Trajectory Optimization Inspired Pneumatic Locomotion on Compliant Terrains

Jacques Meyer

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University.

Supervisor: Dr C. Fisher
Department of Electrical and Electronic Engineering

April 2022

Acknowledgements

I would like to thank my supervisor, Dr. Callen Fisher, for his guidance and enthusiasm throughout my research. I would also like to thank my life partner, Lisa, for her companionship and unconditional support. Thanks to my research partner Dean for showing me the value (or non-value) of crypto-currencies. And finally thanks to my family and friends for being there for me and for reminding me to have a good time during my research.

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

April 2022

Abstract

English

In order to achieve true autonomy, robots have to be able to handle complex and rough terrain generally found outside of the lab. Legged robotics has become the focal point in recent years, aided by the developments in trajectory optimization methods. However, a major problem in legged robotics is dealing with hybrid contacts with different terrain types. The difference in dynamics due to the interaction between the foot and the ground makes it increasingly difficult to design controllers that successfully execute on multiple surfaces.

This work investigates trajectory optimization methods for a pneumatically actuated mono-pod on rigid and compliant terrain. Trajectory optimization was utilized to obtain trajectories for acceleration, steady-state and deceleration hopping on compliant terrain, as well as rigid terrain surfaces. For the compliant terrain trajectories a novel method was developed to model the specific characteristics of the compliant terrain. Trajectories were generated using this method for two different compliant terrain types, namely: rough gravel and fine gravel.

To mimic the pneumatic actuation in the trajectory optimization problem, a simplified mathematical model was developed to accommodate the bang-bang force of the pneumatic actuator. This model used complementarity constraints and node bucketing techniques to mimic the behaviour of a real pneumatic actuator with damping and delay.

Once these methods and models were implemented, trajectories were executed, in open-loop, on a fixed body robotic platform that was designed and built for this thesis. The executions were compared to the trajectory results. The rigid terrain trajectories were executed successfully on a hard surface, but failed on gravel surfaces. The compliant terrain trajectories executed successfully on gravel surfaces, indicating that the method developed to model compliant terrain is a more accurate representation of the gravel surfaces compared to the rigid terrain trajectories.

After these results showed that the methods used to describe the compliant terrain proved to be accurate, a free body mono-pod robot and support rig was designed and built. The support rig limited the movement of the mono-pod to the sagittal plane to mimic the limitations of the trajectory optimization model. Acceleration, steady-state and deceleration trajectories were generated for the free body mono-pod on compliant terrain surfaces and a rigid terrain surface. From these trajectories a controller was designed with the main sources of feedback being the height of the robot and the angle of the free

moving body of the robot.

The free body mono-pod robot used the controller to execute hopping from rest back to rest with three steady-state hops in between. For each terrain type the controller was adjusted based on the generated trajectories. The results show successful execution of the trajectories on all terrain types using the controller. Lastly multi-surface hopping was executed on the mono-pod robot platform. The controller was adjusted to hop from a hard surface to a compliant surface and executed these trajectories successfully.

Afrikaans

Om ware outonomie te bewerkstellig, moet robotte komplekse en rowwe terreine, wat gewoonlik buite die laboratorium voorkom, kan hanteer. Been aangedrewe robotika het die afgelope paar jaar 'n fokuspunt geword in die literatuur, aangehelp deur ontwikkelinge in trajek-optimaliseringsmetodes. 'n Groot probleem in been aangedrewe robotika is egter die hantering van verskillende terreintipes. Die verskil in dinamika as gevolg van die interaksie tussen die voet en die grond maak dit steeds moeiliker om beheerders te ontwerp wat suksesvol op verskeie oppervlaktes uitgevoer kan word.

Hierdie werk ondersoek optimaliseringsmetodes van trajekte vir 'n pneumaties aangedrewe eenbenige robot op stewige en sagte terrein. Trajek-optimalisering is gebruik om versnelling-, bestendige- en vertraging-trajekte te genereer wat op sagte terreine sowel as stewige terreinoppervlakke spring. Hierdie trajekte is uitgevoer op 'n robot platvorm met 'n vaste liggaam en is vergelyk met die trajek resultate. Die stewige terrein trajek is suksesvol uitgevoer op 'n harde oppervlak, maar het op 'n gruisoppervlak misluk. Die sagte terrein is suksesvol uitgevoer op twee gruisoppervlaktes, wat aandui dat dit 'n meer akkurate voorstelling van die gruisoppervlakke is in vergelyking met die rigiede terrein trajek.

