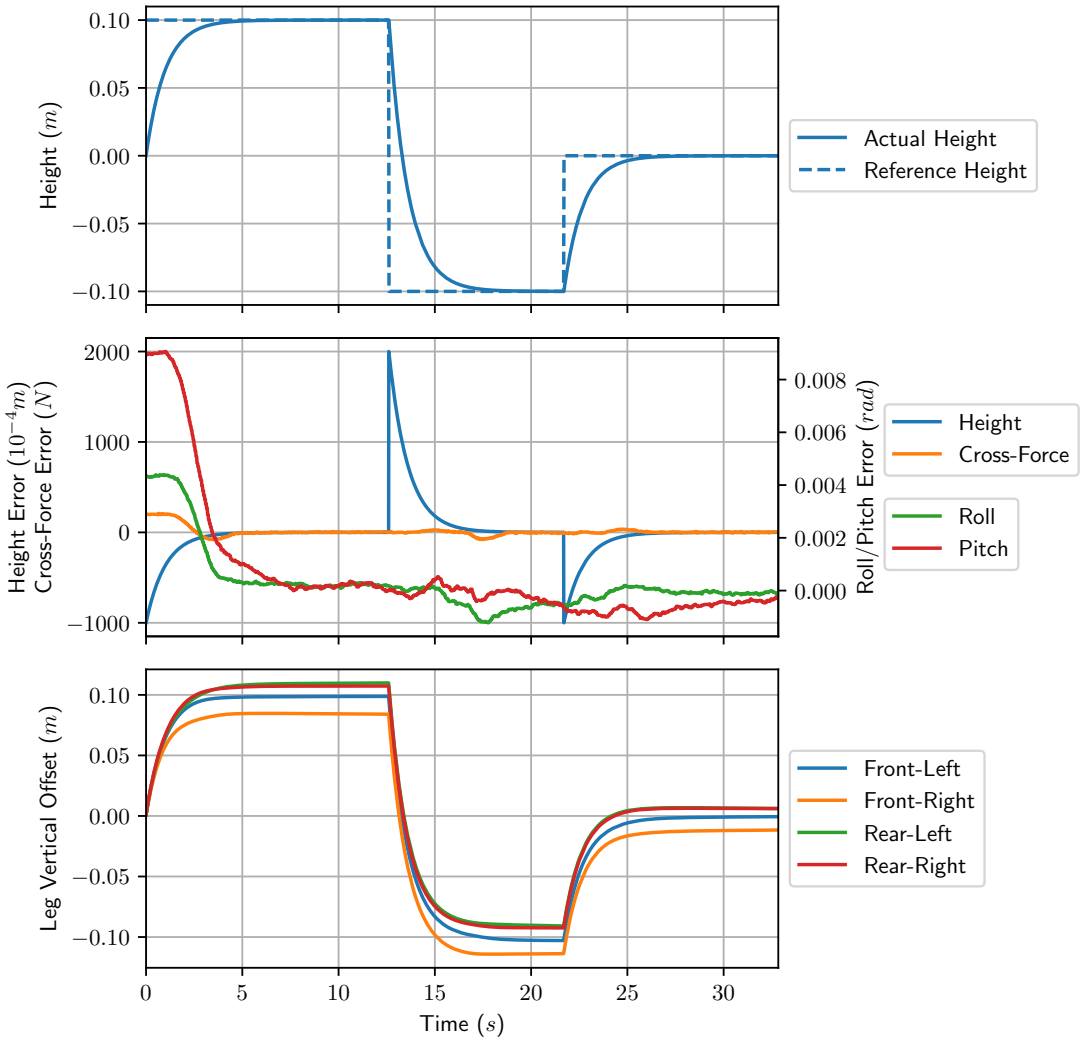


Figure B.1: Step response of GAP mean height ($h_{m,ref}$) controller.

The GAP controller developed in Section 3.5.1 can automatically adjust the legged-wheels to adapt to the changes in the ground. This ensures good ground contact and more even force distribution. Additionally, it is also capable of maintaining desired roll and pitch angles with respect to gravity. The controller was tested during motion in laboratory setup for which the results are presented in 3.5.3.

In addition to the experiments during motion, several experiments were conducted to evaluate the step response for different commands on a flat ground. The initial state of the robot is such that the robot legs are all in the same nominal pose where the z-coordinates of the legs are identical. Even in this case on flat ground, the ground contact is not even due to several uncertainties in the system. Different experiments were conducted for step responses for height, roll angle and pitch angles. An experiment with simultaneously step input for all the three references was also conducted.

The results from the step response for the different controllers are plotted in Figures B.1 and B.2, Figures B.3 and B.4. Each of these plots have three sub-plots. The top sub-plot shows the reference

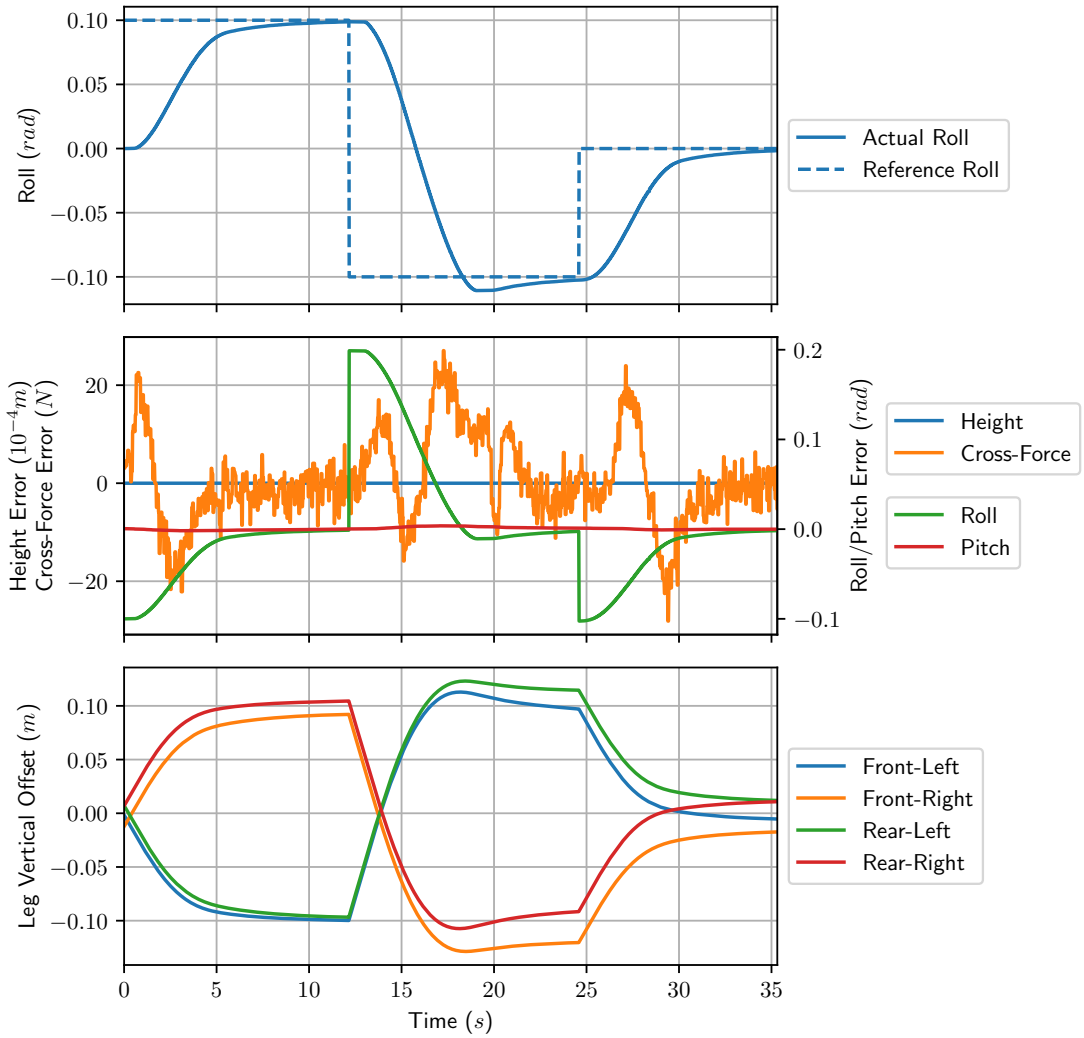


Figure B.2: Step response of GAP roll (θ_r) controller.

values and the actual value, plotted against time. The error values for all the components are plotted in the middle sub-plot. The total vertical motion for all the legs are plotted in the lowest sub-plot.

Figure B.1 shows the results of the step response to change the height reference. To obtain the step response of mean-height ($h_{m,ref}$), a reference value of $0.1\,\mathrm{m}$ was given for a time-period of $12.5\,\mathrm{s}$ initially. This is followed by a reference values of $-0.1\,\mathrm{m}$ and back to $0.0\,\mathrm{m}$ at $22.5\,\mathrm{s}$. The controllers for cross-force, roll control and pitch control are activated simultaneous at the start of the step signal. The results show that all the control values reaches steady-state within $5.0\,\mathrm{s}$ of the start of the step input. During the second step input from positive value to negative value, it takes $10.0\,\mathrm{s}$ to reach the steady state. Even though the roll and pitch errors are stable, there is small spike in the cross-force error. The change in vertical offsets of the legs also show the variations that are necessary for each leg to adjust to the uncertainties.

Similarly, step input of roll (θ_r) and pitch (θ_p) controllers were also tested with references of $0.1\,\mathrm{rad}$, $-0.1\,\mathrm{rad}$ and $0.0\,\mathrm{rad}$ at the time-periods $0.0\,\mathrm{s}$, $12.0\,\mathrm{s}$ and $25.0\,\mathrm{s}$ respectively. The results for roll

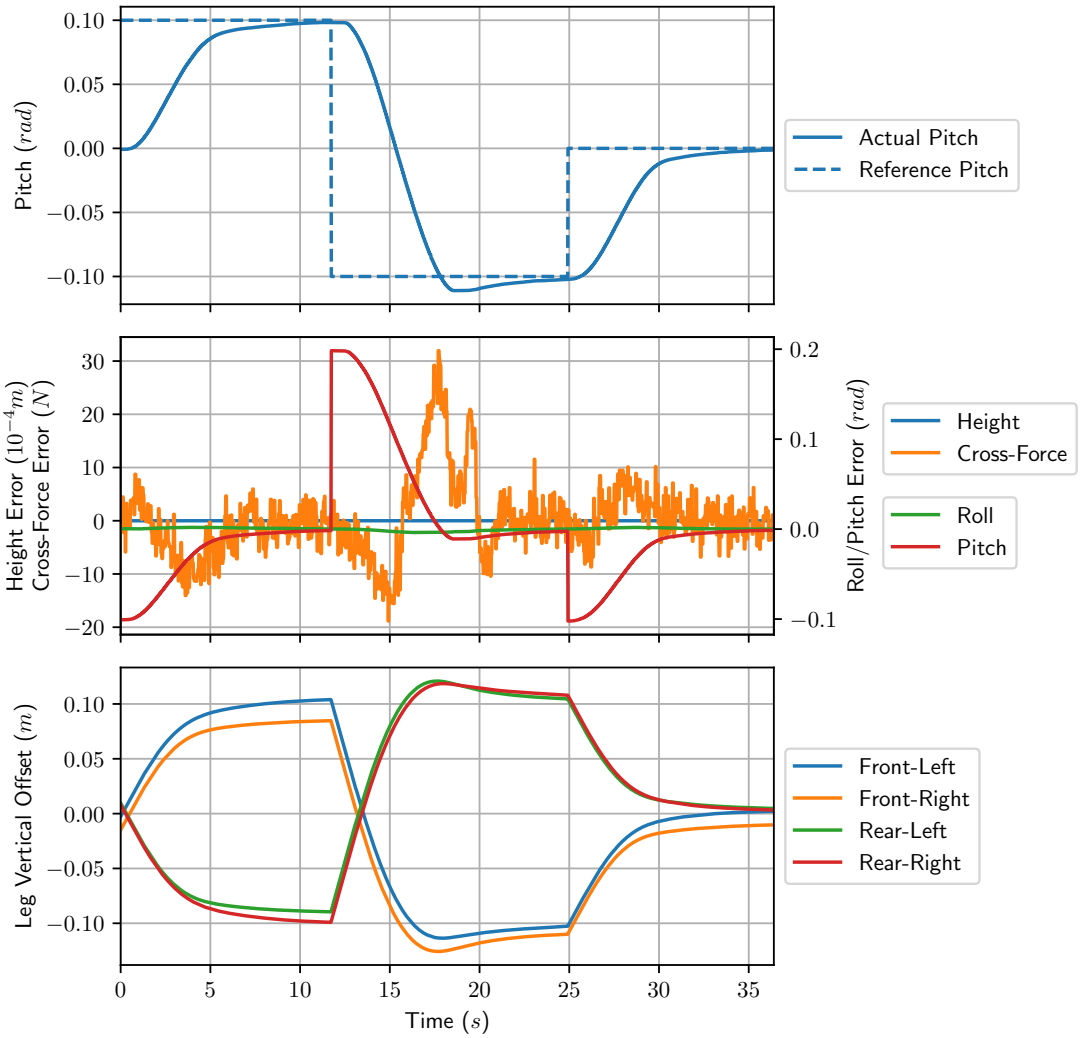


Figure B.3: Step response of GAP pitch (θ_p) controller.

and pitch are plotted in Figure B.2 and fig:app:GAP-performance-pitch respectively. All the control values converge to steady state after 7.0 s, 12.0 s and 6.0 s of giving the step inputs. The vertical leg positions are adapted automatically, such that depending on whether the input is for roll or pitch the corresponding legs are adjusted. The step responses for roll and pitch show a small overshoot as well.

To evaluate the stability of the controller under extreme circumstances, simultaneous step inputs were given for height, roll and pitch references. Under normal operating conditions, this is not expected to happen where the change in terrain is mostly continuous, and the reference values are only expected to change smoothly and continuously. Figure B.4 shows the step response of the controllers when simultaneously commanded. The magnitudes of step for height are ± 0.1 m and ± 0.05 rad for roll and pitch references. The roll and pitch values show overshoots, along with overshoots in the cross-force values. Nevertheless, all the controllers values stabilize and reach steady-state with in 10.0 s, 20.0 s and 10.0 s of start of the respective step signal.

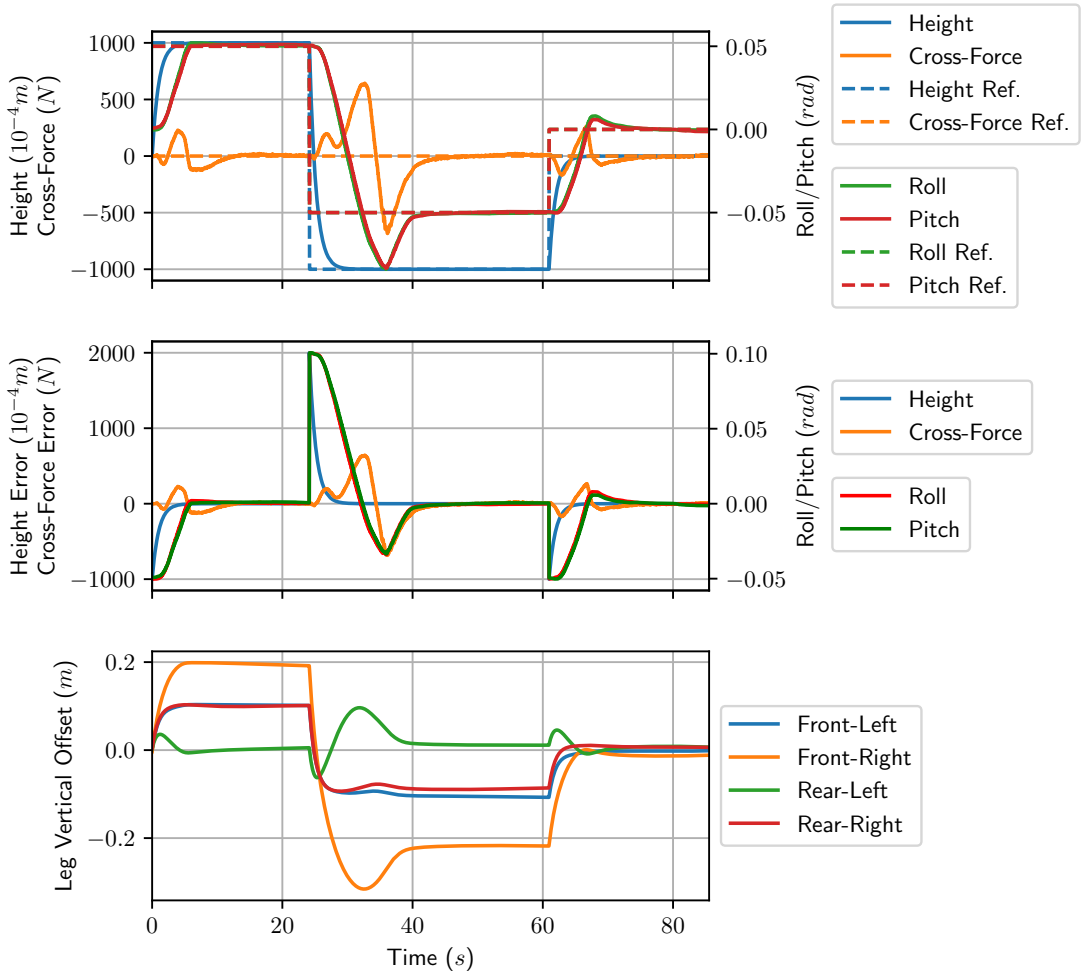


Figure B.4: Simultaneous step response of mean height ($h_{m,ref}$), roll (θ_r) and pitch (θ_p) controllers.

C Arter-MCS Low-Level Controller Design

The MCS of the ARTER robot which is described in Chapter 4 has different layers of control. The middle and the higher layers which controls different aspects of the robot are detailed in Section 4.3. The details of the lower-layer which controls the individual joints are described here, with the depiction shown in Figure C.1.

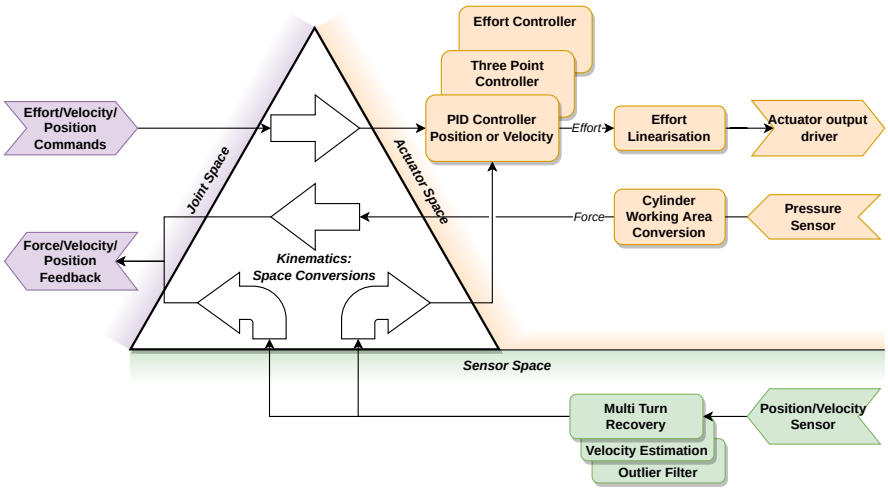


Figure C.1: Low-level control structure of ARTER showing the joint level controllers running on the PLC. (Image source: (Babu et al. 2024))

The objective of the low-level control architecture is to have a unified design for control of the joints such that it encapsulates all the complexities and variations. The control of individual joints of ARTER is complex due to multiple reasons: 1. Complex parallel kinematics, 2. Variation in placement of sensors, 3. Multitude of control modes, 4. Different types of commands to actuators, and 5. Non-linearity in hydraulic valve control. Every joint control is unique due to these reasons and considering the large degrees of freedom of the robot, it is inevitable to have a common architecture unifying the control of all the joints.

The parallel kinematics of joint are sometimes complex and differs from joint to joint. Additionally, due to the constraints in mounting the joint encoders, it is not always possible to attach it to the joint axis. The mounting location can be the joint axis, the actuator or an intermediate axis in the parallel kinematic. This gives rise to three different control spaces: joint, actuator and sensor. The conversion of the commands (position, velocity and effort) and states (position, velocity and force) between these different spaces is necessary to facilitate accurate control of the joints.

The valves that control the actuators are either direction control valves or proportional flow control valves. For the joints corresponding to the direction control valves, position and effort controls are available. The joints with proportional control valves have an additional velocity control implemented.

The controller of the joints are all working in the actuator space and all variables in other spaces are converted to the actuator space using the kinematics of the joint. The actuator mechanism with hydraulic valve and actuators have very high non-linearity which cannot be accounted for by the classical PID controllers. The main contributors for the non-linearities are the dead-zones and inherent non-linear behavior of the hydraulic valves. The commands to the valves are hence linearized using lookup-tables that map the desired velocities to the appropriate valve currents.

The joint position or velocity sensors are processed further to account for multiple turns, velocity estimation, filtering and outlier removal. The hydraulic pressures of either sides of the actuator cylinders are measured using pressure sensors. These values then converted to the corresponding force or torque measurements at the joint space using the cross-sectional area or each sides of the cylinder, and the kinematics of the joint.

The low-level controller is implemented in a control PLC and communicates with the mid-level and high-level control computer using User Datagram Protocol (UDP) messages. The sending and receiving of commands and states runs at 50 Hz and the control loop in the PLC run at 100 Hz. More details about the control aspects can be found in Babu et al. 2024.

D Normalized Energy Stability Margin

A good stability margin which encapsulates all the necessary factors and provide a value representing the stability of a mobile system is inevitable, especially if these systems operate on uneven and challenging terrain. NESM is one of the most effective and widely used stability margin for mobile robots, which is introduced in Section 5.2.3. The potential energy necessary to destabilize a robot is given by ESM. Normalizing ESM by the weight of the robot gives the NESM margin which is comparable between different robots and configurations.