Om die pneumatiese kragte in die tajekoptimalisering na te boots, is 'n vereenvoudigde wiskundige model ontwikkel om die drukstootbeweging van die pneumatiese aandrywer te akkommodeer. Hierdie model het komplementariteitsbeperkings en knooppunte gebruik om die gedrag van 'n werklike pneumatiese aandrywer na te boots.

Nadat hierdie resultate getoon het dat die metodes wat gebruik is om die terrein te beskryf akkuraat is, is 'n eenbenige robot en 'n ondersteuningsplatvorm vir 'n vrye liggaam robot ontwerp en gebou. Die ondersteuningsplatvorm het die beweging van die eenbenige robot tot die sagittale vlak beperk om die beperkings van die trajekoptimaliseringsmodel na te boots. Versnelling-, bestendige- en vertragings-trajekte is gegenereer vir die eenbenige robot met 'n vrye liggaam op sagte terrein en op stewige terrein. Uit hierdie trajekte is 'n beheerder ontwerp, met die belangrikste bronne van terugvoer die hoogte van die robot en die hoek van die vry bewegende liggaam van die robot.

Die eenbenige robot het die beheerder gebruik om van rus na rus te spring, met drie bestendige hoppe tussenin. Vir elke terrein tipe is die beheerder aangepas op grond van

die gegenereerde trajekte. Die resultate toon 'n suksesvolle uitvoering van die trajekte op alle terreintipes met behulp van die beheerder.

Laastens is 'n twee-oppervlakte-hop uitgevoer op die mono-pod robotplatform. Die beheerder is aangepas om van 'n harde oppervlak na 'n sagte oppervlak te spring en hierdie trajekte was suksesvol uitgevoer.

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

$q(n)$	Generalized coordinates at node n.
λ_z	Ground reaction force in the z-direction.
λ_x	Ground reaction force in the x-direction
ϵ	Relaxation variable used in complemenarity constraints
z_{foot}	Position of robot foot in the z-domain
F_a	Prismatic Actuator applied force
θ_L	Angle of the leg measured at the hip.
θ_{FB}	Angle of the body measured at the hip.
k_{spring}	Spring constant
τ	Torque
h_m	Time constant
h_n	Time variable

Acronyms and abbreviations

CoM	Centre of Motion
DoF	Degree of Freedom
GRF	Ground Reaction Force
MCU	Micro Controller Unit
PD	Proportional Derivative controller
PEA	Parallel Elastic Actuator
PPR	Pulses Per Revolution
RT	Rigid Terrain
SEA	Series Elastic Actuator
SLIP	Spring Loaded Inverted Pendulum
QDD	Quasi-Direct Drive
EOM	Equations of Motion

Chapter 1

Introduction

Legged robots have been identified as having advantages over other forms of terrestrial locomotion such as wheeled or treaded locomotion. Legged locomotion on compliant terrain, such as sand or gravel, has been a primary challenge in robotics in the past [1]. Due to the unpredictable nature of different terrain, legged robots struggle to maintain a consistent stable gait, with controllers becoming unstable. Accelerating, decelerating and moving at a constant velocity on different types of terrain is a necessary ability for legged robots to leave the confines of the laboratory.

Most real world applications such as search and rescue, planetary exploration and other terrestrial mobility applications include unknown terrain parameters that a robot should be able to overcome. The current focus in the literature is on steady-state locomotion with energy efficiency being the primary objective [2] [3] [4], with little attention given to transient motions. The reason for this is that transient motions have complex dynamics that are hard to solve. Additionally the controller design for transient motions is challenging.

Figure 1.1: The progress of legged robotics over the past 40 years, from an early one legged robot [4], to a Boston Dynamics Spot robot [5].

1.1. Motivation

Legged locomotion has come a long way in the past 40 years. From the one legged hoppers that Marc Raibert built in 1984 [4], to today where Boston Dynamics are releasing a commercially viable quadruped robot called Spot [5], both are shown in Figure 1.1. Yet we are still far away from reaching the agility and mobility of legged animals. This gap between robotic locomotion and animal locomotion becomes even more apparent when the terrain becomes unknown. Animals are quick to adapt to terrain that they have never faced before; they can accelerate and decelerate rapidly on different surfaces whereas current legged robot systems struggle to deal with the uncertainties of unknown terrain.