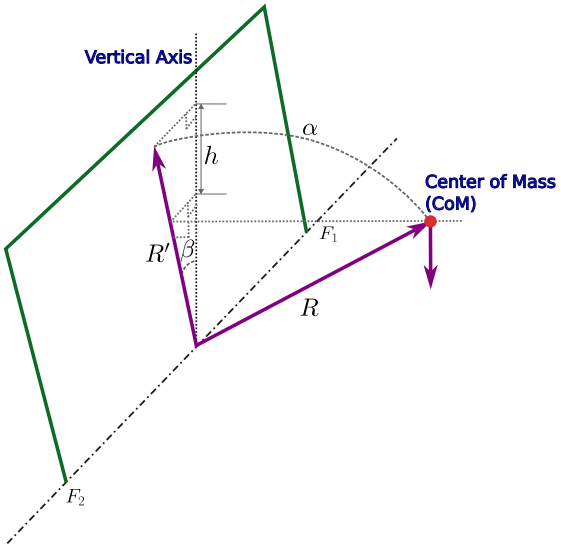


Figure D.1: ESM illustration (Inspired by Figure 6.2 in Brunner 2015)

Computation necessary for computation of ESM is given in Figure D.1. The dash-dotted line connecting F_1 and F_2 is one of the edges of the support polygon formed by the ground contact points of the robot limbs. The vertical plane, represented in green color, passes though the vertical axis and the F_1F_2 line. E is the shortest vector from the line F_1F_2 to the CoM. Vector E' is obtained by rotating the vector E about F_1F_2 until it coincides with the vertical plane. The angle between E and E' is θ. The angle formed between E' and the vertical axis is Ψ which is also the inclination of the support polygon edge with respect to the horizontal plane. Using these, the minimum potential energy needed to tip over the robot about the F_1F_2 polygon edge is given by

$$h = \|R\|(1 - \cos\alpha)\cos\beta. \tag{1}$$

The ESM is the minimum of all the computed potential energies corresponding to all the support polygon edges. The NESM is obtained by dividing the ESM with the weight of the robot.

Different ESM values for ARTER with varying contact points, and hence varying support polygons are shown in Figure D.2. The top image shows the scenario where the manipulator and all the four wheels are in contact with the ground and hence forming a support polygon with five edges. The edge between the front-right and rear-right wheels of the robot has the lowest margin and hence marked in red color. The bottom left image shows the manipulator retracted and moved slightly towards the right side of the robot resulting in further lowering of the stability margin on the right side. Moving the manipulator to the left, as shown in the bottom right image, reduces the stability margin on the

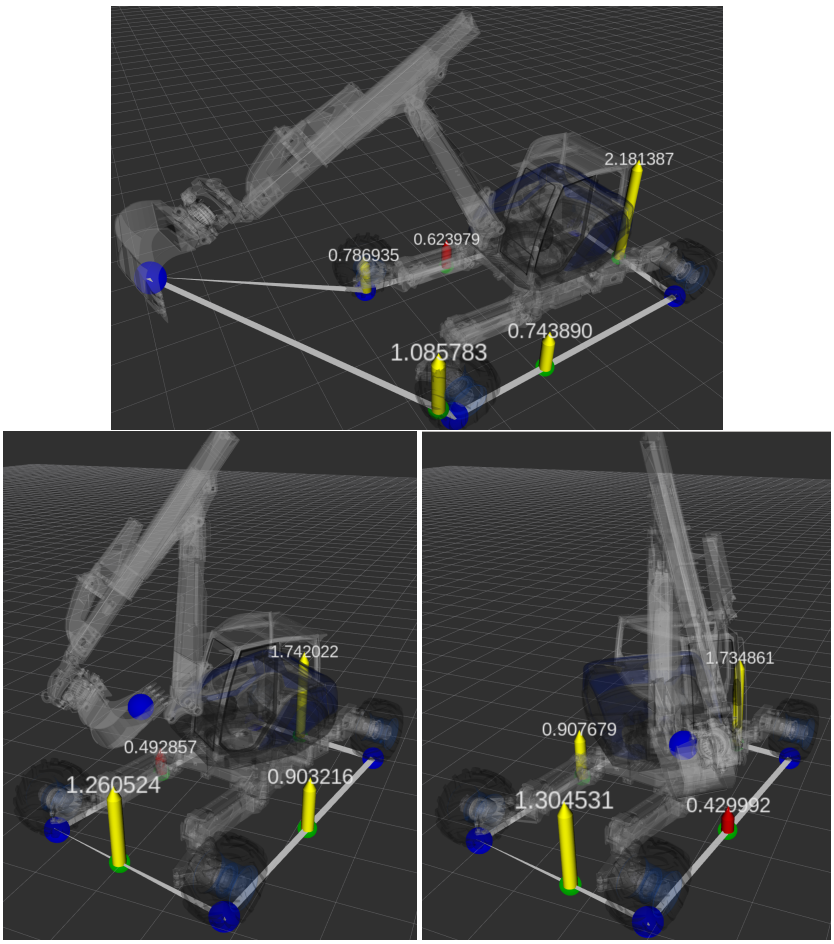


Figure D.2: Screenshots of ARTER visualization with different support polygons and poses of the manipulator to illustrate changes in ESM values.

left side and hence the ESM is caused by the support polygon edge between front-left and rear-left wheels.

E Designs of Autoencoders for terrain representation

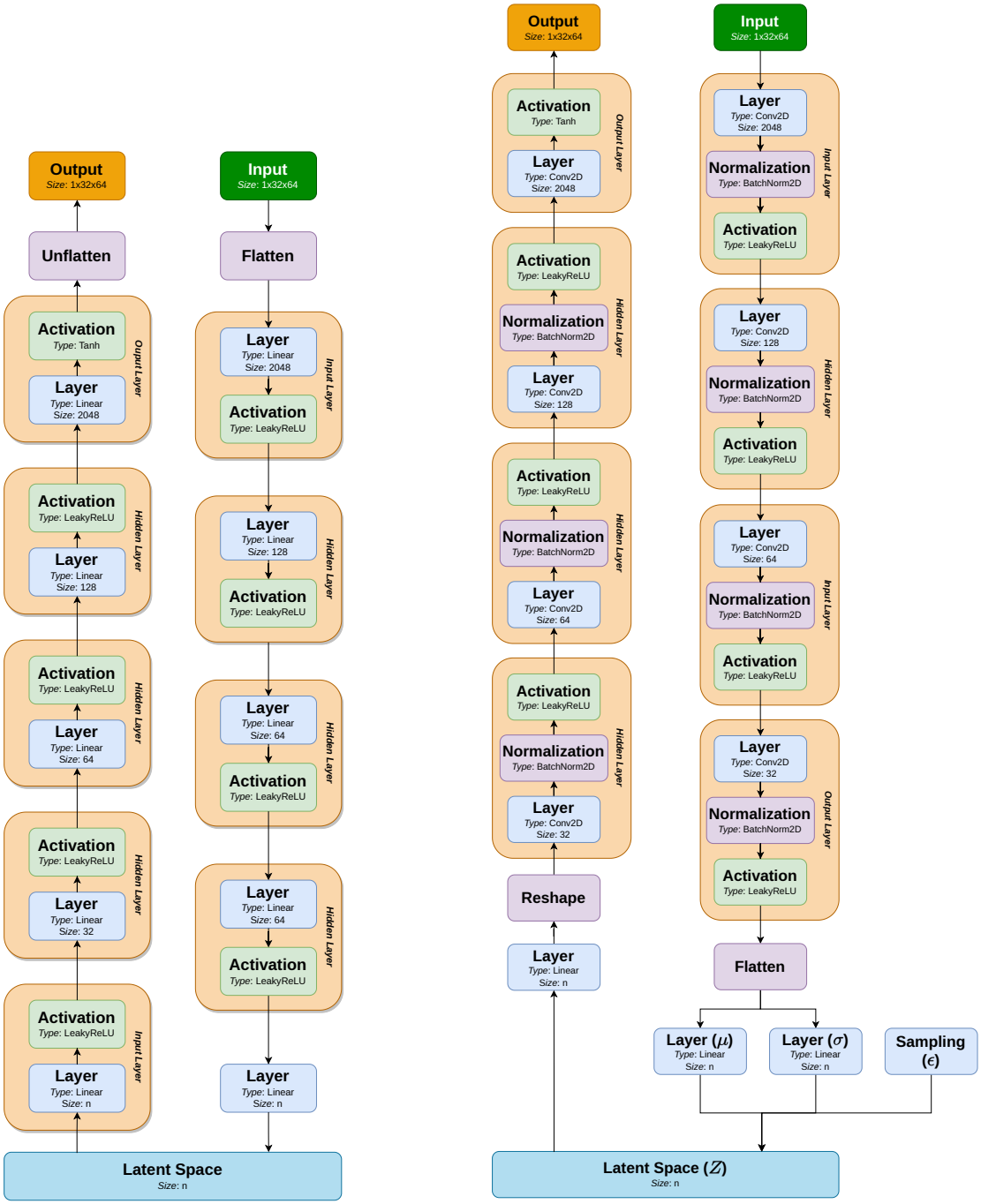


Figure E.1: Network architecture of the linear autoencoder and Variational Autoencoder that are used in Chapter 5.

Acronyms

β-VAE	β-Variational Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Networks
ARTER	Autonomous Rough Terrain Excavator Robot
BC	Base-Controller
BCS	Body Coordinate System
BHC	Body Height Control
BTC	Base-Terrain-Controller
BWBN	Bouc-Wen-Baber-Noori
CAE	Convolutional Autoencoder
CAN	Controller Area Network
CNN	Convolutional Neural Network
CoG	Center of Gravity
CoM	Center of Mass
COP	Center of Pressure
CPG	Central Pattern Generators
CSC	Contact Switch Controller
D-β-VAE	Disentangled β-Variational Autoencoder
DARPA	Defense Advanced Research Projects Agency
DESM	Dynamic Energy Stability Margin
DFKI	German Research Center for Artificial Intelligence
DNN	Deep Neural Networks
DoF	Degree of Freedom
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DS	Double Stance
EMI	Electromechanical Interface
ESM	Energy Stability Margin
FBM	Four-Bar Mechanism
FL	Front-Left
FLC	Force Levelling Controller
FLW	Front-Left Wheel
FM	FORWARD mode
FPGA	Field Programmable Gate Array
FR	Front-Right
FRCO	Front-Rear Cross Offset
FRO	Front-Rear Offset
FRW	Front-Right Wheel
GAE	Generalized Advantage Estimator
GAN	Generative Adversarial Networks

GAP	Ground Adaption Process
GCS	Gravity Aligned Coordinate System
GNMS	Gauss-Newton Multiple Shooting
GP	Goal Point
GPI	Generalized Proportional Integral
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRL	Guided Reinforcement Learning
gSDE	Generalized State Dependent Exploration
GUI	Graphical User Interface
HDRL	Hierarchical Deep Reinforcement Learning
HEAP	Hydraulic Excavator for an Autonomous Purpose
HRL	Hierarchical Reinforcement Learning
HZD	Hybrid Zero Dynamics
IAM	Invalid Action Masking
ICR	Instantaneous Center of Rotation
iLQR	iterative Linear Quadratic Regulator
iLQR-GNMS(M)	Closed Loop Gauss-Newton Multiple Shooting
IMU	Inertial Measurement Unit
IR	Infra-Red
ISCM	Inverted Slider Crank Mechanism
KL	Kullback-Leibler
LAE	Linear Autoencoder
LEP	Leg End Point
LIDAR	Light Detection and Ranging
LMPC	Linear Model Predictive Control
LOT	Load-Optimized Turning
LQR	Linear Quadratic Regulator
LRO	Left-Right Offset
MBC	Move Base Controller
MBM	Move Base and Manipulator Controller
MCS	Motion Control System
MDP	Markov Decision Process
MLP	Multilayer Perceptron
MMC	Move Manipulator Controller
MPC	Model Predictive Control
MPPO	Masked-Proximal Policy Optimization
NDESM	Normalized Dynamic Energy Stability Margin
NESM	Normalized Energy Stability Margin
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
OMPL	Open Motion Planning Library
P	Proportional
PI	Proportional Integral
PID	Proportional Integral Derivative
PIV	Proportional Integral Velocity
PLC	Programmable Logic Controller

PPO	Proximal Policy Optimization
PV	Position-Velocity
PVT	Position-Velocity-Torque
PWM	Pulse Width Modulation
QP	Quadratic Programming
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RGBD	Red Green Blue Depth
RIC	Robotics Innovation Center
RL	Rear-Left
RLW	Rear-Left Wheel
RM	REVERSE mode
RMS	Root-mean-squared
RMSE	Root-mean-squared error
ROCK	Robot Construction Kit
ROS	Robot Operating System
RPA	Roll-Pitch Adaption
RR	Rear-Right
RRW	Rear-Right Wheel
RTK	Real-Time Kinematic
SAC	Soft-Actor-Critic
SC	Stepper Controller
SIC	Step-In Controller
SLAM	Simultaneous Localization And Mapping
SMDP	Semi-Markov Decision Process
SOC	Step-Out Controller
SP	Start Point
SQP	Sequential Quadratic Programming
STC	Stepper Controller
SVD	Singular Value Decomposition
TAC	Terrain Adaption Controller
Tanh	hyperbolic tangent
TBP	Tool Base Point
TC	Terrain-Controller
TEP	Tool End Point
TM	TURN mode
TO	Trajectory Optimization
ToF	Time of Flight
UDP	User Datagram Protocol
URDF	Unified Robotics Description Format
VAE	Variational Autoencoder
VS	Vertical Stance
WBC	Whole Body Control
WCS	World Coordinate System
WDLS	Weighted Damped Least-Square
WSS	Wheel Steering Support
ZMP	Zero Moment Point

Glossary

DARPA Robotics Challenge	Robotics competition funded by the US Defense Advanced Research Projects Agency (DARPA)
ROBDEKON	Competence center for development of robotic systems for decontamination in hazardous environments. https://robdekon.de/en
RQT	Qt-based framework for GUI development for ROS. http://wiki.ros.org/rqt
RViz	3D visualization tool for ROS. http://wiki.ros.org/rviz

Bibliography

Ahmed, M. and A. Babu (2014). "Autonomous Steering Controller for Path Following". In: *RIC Project Day Workgroups "Framework & Standardization" and "Manipulation & Control"*. Vol. 14-05. DFKI Documents ISBN: ISSN 0946-0098. DFKI GmbH. Selbstverlag, pp. 118–119.

Alexander, R. M. (2002). *Principles of Animal Locomotion*. Princeton University Press, Princeton. ISBN: 9781400849512. DOI: doi:10.1515/9781400849512. URL: https://doi.org/10.1515/9781400849512.

Altendorfer, R., D. E. Koditschek, and P. Holmes (2004). "Stability Analysis of a Clock-Driven Rigid-Body SLIP Model for RHex". *The International Journal of Robotics Research* 23:10-11, pp. 1001–1012. DOI: 10.1177/0278364904047390. eprint: https://doi.org/10.1177/0278364904047390. URL: https://doi.org/10.1177/0278364904047390.

Appendix V: Uncertainties and Error Propagation (2004). Technical report. Case Western Reserve University.

Asadi, B. and S. Natarajan (2014). "Motion Planning for Manipulators". In: vol. 14-03. 14-03. Selbstverlag, Bremen, series DFKI Documents, DFKI GmbH, pp. 120–121.

Babu, A. (2014). "Control of Flexible Link Manipulator". In: *RIC Project Day "Framework & Standardization" and "Manipulation & Control"*. Vol. 14-03. DFKI Documents ISBN: ISSN 0946-0098. DFKI GmbH. Selbstverlag, pp. 67–75.

– (2016). *Ground Adaption Process for SherpaTT*. Technical report. DFKI GmbH, pp. 84–91.

Babu, A., L. C. Danter, P. Willenbrock, S. Natarajan, D. Kuehn, and F. Kirchner (2022). *at - Automatisierungstechnik* 70:10, pp. 876–887. DOI: doi:10.1515/auto-2022-0056. URL: https://doi.org/10.1515/auto-2022-0056.

Babu, A., S. Joyeux, J. Schwendner, and F. Grimminger (2010). "Effects of Wheel Synchronization for the Hybrid Leg-Wheel Robot Asguard". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2010), August*.

Babu, A. and F. Kirchner (2021). "Terrain Adaption Controller for a Walking Excavator Robot using Deep Reinforcement Learning". In: *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 64–70. DOI: 10.1109/ICAR53236.2021.9659399.

– (2025). "Stepping Locomotion for a Walking Excavator Robot Using Hierarchical Reinforcement Learning and Action Masking". In: *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)*. IEEE, Hangzhou, CHINA.

Babu, A., P. Willenbrock, J. Tiemann, F. Bernhard, and D. Kuehn (2024). "ARTER: a walking excavator robot". In: *Biologically Inspired Series-Parallel Hybrid Robots*. Ed. by S. Kumar, A. Mueller, and F. Kirchner. 1. Auflage. Vol. 514. Elsevier Science, pp. 235–261. ISBN: 978-0-323-88482-2. URL: https://shop.elsevier.com/books/biologically-inspired-series-parallel-hybrid-robots/kumar/978-0-323-88482-2.

Babu, A., K. Y. Yurtdas, C. E. S. Koch, and M. Yüksel (2019). "Trajectory Following using Nonlinear Model Predictive Control and 3D Point-Cloud-based Localization for Autonomous Driving". In: *2019 European Conference on Mobile Robots (ECMR), Prague, Czech Republic*. IEEE.

Bacon, P.-L., J. Harb, and D. Precup (2017). "The Option-Critic Architecture". *Proceedings of the AAAI Conference on Artificial Intelligence* 31:1. DOI: 10.1609/aaai.v31i1.10916. URL: https://ojs.aaai.org/index.php/AAAI/article/view/10916.

Baek, D., A. Purushottam, and J. Ramos (2022). "Hybrid LMC: Hybrid Learning and Model-based Control for Wheeled Humanoid Robot via Ensemble Deep Reinforcement Learning". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9347–9354. DOI: 10.1109/IROS47612.2022.9981913.

Barto, A. G. and S. Mahadevan (2003). "Recent advances in hierarchical reinforcement learning". *Discrete event dynamic systems* 13, pp. 341–379.

Beal, D. N., F. S. Hover, M. S. Triantafyllou, J. C. Liao, and G. V. Lauder (2006). "Passive propulsion in vortex wakes". *Journal of Fluid Mechanics* 549, pp. 385–402. DOI: 10.1017/S0022112005007925.

Becedas, J., V. Feliu, and H. Sira-Ramírez (2009). "Flatness based GPI Control for Flexible Robots". In: *Advances in Computational Algorithms and Data Analysis*. Ed. by S.-I. Ao, B. Rieger, and S.-S. Chen. Springer Netherlands, Dordrecht, pp. 395–409. ISBN: 978-1-4020-8919-0. DOI: 10.1007/978-1-4020-8919-0_27. URL: https://doi.org/10.1007/978-1-4020-8919-0_27.

Bellegarda, G. and K. Byl (2019). "Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, pp. 7776–7781.

Bellicoso, C. D., F. Jenelten, C. Gehring, and M. Hutter (2018). "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots". *IEEE Robotics and Automation Letters* 3:3, pp. 2261–2268.

Berner, C., G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. (2019). "Dota 2 with large scale deep reinforcement learning". *arXiv preprint arXiv:1912.06680*.

Besseron, G., C. Grand, F. Ben Amar, and P. Bidaud (2008). "Decoupled control of the high mobility robot Hylos based on a dynamic stability margin". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2435–2440. DOI: 10.1109/IROS.2008.4651092.

Bjelonic, M., C. D. Bellicoso, Y. de Viragh, D. Sako, F. D. Tresoldi, F. Jenelten, and M. Hutter (2019). "Keep Rollin'—Whole-Body Motion Control and Planning for Wheeled Quadrupedal Robots". *IEEE Robotics and Automation Letters* 4:2, pp. 2116–2123. DOI: 10.1109/lra.2019.2899750.

Bjelonic, M., R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter (2021). "Whole-body MPC and online gait sequence generation for wheeled-legged robots". In: *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 8388–8395.

Bjelonic, M., P. K. Sankar, C. D. Bellicoso, H. Vallery, and M. Hutter (2020). "Rolling in the deep–hybrid locomotion for wheeled-legged robots using online trajectory optimization". *IEEE Robotics and Automation Letters* 5:2, pp. 3626–3633.

Blickhan, R., A. Seyfarth, H. Wagner, A. Friedrichs, M. Günther, and K. D. Maier (2006). "Robust Behaviour of the Human Leg". In: *Adaptive Motion of Animals and Machines*. Springer-Verlag Tokyo, pp. 5–16.

Boston-Dynamics (2017). URL: https://bostondynamics.com/legacy/.

Botvinick, M. M., Y. Niv, and A. G. Barto (2011). "Hierarchically organised behaviour and its neural foundations: a reinforcement-learning perspective". In: *Modelling Natural Action Selection*. Ed. by A. K. Seth, T. J. Prescott, and J. J. Bryson. Cambridge University Press, pp. 264–299.

Bouc, R. (1967). "Forced vibrations of mechanical systems with hysteresis". In: *Proc. of the Fourth Conference on Nonlinear Oscillations, Prague, 1967*.

Brunner, M., B. Brüggemann, and D. Schulz (2012). "Motion planning for actively reconfigurable mobile robots in search and rescue scenarios". In: *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–6. DOI: 10.1109/SSRR.2012.6523896.

Brunner, M. (2015). "Rough Terrain Motion Planning for Actively Reconfigurable Mobile Robots". Aachen, Techn. Hochsch., Diss., 2015. Dissertation. Aachen: RWTH Aachen University, VII, 173 S. : Ill., graph. Darst. URL: https://publications.rwth-aachen.de/record/462738.

Brunner, M., B. Brüggemann, and D. Schulz (2013). "Hierarchical rough terrain motion planning using an optimal sampling-based method". In: *2013 IEEE International Conference on Robotics and Automation*, pp. 5539–5544. DOI: 10.1109/ICRA.2013.6631372.

Bruzzone, L., M. Baggetta, S. E. Nodehi, P. Bilancia, and P. Fanghella (2021). "Functional Design of a Hybrid Leg-Wheel-Track Ground Mobile Robot". *Machines* 9:1. ISSN: 2075-1702. DOI: 10.3390/machines9010010. URL: https://www.mdpi.com/2075-1702/9/1/10.