In the past researchers oversimplified models by using massless springs as legs or having infinite friction to negate slipping [6]. This was purely because the computers that were used to simulate these problems were incapable of dealing with the complexities of fully descriptive or accurate mathematical models. This lead to solutions that might work in laboratory environments, but struggled to bridge the gap between the laboratory and the real world. As computational power increases, new avenues for allowing robust legged locomotion solutions arise and should be investigated. Increasingly complex models do not need to be oversimplified and computers can therefore produce more robust solutions.

The use of trajectory optimization has resulted in impressive solutions to open loop control. Most of the work done in the past has been on hard contact surfaces [7] [8]. Trajectory optimization inspired controller design can also be a good way to generate closed loop solutions to non-linear movements such as acceleration and deceleration [9]. But how can we use trajectory optimization to find viable solutions to legged locomotion on different types of compliant terrain? And can we develop trajectory optimization inspired controllers that adapt to different terrain properties by implementing gait library switching? These questions will be addressed in this research.

Pneumatic actuation has been used in legged robots in the past in [4] [10]. It is considered mechanically simpler and cheaper than a brushless DC motor leg design as seen on the robot Solo in [11]. Using a pneumatic cylinder in combination with a torque motor at the hip is ideal for rapid acceleration and deceleration. The pneumatic leg and motor hip configuration represents the spring-loaded inverted pendulum (SLIP) model more closely [12], but this type of robot design also presents a challenge since the pneumatic actuator lacks precise force or position control (bang-bang force actuation).

1.2. Objectives

The work done in this thesis is a continuation of the work done in the PhD of Dr Fisher [9]. The same technique for developing trajectory optimization inspired controllers for legged robots is used. However the work in Dr Fisher's PhD does not account for any change in terrain dynamics and uses a pneumatic actuator that can only extend and not contract, whereas the work in this thesis develops specific techniques for compliant terrain modelling and uses a pneumatic actuator that can extend and contract. The work done in this thesis extends the initial techniques developed by Dr Fisher and allows for more real world applications of these techniques. The primary objective of this study is to design and build a one legged robot that traverses over compliant terrain in the sagittal plane and starts and ends at rest.

The first objective of this thesis was to build a single leg mono-pod robot platform. The body of the platform consists of a servo motor and weights to add inertia for a more stable free body. The leg consists of a pneumatic cylinder and a support structure for safety. To limit the movement of the robot leg to the sagittal plane, a support rig had to be built that held the robot in place. The support rig had to allow for different types of terrain to be placed below the foot of the robot leg, allowing testing on various compliant terrains.

The second objective of this thesis was to use trajectory optimization to generate trajectories for the mono-pod robot platform, specifically for compliant terrain. To use trajectory optimization, a model of the robot constrained to the sagittal plane had to be developed. This included specifically modelling the characteristics of the pneumatic cylinder. These characteristics include the bang-bang actuating nature of a pneumatic cylinder as well as an actuation delay. A specific technique for modelling compliant terrain contacts had to be developed as well.

The final objective of this thesis was to use the generated trajectories to to design a controller for each terrain type. The controllers were used to execute long time horizon movements on the robot platform. The data gathered by the robot and support system while executing the control was compared to the generated trajectories to confirm our experiments.

1.3. Scope and Limitations

This thesis is a Masters thesis and therefore a recommended time limit of 24 months was placed on this project. The scope of the project coincided with the time limit to ensure

realistic goal achievement within the allotted time frame. It was also encouraged to use readily available and reasonably priced components when possible since the budget of the project had certain limits.

Some physical limitations were enforced on the robot leg and support rig to ensure successful experiments using the available time and resources. The robot leg did not have to work outside of the laboratory environment without the support rig. The power sources, electronic or pneumatic, did not have to be carried by the robot leg. Optical encoder sensors were attached to critical moving parts of the support rig and the robot to observe the vertical position, horizontal position and angle of the robot body. It was not expected that the robot would need to execute stable control indefinitely and the vertical and horizontal movement of the robot were limited by the support rig.