Bruzzone, L., S. E. Nodehi, and P. Fanghella (2024). "WheTLHLoc 4W: Small-scale inspection ground mobile robot with two tracks, two rotating legs, and four wheels". *Journal of Field Robotics* 41:4, pp. 1146–1166. DOI: https://doi.org/10.1002/rob.22314. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.22314. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22314.

Burgess, C. P., I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner (2018). *Understanding disentangling in β-VAE*. arXiv: 1804.03599 [stat.ML]. URL: https://arxiv.org/abs/1804.03599.

Burzyński, P., E. Pawłuszewicz, L. Ambroziak, and S. Sharma (2024). "Kinematic Analysis and Application to Control Logic Development for RHex Robot Locomotion". *Sensors* 24:5. ISSN: 1424-8220. DOI: 10.3390/s24051636. URL: https://www.mdpi.com/1424-8220/24/5/1636.

Byl, K. and M. Byl (2015). "Design of fast walking with one- versus two-at-a-time swing leg motions for RoboSimian". In: *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pp. 1–7. DOI: 10.1109/TePRA.2015.7219688.

Calzolari, D. C. E., B. Schürmann, and M. Althoff (2017). "Comparison of Trajectory Tracking Controllers for Autonomous Vehicles". In:

Campbell, D. (2004). "Bounding and Stair Descent in the Hexapod RHex". MA thesis. Department of Electrical and Computer Engineering, McGill University, Montreal.

Chen, S.-C., K.-J. Huang, W.-H. Chen, S.-Y. Shen, C.-H. Li, and P.-C. Lin (2013). "Quattroped: a leg–wheel transformable robot". *IEEE/ASME Transactions On Mechatronics* 19:2, pp. 730–742.

Chen, S.-C., K. J. Huang, C.-H. Li, and P.-C. Lin (2011). "Trajectory planning for stair climbing in the leg-wheel hybrid mobile robot quattroped". In: *2011 IEEE International Conference on Robotics and Automation*, pp. 1229–1234. DOI: 10.1109/ICRA.2011.5980091.

Chen, W.-H., H.-S. Lin, Y.-M. Lin, and P.-C. Lin (2017). "TurboQuad: A Novel Leg–Wheel Transformable Robot With Smooth and Fast Behavioral Transitions". *IEEE Transactions on Robotics* 33:5, pp. 1025–1040. DOI: 10.1109/TRO.2017.2696022.

Chiou, M., G.-T. Epsimos, G. Nikolaou, P. Pappas, G. Petousakis, S. Mühl, and R. Stolkin (2022). "Robot-assisted nuclear disaster response: Report and insights from a field exercise". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4545–4552.

Coleman, D., I. A. Sucan, S. Chitta, and N. Correll (2014). "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study". *CoRR* abs/1404.3785. arXiv: 1404.3785.

Cordes, F. (2018). "Design and Experimental Evaluation of a Hybrid Wheeled-Leg Exploration Rover in the Context of Multi-Robot Systems". PhD thesis. Bremen, Germany: University of Bremen. URL: http://nbn-resolving.de/urn:nbn:de:gbv:46-00106941-11.

Cordes, F. and A. Babu (2016). "SherpaTT: A Versatile Hybrid Wheeled-Leg Rover". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2016), June*.

Cordes, F., A. Babu, and F. Kirchner (2017). "Static Force Distribution and Orientation Control for a Rover with an Actively Articulated Suspension System". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2017)*.

Cordes, F., A. Babu, and T. Stark (2024). "Sherpa, a family of wheeled-leg rovers". In: *Biologically Inspired Series-Parallel Hybrid Robots*. Ed. by S. Kumar, A. Mueller, and F. Kirchner. 1. Auflage. Vol. 514. Elsevier Science, pp. 281–304. ISBN: 978-0-323-88482-2. URL: https://shop.elsevier.com/books/biologically-inspired-series-parallel-hybrid-robots/kumar/978-0-323-88482-2.

Cordes, F., F. Kirchner, and A. Babu (2018). "Design and field testing of a rover with an actively articulated suspension system in a Mars analog terrain". *Journal of Field Robotics* 35:7, pp. 1149–1181. DOI: 10.1002/rob.21808. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21808.

Cordes, F., C. Oekermann, A. Babu, D. Kuehn, T. Stark, and F. Kirchner (2014). "An active suspension system for a planetary rover". In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014), June*, pp. 17–19.

Coumans, E. (2015). "Bullet Physics Simulation". In: *ACM SIGGRAPH 2015 Courses*. SIGGRAPH '15. Association for Computing Machinery, Los Angeles, California. ISBN: 9781450336345. DOI: 10.1145/2776880.2792704. URL: https://doi.org/10.1145/2776880.2792704.

Coumans, E. and Y. Bai (2016). "Pybullet, a python module for physics simulation for games, robotics and machine learning".

Cui, L., S. Wang, J. Zhang, Z. Dongsheng, J. Lai, Y. Zheng, Z. Zhang, and Z.-P. Jiang (2021). "Learning-Based Balance Control of Wheel-Legged Robots". *IEEE Robotics and Automation Letters* PP, pp. 1–1. DOI: 10.1109/LRA.2021.3100269.

Dettmann, A., Z. Wang, W. Wenzel, F. Cordes, and F. Kirchner (2011). "Heterogeneous Modules with a Homogeneous Electromechanical Interface in Multi-Module Systems for Space Exploration". In: *2011 IEEE International Conference on Robotics and Automation (ICRA'11)*. ©2011 IEEE. Reprinted with permission. Shanghai, P.R. China.

DFKI (Slam3D). https://github.com/dfki-ric/slam3d.

Donnarumma, F., R. Prevete, D. Maisto, S. Fuscone, E. Irvine, M. Meer, C. Kemere, and G. Pezzulo (2021). "A framework to identify structured behavioral patterns within rodent spatial trajectories". *Scientific Reports* 11, p. 468. DOI: 10.1038/s41598-020-79744-7.

Doty, K. L., C. Melchiorri, and C. Bonivento (1993). "A Theory of Generalized Inverses Applied to Robotics". *The International Journal of Robotics Research* 12:1, pp. 1–19. DOI: 10.1177/027836499301200101. eprint: https://doi.org/10.1177/027836499301200101. URL: https://doi.org/10.1177/027836499301200101.

Duriez, O., G. Peron, D. Gremillet, A. Sforzi, and F. Monti (2018). "Migrating ospreys use thermal uplift over the open sea". *Biology letters* 14:12, p. 20180687.

Eckstein, M. and A. Collins (2019). *Computational Evidence for Hierarchically-Structured Reinforcement Learning in Humans*. DOI: 10.1101/731752.

Eich, M., F. Grimminger, S. Bosse, D. Spenneberg, and F. Kirchner (2008a). "Asguard: A hybrid legged wheel security and sar–robot using bio–inspired locomotion for rough terrain". In: *IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance*. Benicssim, Spain.

Eich, M., F. Grimminger, and F. Kirchner (2008b). "Adaptive stair–climbing behaviour with a hybrid legged–wheeled robot". In: *11th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR)*. Coimbra, Portugal.

Eich, M., F. Grimminger, and F. Kirchner (2009). "Adaptive Compliance Control of a Multi-legged Stair-Climbing Robot Based on Proprioceptive Data". In: *In Industrial Robot: An International Journal*. volume 36 Issue 4. Emerald Group Publishing Limited, pages 331–339.

Eßer, J., N. Bach, C. Jestel, O. Urbann, and S. Kerner (2023). "Guided Reinforcement Learning: A Review and Evaluation for Efficient and Effective Real-World Robotics [Survey]". *IEEE Robotics & Automation Magazine* 30:2, pp. 67–85. DOI: 10.1109/MRA.2022.3207664.

Eßer, J., S. Kumar, H. Peters, V. Bargsten, J. d. G. Fernandez, C. Mastalli, O. Stasse, and F. Kirchner (2021). "Design, analysis and control of the series-parallel hybrid RH5 humanoid robot". In: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pp. 400–407. DOI: 10.1109/HUMANOIDS47582.2021.9555770.

Fankhauser, P. and M. Hutter (2016). "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation". In: *Robot Operating System (ROS) – The Complete Reference (Volume 1)*. Ed. by A. Koubaa. Springer. Chap. 5. ISBN: 978-3-319-26052-5. DOI: 10.1007/978-3-319-26054-9_5. URL: http://www.springer.com/de/book/9783319260525.

Fondahl, K., D. Kuehn, F. Beinersdorf, F. Bernhard, F. Grimminger, M. Schilling, T. Stark, and F. Kirchner (2012). "An adaptive sensor foot for a bipedal and quadrupedal robot". In: *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE, pp. 270–275. ISBN: 978-1-4577-1200-5. DOI: 10.1109/BioRob.2012.6290735.

Fujimoto, S., H. van Hoof, and D. Meger (2018). *Addressing Function Approximation Error in Actor-Critic Methods*. arXiv: 1802.09477 [cs.AI].

Gabrielli, G. and T. H. von Karman (1950). "What price speed?" In: *Mechanical Engineering*. Vol. 72. 10, pp. 775–781.

Gao, X. (2018). "Deep reinforcement learning for time series: playing idealized trading games". *arXiv preprint arXiv:1803.03916*.

Garcia, E. and P. G. De Santos (2005). "An improved energy stability margin for walking machines subject to dynamic effects". *Robotica* 23:1, pp. 13–20.

Garcia, E., J. Estremera, and P. G. De Santos (2002). "A classification of stability margins for walking robots". *Robotica* 20:6, pp. 595–606.

Geilinger, M., R. Poranne, R. Desai, B. Thomaszewski, and S. Coros (2018). "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels". *ACM Transactions on Graphics (TOG)* 37:4, pp. 1–12.

Giftthaler, M., M. Neunert, M. Stäuble, and J. Buchli (2018). "The Control Toolbox - An Open-Source C++ Library for Robotics, Optimal and Model Predictive Control". In: *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pp. 123–129. DOI: 10.1109/SIMPAR.2018.8376281.

Giftthaler, M., M. Neunert, M. Stäuble, J. Buchli, and M. Diehl (2017). "A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control". *arXiv preprint arXiv:1711.11006*.

Gobet, F., P. C. Lane, S. Croker, P. C. Cheng, G. Jones, I. Oliver, and J. M. Pine (2001). "Chunking mechanisms in human learning". *Trends in cognitive sciences* 5:6, pp. 236–243.

Gong, Y., R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle (2019). "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway". In: *2019 American control conference (ACC)*. IEEE, pp. 4559–4566.

Gregorio, P., M. Ahmadi, and M. Buehler (1997). "Design, Control, and Energetics of an Electrically Actuated Legged Robot". *IEEE Trans. Systems, Man, and Cybernetics* 27, pp. 626–634.

Gros, S., M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl (2016). "From linear to nonlinear MPC: bridging the gap via the real-time iteration". *International Journal of Control* 0:0, pp. 1–19.

DOI: 10.1080/00207179.2016.1222553. eprint: https://doi.org/10.1080/00207179.2016.1222553. URL: https://doi.org/10.1080/00207179.2016.1222553.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine (2018a). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". *CoRR* abs/1801.01290. arXiv: 1801.01290.

Haarnoja, T., A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine (2018b). "Soft Actor-Critic Algorithms and Applications". *CoRR* abs/1812.05905. arXiv: 1812.05905.

Hackert, R., H. Witte, and M. S. Fischer (2006). "Interactions between Motions of the Trunk and the Angle of Attack of the Forelimbs in Synchronous Gaits of the Pika (Ochotona rufescens)". In: *Adaptive Motion of Animals and Machines*. Springer-Verlag Tokyo, pp. 69–77.

Hauser, K., T. Bretl, J. C. Latombe, and B. Wilcox (2008a). "Motion planning for a six-legged lunar robot". *Springer Tracts in Advanced Robotics* 47, pp. 301–316. ISSN: 1610-7438. DOI: 10.1007/978-3-540-68405-3_19.

Hauser, K., T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox (2008b). "Motion Planning for Legged Robots on Varied Terrain". *The International Journal of Robotics Research* 27:11-12, pp. 1325–1349. ISSN: 0278-3649. DOI: 10.1177/0278364908098447.

Haynes, G. C., D. Stager, A. Stentz, J. M. Vande Weghe, B. Zajac, H. Herman, A. Kelly, E. Meyhofer, D. Anderson, D. Bennington, J. Brindza, D. Butterworth, C. Dellin, M. George, J. Gonzalez-Mora, M. Jones, P. Kini, M. Laverne, N. Letwin, E. Perko, C. Pinkston, D. Rice, J. Scheifflee, K. Strabala, M. Waldbaum, and R. Warner (2017). "Developing a Robust Disaster Response Robot: CHIMP and the Robotics Challenge". *Journal of Field Robotics* 34:2, pp. 281–304. DOI: https://doi.org/10.1002/rob.21696. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21696. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21696.

Henze, B., A. Dietrich, and C. Ott (2015). "An approach to combine balancing with hierarchical whole-body control for legged humanoid robots". *IEEE Robotics and Automation Letters* 1:2, pp. 700–707.

Hidalgo-Carrio, J., A. Babu, and F. Kirchner (2014). "Static forces weighted Jacobian motion models for improved Odometry". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, pp. 169–175.

Hirose, S., H. Tsukagoshi, and K. Yoneda (1998). "Normalized energy stability margin: generalized stability criterion for walking vehicles". In: *Proc. Int. Conf. Climbing and Walking Robots*, pp. 71–76.

Högemann, P. (2008). *Biologically Inspired Optimization of the Hybrid Quadruped Asguard Robot's Legged-wheel Design*.

Hornung, A., K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard (2013). "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". *Autonomous Robots*. Software available at https://octomap.github.io. DOI: 10.1007/s10514-012-9321-0. URL: https://octomap.github.io.

Hosseini, M., D. Rodriguez, and S. Behnke (2023). "Dynamic Hybrid Locomotion and Jumping for Wheeled-Legged Quadrupeds". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 793–799.

Huang, S. and S. Ontañón (2022). "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms". *The International FLAIRS Conference Proceedings* 35. ISSN: 2334-0762. DOI: 10.32473/flairs.v35i.130584. URL: http://dx.doi.org/10.32473/flairs.v35i.130584.

Huntsberger, T., A. Stroupe, H. Aghazarian, M. Garrett, P. Younse, and M. Powell (2007). "TRESSA: Teamed robots for exploration and science on steep areas". *Journal of Field Robotics* 24:11-12, pp. 1015–1031. ISSN: 1556-4967. DOI: 10.1002/rob.20219. URL: http://dx.doi.org/10.1002/rob.20219.

Hutsebaut-Buysse, M., K. Mets, and S. Latré (2022). "Hierarchical Reinforcement Learning: A Survey and Open Research Challenges". *Machine Learning and Knowledge Extraction* 4:1, pp. 172–221. ISSN: 2504-4990. DOI: 10.3390/make4010009. URL: https://www.mdpi.com/2504-4990/4/1/9.

Hutter, M., P. Leemann, G. Hottiger, R. Figi, S. Tagmann, G. Rey, and G. Small (2017). "Force Control for Active Chassis Balancing". *IEEE/ASME Transactions on Mechatronics* 22:2, pp. 613–622. ISSN: 1083-4435. DOI: 10.1109/TMECH.2016.2612722.

Iagnemma, K., A. Rzepniewski, S. Dubowsky, and P. Schenker (2003). "Control of Robotic Vehicles with Actively Articulated Suspensions in Rough Terrain". *Autonomous Robots* 14:1, pp. 5–16. ISSN: 1573-7527. DOI: 10.1023/A:1020962718637.

Ijspeert, A. J. (2008). "Central pattern generators for locomotion control in animals and robots: a review". *Neural networks* 21:4, pp. 642–653.

Inman, V. T. (1966). "Human locomotion". *Canadian Medical Association Journal* 94:20, p. 1047.

Javadi, M., D. Harnack, P. Stocco, S. Kumar, S. Vyas, D. Pizzutilo, and F. Kirchner (2023). "AcroMonk: A Minimalist Underactuated Brachiating Robot". *IEEE Robotics and Automation Letters* 8:6, pp. 3637–3644. DOI: 10.1109/LRA.2023.3269296.

Jelavic, E., Y. Berdou, D. Jud, S. Kerscher, and M. Hutter (2020). "Terrain-Adaptive Planning and Control of Complex Motions for Walking Excavators". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2684–2691. DOI: 10.1109/IROS45743.2020.9341655.

Jelavic, E., F. Farshidian, and M. Hutter (2021a). "Combined Sampling and Optimization Based Planning for Legged-Wheeled Robots". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, Xi'an, China, pp. 8366–8372. DOI: 10.1109/ICRA48506.2021.9560731.

Jelavic, E. and M. Hutter (2019). "Whole-Body Motion Planning for Walking Excavators". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2292–2299. DOI: 10.1109/IROS40897.2019.8967631.

Jelavic, E., D. Jud, P. Egli, and M. Hutter (2021b). "Towards Autonomous Robotic Precision Harvesting". *arXiv preprint arXiv:2104.10110*.

Jelavic, E., K. Qu, F. Farshidian, and M. Hutter (2023). "LSTP: Long Short-Term Motion Planning for Legged and Legged-Wheeled Systems". en. *IEEE Transactions on Robotics*. ISSN: 1552-3098. DOI: 10.3929/ethz-b-000625515.

Joyeux, S. and J. Albiez (2011). "Robot development: from components to systems". In: *6th National Conference on Control Architectures of Robots*. INRIA Grenoble Rhône-Alpes. Grenoble, France, 15 p. URL: https://hal.inria.fr/inria-00599679.

Joyeux, S., J. Schwendner, F. Kirchner, A. Babu, F. Grimminger, J. Machowinski, P. Paranhos, and C. Gaudig (2011). "Intelligent Mobility". *KI - Künstliche Intelligenz* 25:2, pp. 133–139. ISSN: 1610-1987. DOI: 10.1007/s13218-011-0089-8. URL: https://doi.org/10.1007/s13218-011-0089-8.

Jud, D., S. Kerscher, M. Wermelinger, E. Jelavic, P. Egli, P. Leemann, G. Hottiger, and M. Hutter (2021). "HEAP - The autonomous walking excavator". *Automation in Construction* 129, p. 103783. ISSN: 0926-5805. DOI: 10.1016/j.autcon.2021.103783. URL: https://www.sciencedirect.com/science/article/pii/S092658052100234X.

Kanervisto, A., C. Scheller, and V. Hautamäki (2020). "Action space shaping in deep reinforcement learning". In: *2020 IEEE conference on games (CoG)*. IEEE, pp. 479–486.

Karumanchi, S., K. Edelberg, I. Baldwin, J. Nash, J. Reid, C. Bergh, J. Leichty, K. Carpenter, M. Shekels, M. Gildner, D. Newill-Smith, J. Carlton, J. Koehler, T. Dobreva, M. Frost, P. Hebert, J. Borders, J. Ma, B. Douillard, P. Backes, B. Kennedy, B. Satzinger, C. Lau, K. Byl, K. Shankar, and J. Burdick (2017). "Team RoboSimian: Semi-autonomous Mobile Manipulation at the 2015 DARPA Robotics Challenge Finals". *Journal of Field Robotics* 34:2, pp. 305–332. DOI: https://doi.org/10.1002/rob.21676. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21676. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21676.

Kingma, D. P. (2013). "Auto-encoding variational bayes". *arXiv preprint arXiv:1312.6114*.

Kirchner, F. (1998). "Q-learning of complex behaviours on a six-legged walking machine". *Robotics and autonomous systems* 25:3-4, pp. 253–262.

Klamt, T. and S. Behnke (2017). "Anytime Hybrid Driving-Stepping Locomotion Planning". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* September, pp. 4444–4451. URL: http://www.ais.uni-bonn.de/papers/IROS%7B%5C_%7D2017%7B%5C_%7DKlamt.pdf.

– (2018). "Planning Hybrid Driving-Stepping Locomotion on Multiple Levels of Abstraction". *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1695–1702. URL: https://api.semanticscholar.org/CorpusID:4945368.

Klokowski, P., J. Eßer, N. Gramse, B. Pschera, M. Plitt, F. Feldmeier, S. Bajpai, C. Jestel, N. Bach, O. Urbann, and S. Kerner (2023). "evoBOT – Design and Learning-Based Control of a Two-Wheeled Compound Inverted Pendulum Robot". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10425–10432. DOI: 10.1109/IROS55552.2023.10342128.

Komsuoglu, A. M., R. Altendorfer, N. Moore, M. Buehler, H. B. Brown, D. Mcmordie, U. Saranli, R. Full, and D. E. Koditschek (2001). "RHex: A Biologically Inspired Hexapod Runner". *Autonomous Robots* 11, pp. 207–213.

Kong, J., M. Pfeiffer, G. Schildbach, and F. Borrelli (2015). "Kinematic and dynamic vehicle models for autonomous driving control design". In: *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE, pp. 1094–1099.

Krotkov, E., D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski (2018). "The DARPA Robotics Challenge Finals: Results and Perspectives". In: *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Ed. by M. Spenko, S. Buerger, and K. Iagnemma. Springer International Publishing, Cham, pp. 1–26. ISBN: 978-3-319-74666-1. DOI: 10.1007/978-3-319-74666-1_1. URL: https://doi.org/10.1007/978-3-319-74666-1_1.

Kuehn, D., M. Schilling, T. Stark, M. Zenzes, and F. Kirchner (2017). "System design and testing of the hominid robot charlie". *Journal of Field Robotics* 34:4, pp. 666–703.

Laurenzi, A., E. M. Hoffman, and N. G. Tsagarakis (2018). "Quadrupedal walking motion and footstep placement through Linear Model Predictive Control". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2267–2273. DOI: 10.1109/IROS.2018.8593692.

Lee, J., M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter (2024). "Learning robust autonomous navigation and locomotion for wheeled-legged robots". *Science Robotics* 9:89, eadi9641. DOI: 10.1126/scirobotics.adi9641. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.adi9641. URL: https://www.science.org/doi/abs/10.1126/scirobotics.adi9641.

Levenberg, K. (1944). "A method for the solution of certain non-linear problems in least squares". *Quarterly of applied mathematics* 2:2, pp. 164–168.