The use of trajectory optimization also called for certain limitations to the optimization model to ensure time realistic and feasible solutions. The number of nodes where kept small. This was done to ensure that the complex problems did not take to long to solve. A friction coefficient, $\mu = 1$, was assumed at contact. This was done to limit the amount of analyses needed around the different contact surfaces. The epsilon value, ϵ, that determined the accuracy of the contact solution was limited to $\epsilon > 0.001$. This also allowed time realistic solutions while still obtaining a high degree of accuracy.

1.4. Plan of Development

This thesis starts of with a literature review of legged robotics (Chapter 2). In the literature review it covers different robotic platforms developed for research and commercial use. It focuses in on different robot leg actuation methods and different support rig designs. It also covers past attempts at dealing with compliant terrain during legged locomotion. The use of trajectory optimization techniques for legged robotics in the past is also detailed.

Chapter 3 explains how trajectory optimization was used to generate feasible trajectories for a one legged robot. The chapter specifically focuses on the methods used to realistically model rigid terrain, compliant terrain and pneumatic actuation.

The design and building of the real world robot and support rig is detailed in Chapter 4. Details around each component on the robot as well as the support rig are included. Certain critical design choices are pointed out and discussed. Photos of the complete robot and support rig are shown.

Chapters 5 and 6 detail how first a fixed body robot leg and then a free body robot leg

was implemented in the real world. These chapters detail what techniques were used to allow the robot to move a specified distance effectively. The results of the real world robot testing are also compared to the simulations and any discrepancies are pointed out.

Lastly any conclusions and future recommendations on this work is done in Chapter 7.1. Where specific comments are made around the effectiveness of the techniques used in this thesis. In Figure 1.2 a visual representation of the report structure is shown.

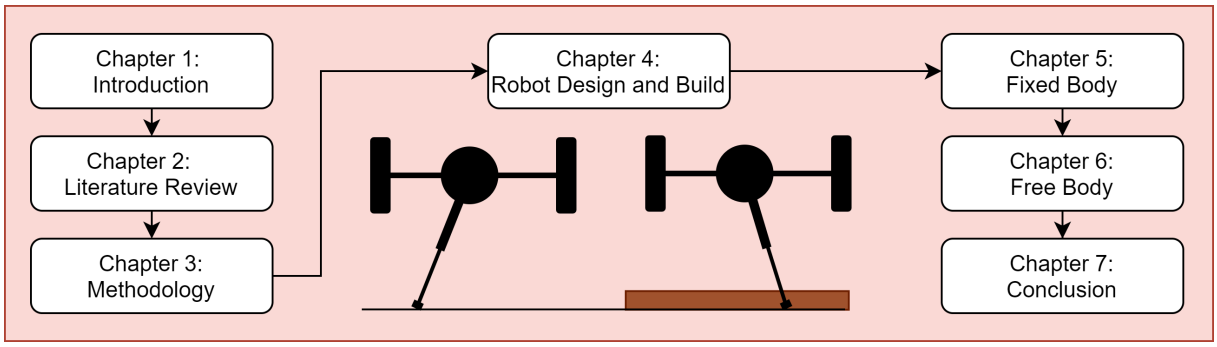

Figure 1.2: Structure of report visualized.

The work done in this thesis has been written up in an article and is currently under review for the ICRA 2022 conference. The article focuses on the work done in Chapter 6 and represents the work done to describe different terrain types as a novel method.

Chapter 2

Literature Review

This chapter will cover all the background literature that was studied before attempting the work that was done in this thesis. Legged locomotion seems like an easily executable movement when looking at animals running and performing agile maneuvers. Only when the intricacies of general locomotion are investigated does the complexity of the different phases of legged locomotion become evident. Sudden movements, changes in terrain dynamics, and overcoming obstacles are only a few of the problems that arise. This becomes even more apparent when looking at the past research and how far the research has come with legged locomotion in robotics. The amount of small adjustments and dynamic changes that an animal makes become impossible to model using traditional methods. Mechanical components also limit how closely animal movements can be mimicked since actuators used in robotics cannot accurately replicate muscle movements. To find and design actuators that are close to muscle and ligament movement in animals is another challenge which is out of the scope of this thesis.