Li, J., H. Gao, Y. Wan, J. Humphreys, C. Peers, H. Yu, and C. Zhou (2022). "Whole-Body Control for a Torque-Controlled Legged Mobile Manipulator". *Actuators* 11:11. ISSN: 2076-0825. DOI: 10.3390/act11110304. URL: https://www.mdpi.com/2076-0825/11/11/304.

Li, P., Y. Pei, and J. Li (2023). "A comprehensive survey on design and application of autoencoder in deep learning". *Applied Soft Computing* 138, p. 110176. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2023.110176. URL: https://www.sciencedirect.com/science/article/pii/S1568494623001941.

Li, T., N. Lambert, R. Calandra, F. Meier, and A. Rai (2020). "Learning generalizable locomotion skills with hierarchical reinforcement learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 413–419.

Li, W. and E. Todorov (2004). "Iterative linear quadratic regulator design for nonlinear biological movement systems." In: *ICINCO (1)*, pp. 222–229.

Liechti, F. (2006). "Birds: blowin'by the wind?" *Journal of Ornithology* 147, pp. 202–211.

Likhachev, M., G. Gordon, and S. Thrun (2003). "ARA*: Anytime A* with Provable Bounds on Sub-Optimality". In: vol. 16.

Lim, J., I. Lee, I. Shim, H. Jung, H. M. Joe, H. Bae, O. Sim, J. Oh, T. Jung, S. Shin, K. Joo, M. Kim, K. Lee, Y. Bok, D.-G. Choi, B. Cho, S. Kim, J. Heo, I. Kim, J. Lee, I. S. Kwon, and J.-H. Oh (2017). "Robot System of DRC-HUBO+ and Control Strategy of Team KAIST in DARPA Robotics Challenge Finals". *Journal of Field Robotics* 34:4, pp. 802–829. DOI: https://doi.org/10.1002/rob.21673. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21673. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21673.

Limpert, N., S. Schiffer, and A. Ferrein (2015). "A local planner for Ackermann-driven vehicles in ROS SBPL". In: *2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, pp. 172–177. DOI: 10.1109/RoboMech.2015.7359518.

Liu, M., R. Lober, and V. Padois (2016). "Whole-body hierarchical motion and force control for humanoid robots". *Autonomous Robots* 40, pp. 493–504.

Lu, D., E. Dong, C. Liu, M. Xu, and J. Yang (2013). "Design and development of a leg-wheel hybrid robot "HyTRo-I"". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6031–6036. DOI: 10.1109/IROS.2013.6697232.

Macenski, S., F. Martin, R. White, and J. Ginés Clavero (2020). "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Machowinski, J. (2009). *Dynamisches Laufen mit dem System ASGuard*. Germany.

Machowinski, J., A. Böckmann, S. Arnold, C. Hertzberg, and S. Planthaber (2017). "Climbing steep inclines with a six-legged robot using locomotion planning". In: *International conference on robotics and automation*. Vol. 29.

Marder, E. and D. Bucher (2001). "Central pattern generators and the control of rhythmic movements". *Current biology* 11:23, R986–R996.

Marquardt, D. W. (1963). "An algorithm for least-squares estimation of nonlinear parameters". *Journal of the society for Industrial and Applied Mathematics* 11:2, pp. 431–441.

McGeer, T. (1990). "Passive Dynamic Walking". *The International Journal of Robotics Research* 9:2, pp. 62–82. DOI: 10.1177/027836499000900206. eprint: https://doi.org/10.1177/027836499000900206. URL: https://doi.org/10.1177/027836499000900206.

Medeiros, V. S., E. Jelavic, M. Bjelonic, R. Siegwart, M. A. Meggiolaro, and M. Hutter (2020). "Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Driving in Challenging Terrain". *IEEE Robotics and Automation Letters* 5:3, pp. 4172–4179. DOI: 10.1109/LRA.2020.2990720.

Messuri, D. A. (1985). "Optimization of the locomotion of a legged vehicle with respect to maneuverability (robot, walking, hexapod, stability)". PhD thesis. The Ohio State University.

Micaelli, A. and C. Samson (1993). *Trajectory tracking for unicycle-type and two-steering-wheels mobile robots*. Research Report RR-2097. INRIA. URL: https://hal.inria.fr/inria-00074575.

Moore, E. Z., D. Campbell, F. Grimminger, and M. Buehler (2002). "Reliable Stair Climbing in the Simple Hexapod 'RHex'". In: *IEEE Int. Conf. on Robotics and Automation (ICRA), Washington DC, USA*.

NASA (2009). URL: https://www.nasa.gov/audience/foreducators/robotics/imagegallery/r_athlete.jpg.html.

Neftci, E. O. and B. B. Averbeck (2019). "Reinforcement learning in artificial and biological systems". *Nature Machine Intelligence* 1, pp. 133–143. URL: https://api.semanticscholar.org/CorpusID:189473795.

Nesnas, I. A., J. B. Matthews, P. Abad-Manterola, J. W. Burdick, J. A. Edlund, J. C. Morrison, R. D. Peters, M. M. Tanner, R. N. Miyake, B. S. Solish, and R. C. Anderson (2012). "Axel and DuAxel rovers for the sustainable exploration of extreme terrains". *Journal of Field Robotics*. ISSN: 1556-4967. DOI: 10.1002/rob.21407. URL: http://dx.doi.org/10.1002/rob.21407.

Oh, S., T. Kim, and J. Song (2023). "Bouc–Wen class models considering hysteresis mechanism of RC columns in nonlinear dynamic analysis". *International Journal of Non-Linear Mechanics* 148, p. 104263. ISSN: 0020-7462. DOI: https://doi.org/10.1016/j.ijnonlinmec.2022.104263. URL: https://www.sciencedirect.com/science/article/pii/S0020746222002335.

Orin, D. E., A. Goswami, and S.-H. Lee (2013). "Centroidal dynamics of a humanoid robot". *Autonomous robots* 35, pp. 161–176.

Paden, B., M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli (2016). "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". *IEEE Transactions on Intelligent Vehicles* 1:1, pp. 33–55. ISSN: 2379-8904. DOI: 10.1109/TIV.2016.2578706.

Pan, Z., B. Li, H. Jing, Z. Niu, and R. Wang (2023). "Wheel-Leg Collaborative Control for Wheel-legged Robots Based on MPC with Preview". In: *2023 IEEE International Automated Vehicle Validation Conference (IAVVC)*. IEEE, pp. 1–8.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Patankar, S., C. Usakoyal, P. Patil, and K. Raut (2024). "A Survey of Deep Reinforcement Learning in Game Playing". In: *2024 MIT Art, Design and Technology School of Computing International Conference (MITADTSoCiCon)*, pp. 1–5. DOI: 10.1109/MITADTSoCiCon60330.2024.10575819.

Pateria, S., B. Subagdja, A.-h. Tan, and C. Quek (2021). "Hierarchical Reinforcement Learning: A Comprehensive Survey". *ACM Comput. Surv.* 54:5. ISSN: 0360-0300. DOI: 10.1145/3453160. URL: https://doi.org/10.1145/3453160.

Peng, X. B., G. Berseth, K. Yin, and M. Van De Panne (2017). "DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning". *ACM Trans. Graph.* 36:4. ISSN: 0730-0301. DOI: 10.1145/3072959.3073602.

Pengfei, W., H. Bo, and S. Lining (2005). "Walking research on multi-motion mode quadruped bionic robot based on moving ZMP". In: *IEEE International Conference Mechatronics and Automation, 2005*. Vol. 4, 1935–1940 Vol. 4. DOI: 10.1109/ICMA.2005.1626858.

Perlin, K. (1985). "An image synthesizer". *ACM Siggraph Computer Graphics* 19:3, pp. 287–296.

– (2002). "Improving noise". In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 681–682.

Petereit, J., J. Beyerer, T. Asfour, S. Gentes, B. Hein, U. D. Hanebeck, F. Kirchner, R. Dillmann, H. H. Götting, M. Weiser, M. Gustmann, and T. Egloffstein (2019). "ROBDEKON: Robotic Systems

for Decontamination in Hazardous Environments". In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 249–255. DOI: 10.1109/SSRR.2019.8848969.

Poulakakis, I., J. A. Smith, and M. Buehler (2006). "On the Dynamics of Bounding and Extensions: Towards the Half-Bound and Gallop Gaits". In: *Adaptive Motion of Animals and Machines*. Springer-Verlag Tokyo, pp. 79–88.

Precup, D. and R. S. Sutton (2000). "Temporal abstraction in reinforcement learning". AAI9978540. PhD thesis. ISBN: 0599844884.

Raffin, A., A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann (2019). *Stable Baselines3*. https://github.com/DLR-RM/stable-baselines3.

Ranjan, R., V. M. Patel, and R. Chellappa (2017). "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition". *IEEE transactions on pattern analysis and machine intelligence* 41:1, pp. 121–135.

Ranzato, M., F. J. Huang, Y.-L. Boureau, and Y. LeCun (2007). "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. DOI: 10.1109/CVPR.2007.383157.

Rasmussen, D. and C. Eliasmith (2014). "A neural model of hierarchical reinforcement learning". In:

Reher, J. P., A. Hereid, S. Kolathaya, C. M. Hubicki, and A. D. Ames (2020). "Algorithmic foundations of realizing multi-contact locomotion on the humanoid robot DURUS". In: *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Springer, pp. 400–415.

Reid, W., R. Fitch, A. H. Göktoğan, and S. Sukkarieh (2020). "Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover". *Journal of Field Robotics* 37:5, pp. 786–811.

Reid, W., A. H. Göktogan, and S. Sukkarieh (2014). "Moving mammoth: Stable motion for a reconfigurable wheel-on-leg rover". In: *Proceedings of Australasian Conference on Robotics and Automation*, pp. 1–10.

Ribas Fernandes, J., A. Solway, C. Diuk, J. McGuire, A. Barto, Y. Niv, and M. Botvinick (2011). "Neuron Article A Neural Signature of Hierarchical Reinforcement Learning". *Neuron* 71, pp. 370–9. DOI: 10.1016/j.neuron.2011.05.042.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, Cambridge, MA, USA, pp. 318–362. ISBN: 026268053X.

Russo, M. and M. Ceccarelli (2020). "A survey on mechanical solutions for hybrid mobile robots". *Robotics* 9:2, p. 32.

Saranli, U., A. A. Rizzi, and D. E. Koditschek (2004a). "Model-Based Dynamic Self-Righting Maneuvers for a Hexapedal Robot". *International Journal of Robotics Research* 23:9, pp. 903–918.

Saranli, U., M. Buehler, and D. E. Koditschek (2000). "Design, modeling and preliminary control of a compliant hexapod robot". In: *IEEE Int. Conf. Robotics and Automation*, pp. 2589–2596.

– (2001). "RHex: A simple and highly mobile hexapod robot". *International Journal of Robotics Research* 20, pp. 616–631.

Saranli, U. and D. E. Koditschek (2010). "Design and analysis of a flipping controller for RHex". *Science* 1001, pp. 48109–2122.

Saranli, U., A. A. Rizzi, and D. E. Koditschek (2004b). "Model-based dynamic self-righting maneuvers for a hexapedal robot". *The International Journal of Robotics Research* 23:9, pp. 903–918.