2.1. Legged Robots

A legged robot can be defined as any robot that has a body and a leg or legs separated by an actuator [13]. Legged robots have been considered superior to other locomotive robots when overcoming obstacles [14]. The advantages of overcoming obstacles is hampered by the fact that legged robots struggle with speed, stability, and acceleration when compared to wheeled robots [14].

As computational power increased and mathematical models of legged robots became more accurate; the ability to execute complex movements improved. Initially studies were done only on the ability of legged robots to walk [15] [16]. With walking there is always a supporting leg on the ground that aids with the overall stability of the system. The disadvantages of walking is the lack of speed and agility that more closely mimics animal locomotion.

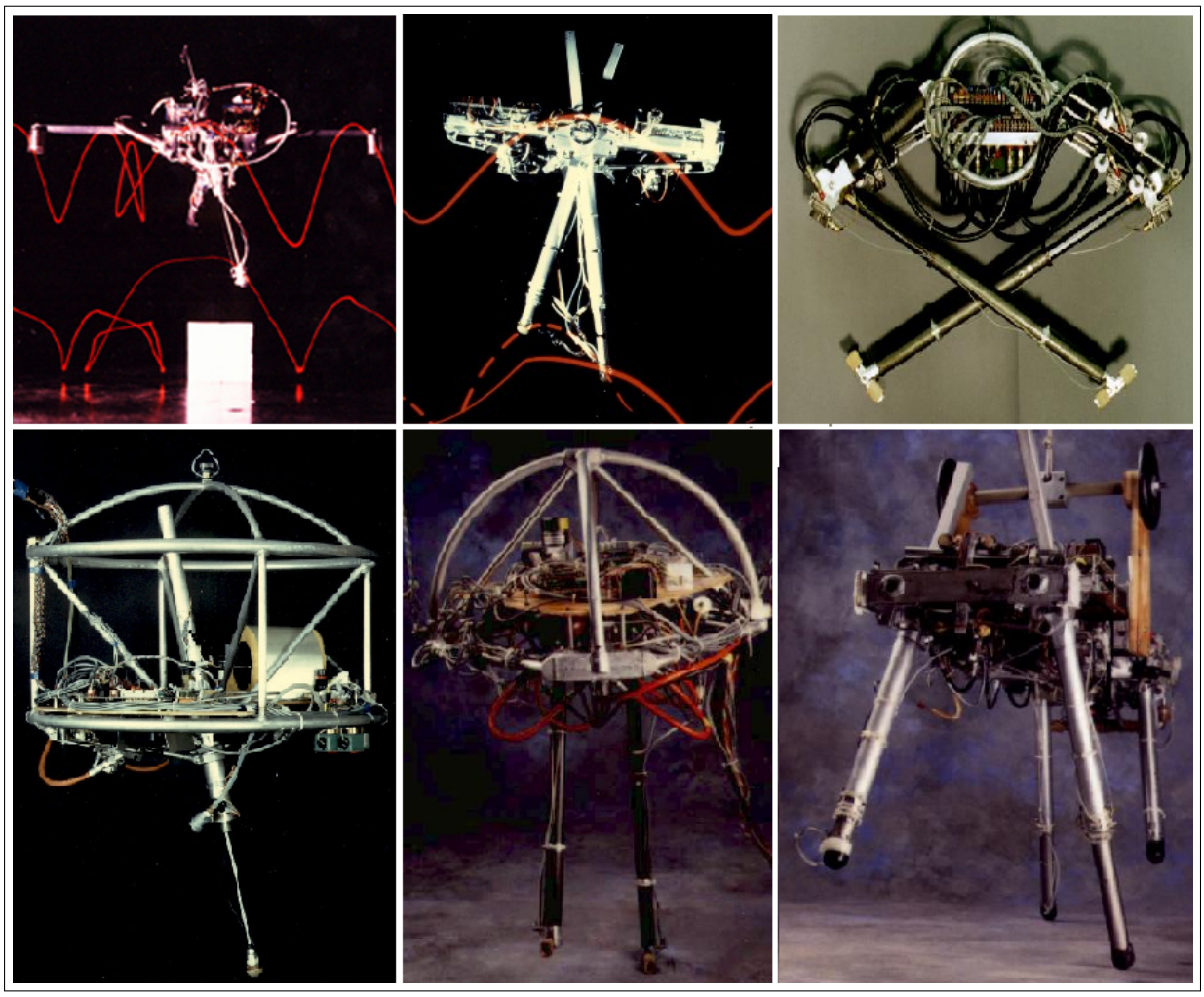

Figure 2.1: The evolution of fast moving legged robots developed by MIT [4] [17] [18] [19] [20].

One of the first studies done on running legged robots to increase speed used a single leg hopper constrained to the sagittal plane [4], and follow up studies were done by Raibert on bipedal and quadrupedal robots in 3D movement and sagittal plane movement [17] [18] [19] [20].

Dealing with obstacles efficiently is another crucial ability for legged robots. To gain any advantages over its fast locomotive rival, wheeled robots, legged robots had to be able to climb stairs and overcome obstacles found in their way. Robots such as *Guara* used contact sensors attached to the robot's feet to sense a step and initiate a movement sequence to overcome the step [21]. The strategy of total obstacle avoidance has also been used by implementing stereo cameras on a legged robot and detecting walls and turning effectively [22]. This is similar to obstacle avoidance techniques used on wheeled robots. Decision making algorithms in combination with sensors is another technique that has been investigated [23]. This strategy uses a number of stable walking patterns and a decision making algorithm gets feedback from sensors. The algorithm then chooses the

best walking pattern that will avoid the upcoming obstacle. Different sensor types are used on different robots, the ultrasonic sensors on the ARSR humanoid robot allowed the robot to implement obstacle avoidance techniques by changing its walking direction based on a decision tree method [24].

All of the research of the past 40 years in legged robotics have resulted in highly advanced robotics platforms that have been developed in publicly funded research laboratories, as well as in private research and development laboratories. ETH Zurich's ANYmal [25] is a quadruped that is able to execute several gaits such as the walk or trot gait. It is lightweight and made to be experimented on in real world scenarios. Boston Dynamics have the Atlas and Spot legged robots [26], a biped and quadruped respectively. They are being developed for commercial use and therefore have very limited academic publications about their workings. Another successful quadruped specifically made for fast and agile movements is the MIT Mini Cheetah [27]. The Mini Cheetah is light weight and can reach speeds of up to 2.45 meters per second. This indicates interest and advancements in high speed legged robots. Figure 2.2 show what these robots look like.

All of these advanced robots come from different institutions and are created using several approaches. How the robot is actuated is a crucial design choice that affects the movement characteristics of the robot as well as the amount of control the system has over the robot. In the next section we will look at different actuation methods.