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

Schwarke, C., V. Klemm, M. v. d. Boon, M. Bjelonic, and M. Hutter (2023). "Curiosity-Driven Learning of Joint Locomotion and Manipulation Tasks". In: *Proceedings of The 7th Conference on Robot Learning*. Ed. by J. Tan, M. Toussaint, and K. Darvish. Vol. 229. Proceedings of Machine Learning Research. PMLR, pp. 2594–2610. URL: https://proceedings.mlr.press/v229/schwarke23a.html.

Schwarz, M., T. Rodehutskors, M. Schreiber, and S. Behnke (2016). "Hybrid driving-stepping locomotion with the wheeled-legged robot Momaro". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5589–5595. DOI: 10.1109/ICRA.2016.7487776.

Schwendner, J., S. Joyeux, and F. Kirchner (2014). "Using embodied data for localization and mapping". *Journal of Field Robotics* 31:2, pp. 263–295.

Sentis, L. and O. Khatib (2006). "A whole-body control framework for humanoids operating in human environments". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 2641–2648.

Shahid, A. A., D. Piga, F. Braghin, and L. Roveda (2022). "Continuous control actions learning and adaptation for robotic manipulation through reinforcement learning". *Autonomous Robots* 46:3, pp. 483–498.

Shao, K., Z. Tang, Y. Zhu, N. Li, and D. Zhao (2019). *A Survey of Deep Reinforcement Learning in Video Games*. arXiv: 1912.10944 [cs.MA]. URL: https://arxiv.org/abs/1912.10944.

Shi, J., T. Dear, and S. D. Kelly (2020). "Deep Reinforcement Learning for Snake Robot Locomotion". *IFAC-PapersOnLine* 53:2. 21st IFAC World Congress, pp. 9688–9695. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.2619. URL: https://www.sciencedirect.com/science/article/pii/S2405896320333772.

Siciliano, B. and O. Khatib (2008). *Springer Handbook of Robotics*. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-30301-5.

Siegwart, R. and I. R. Nourbakhsh (2004). *Introduction to Autonomous Mobile Robots*. Intelligent Robotics and Autonomous Agents. MIT Press. ISBN: 9780262195027. URL: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=122550&site=ehost-live.

Singer, N. C. and W. P. Seering (1989). "Design and comparison of command shaping methods for controlling residual vibration". In: Cited by: 56, pp. 888–893. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0024863321&partnerID=40&md5=55ede90207c6380abcd76bb0f566772a.

Sonsalla, R., F. Cordes, L. Christensen, S. Planthaber, J. Albiez, I. Scholz, and F. Kirchner (2014). "Towards a Heterogeneous Modular Robotic Team in a Logistic Chain for Extraterrestrial Exploration". In: *12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'14)*. Montreal, Canada. URL: https://www.researchgate.net/publication/265085869_Towards_a_Heterogeneous_Modular_Robotic_Team_in_a_Logistic_Chain_for_Extraterrestrial_Exploration.

Sonsalla, R., F. Cordes, L. Christensen, T. M. Roehr, T. Stark, S. Planthaber, M. Maurus, M. Mallwitz, and E. A. Kirchner (2017). "Field Testing of a Cooperative Multi-Robot Sample Return Mission in Mars Analogue Environment". In: *Proceedings of the 14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 20017). ESA/Estec Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA-2017), 14th, June 20-22, Leiden, Netherlands*. ESA/ESTEC. ESA. URL: https://www.dfki.de/fileadmin/user_upload/import/9091_3A-sonsalla.pdf.

Sonsalla, R. U., J. B. Akpo, and F. Kirchner (2015). "Coyote III: Development of a modular and highly mobile micro rover". In: *Proc. of the 13th Symp. on Advanced Space Technologies in Robotics and Automation (ASTRA-2015)*.

Stolle, M. and D. Precup (2002). "Learning Options in Reinforcement Learning". In: *Abstraction, Reformulation, and Approximation*. Ed. by S. Koenig and R. C. Holte. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 212–223. ISBN: 978-3-540-45622-3.

Şucan, I. A., M. Moll, and L. E. Kavraki (2012). "The Open Motion Planning Library". *IEEE Robotics & Automation Magazine* 19:4. https://ompl.kavrakilab.org, pp. 72–82. DOI: 10.1109/MRA.2012.2205651.

Sun, C., G. Yang, S. Yao, Q. Liu, J. Wang, and X. Xiao (2023). "RHex-T3: A Transformable Hexapod Robot With Ladder Climbing Function". *IEEE/ASME Transactions on Mechatronics* 28:4, pp. 1939–1947. DOI: 10.1109/TMECH.2023.3276756.

Sun, J., Y. You, X. Zhao, A. H. Adiwahono, and C. M. Chew (2020). "Towards More Possibilities: Motion Planning and Control for Hybrid Locomotion of Wheeled-Legged Robots". *IEEE Robotics and Automation Letters* 5:2, pp. 3723–3730. DOI: 10.1109/LRA.2020.2979626.

Tadakuma, K., R. Tadakuma, A. Maruyama, E. Rohmer, K. Nagatani, K. Yoshida, A. Ming, M. Shimojo, M. Higashimori, and M. Kaneko (2010). "Mechanical design of the Wheel-Leg hybrid mobile robot to realize a large wheel diameter". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3358–3365. DOI: 10.1109/IROS.2010.5651912.

Talebi, S., I. Poulakakis, E. Papadopoulos, and M. Buehler (2000). "Quadruped Robot Running With A Bounding Gate". In: *Proc. 7 th Int. Symp. on Experimental Robotic (ISER'00)*. Springer-Verlag, pp. 281–289.

Tan, W., X. Fang, W. Zhang, R. Song, T. Chen, Y. Zheng, and Y. Li (2021). "A Hierarchical Framework for Quadruped Locomotion Based on Reinforcement Learning". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8462–8468. DOI: 10.1109/IROS51168.2021.9636757.

Tang, C.-Y., C.-H. Liu, W.-K. Chen, and S. D. You (2020). "Implementing action mask in proximal policy optimization (PPO) algorithm". *ICT Express* 6:3, pp. 200–203. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2020.05.003. URL: https://www.sciencedirect.com/science/article/pii/S2405959520300746.

Tarokh, M., H. D. Ho, and A. Bouloubasis (2013). "Systematic kinematics analysis and balance control of high mobility rovers over rough terrain". *Robotics and Autonomous Systems* 61:1, pp. 13–24. ISSN: 0921-8890. DOI: 10.1016/j.robot.2012.09.010. URL: http://www.sciencedirect.com/science/article/pii/S0921889012001649.

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. (2019). "Grandmaster level in StarCraft II using multi-agent reinforcement learning". *nature* 575:7782, pp. 350–354.

Viragh, Y. de, M. Bjelonic, C. D. Bellicoso, F. Jenelten, and M. Hutter (2019). "Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Using Linearized ZMP Constraints". *IEEE Robotics and Automation Letters* 4:2, pp. 1633–1640. DOI: 10.1109/LRA.2019.2896721.

Vollenweider, E., M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter (2023). "Advanced Skills through Multiple Adversarial Motion Priors in Reinforcement Learning". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5120–5126. DOI: 10.1109/ICRA48891.2023.10160751.

Vukobratović, M. and B. Borovac (2004). "Zero-moment point—thirty five years of its life". *International journal of humanoid robotics* 1:01, pp. 157–173.

Wang, J. and E. Olson (2016). "AprilTag 2: Efficient and robust fiducial detection". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Wei, Z., J. Zhang, Y. Yang, S. Xiang, H. Sun, and A. Song (2023). "Design, Control and Simulation of a Leg-Wheel Robot: STransleg". In: *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*, pp. 234–238. DOI: 10.1109/ICCCR56747.2023.10193990.

Weingarten, J. D., G. A. D. Lopes, M. Buehler, R. E. Groff, and D. E. Koditschek (2004). "Automated Gait Adaptation for Legged Robots". In: *IEEE Int. Conf. Robotics and Automation (ICRA), New Orleans, LA*.

Wen, Y.-K. (1976). "Method for random vibration of hysteretic systems". *Journal of the engineering mechanics division* 102:2, pp. 249–263.

Westervelt, E. R., J. W. Grizzle, and D. E. Koditschek (2003). "Hybrid zero dynamics of planar biped walkers". *IEEE transactions on automatic control* 48:1, pp. 42–56.

Wilcox, B. H., T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirota, and B. Cooper (2007). "Athlete: A cargo handling and manipulation robot for the moon". *Journal of Field Robotics* 24:5, pp. 421–434. DOI: https://doi.org/10.1002/rob.20193. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20193. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20193.

Wirkus, M., J. Babel, and C. Backe (2024a). "Advantages of Active and Passive Suspension Systems in Obstacle Negotiation for Planetary Rovers". In: *Walking Robots into Real World*. Ed. by K. Berns, M. O. Tokhi, A. Roennau, M. F. Silva, and R. Dillmann. Springer Nature Switzerland, Cham, pp. 169–180. ISBN: 978-3-031-71301-9.

Wirkus, M., S. Hinck, C. Backe, J. Babel, V. Riedel, N. Reichert, A. Kolesnikov, T. Stark, J. Hilljegerdes, H. D. Küçüker, et al. (2024b). "Comparative study of soil interaction and driving characteristics of different agricultural and space robots in an agricultural environment". *Journal of Field Robotics* 41:6, pp. 2009–2042.

Woock, P. and A. Babu (2022). "Autonome Robotersysteme in der Altlastensanierung". *Handbuch Altlastensanierung und Flächenmanagement*. Handbuch Altlastensanierung und Flächenmanagement 93. Aktualisierung, 3. Aufl.5111. Ed. by V. Franzius, M. Altenbockum, and T. Gerhold.

Wu, Y.-C., B.-H. Tseng, and C. E. Rasmussen (2020). "Improving Sample-Efficiency in Reinforcement Learning for Dialogue Systems by Using Trainable-Action-Mask". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8024–8028. DOI: 10.1109/ICASSP40776.2020.9053235.

Yamagata, T. and R. Santos-Rodriguez (2024). *Safe and Robust Reinforcement Learning: Principles and Practice*. arXiv: 2403.18539 [cs.LG]. URL: https://arxiv.org/abs/2403.18539.

Yang, S., M. Barlow, T. Townsend, X. Liu, D. Samarasinghe, E. Lakshika, G. Moy, T. Lynar, and B. Turnbull (2023). "Reinforcement Learning Agents Playing Ticket to Ride–A Complex Imperfect Information Board Game With Delayed Rewards". *IEEE Access* 11, pp. 60737–60757. DOI: 10.1109/ACCESS.2023.3287100.

Ye, D., Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo, et al. (2020). "Mastering complex control in moba games with deep reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04, pp. 6672–6679.

Yu, C., A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. WU (2022). "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., pp. 24611–24624. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

Yu, J., Z. Zhu, J. Lu, S. Yin, and Y. Zhang (2023). "Modeling and MPC-Based Pose Tracking for Wheeled Bipedal Robot". *IEEE Robotics and Automation Letters*.

Zahavy, T., M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor (2018). "Learn what not to learn: Action elimination with deep reinforcement learning". *Advances in neural information processing systems* 31.