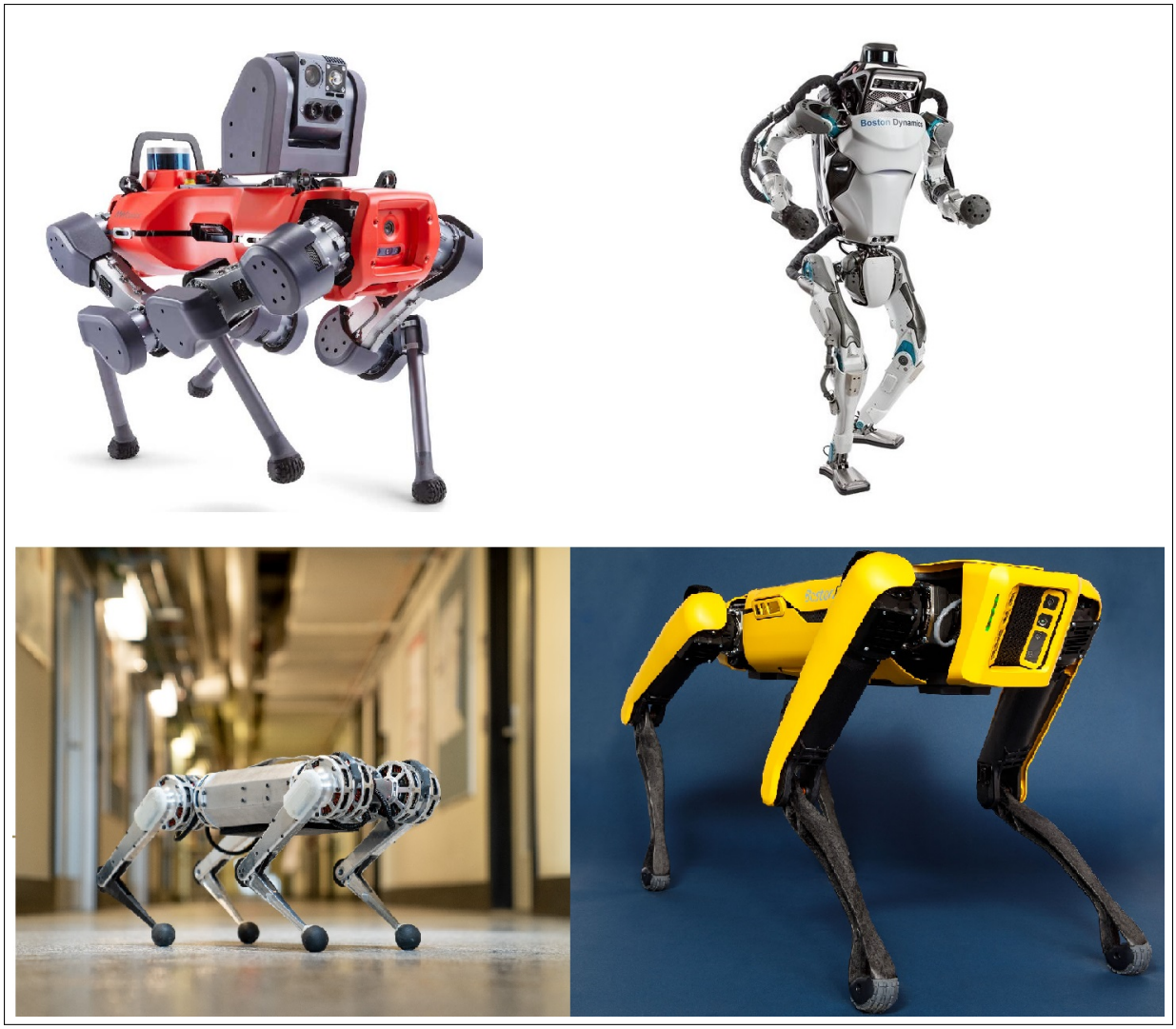

Figure 2.2: Top left is the ANYmal robot developed by ETH Zurich [25]. Bottom left is the Mini Cheetah made by MIT [27]. A biped and quadruped developed by Boston Dynamics, Atlas and Spot are shown on the top and bottom right of the figure [26].

2.2. Actuation in Robotics

When trying to mimic legged locomotion using robots the actuation mechanism is one of the most difficult components to replicate. Evolution produced a complicated yet astonishing actuation mechanism for most legged animals. Reproducing this actuation mechanism using mechanical parts has been attempted but researchers have struggled to reach the same level of performance as the ligaments and muscle mechanics in animals [28] [29] [30].

In legged robotics it is not just the mechanics of the actuation, but also the source of power in the actuation that can differ in the approach. Several different actuation approaches will be discussed in this section.

2.2.1. Direct Drive Actuation

Direct drive actuation consists of a motor attached directly to the driven component or link. This means there are no gears, chains or belts and in the case of a robot leg, the shaft of motor is connected directly to the leg of the robot. Direct drive actuators are transparent, this means that without gear backlash, and with high mechanical stiffness, any external impact on the actuator is immediately observed by the motor. Not having any gears also allows for increased mechanical robustness. Gears require maintenance and add weight and size to an actuator. All of this increases overall efficiency when compared to geared motors [31]. The repetitive impact forces experienced during legged locomotion make direct drive actuation ideal since it can absorb these impacts, whereas higher gear ratios will put a lot of strain on the gears to absorb the impacts.

Electric motors perform at their best in high speed, low torque situations and this is a major downfall of direct drive actuators. Due to the nature of legged robots and the relatively low speeds that the legs need to move at from standstill, direct drive actuators suffer from joule heating which causes motors to lack peak power at low speeds [31]. Another issue is the size to power ratio of direct drive actuators, once you have a powerful enough motor, it is often too large to fit onto the body of an appropriate robot [32].

Even with this disadvantage, direct drive actuators are still being used in robots such as Penn Jerboa [34] and Minitaur [33], seen in Figure 2.3. A company called Genesis Robotics have built a new direct drive motor specifically for robot actuation called LiveDrive [32]. It overcomes a lot of the issues with direct drive motors by changing the geometry of the motors, making them smaller, while still allowing for efficient heat dissipation.

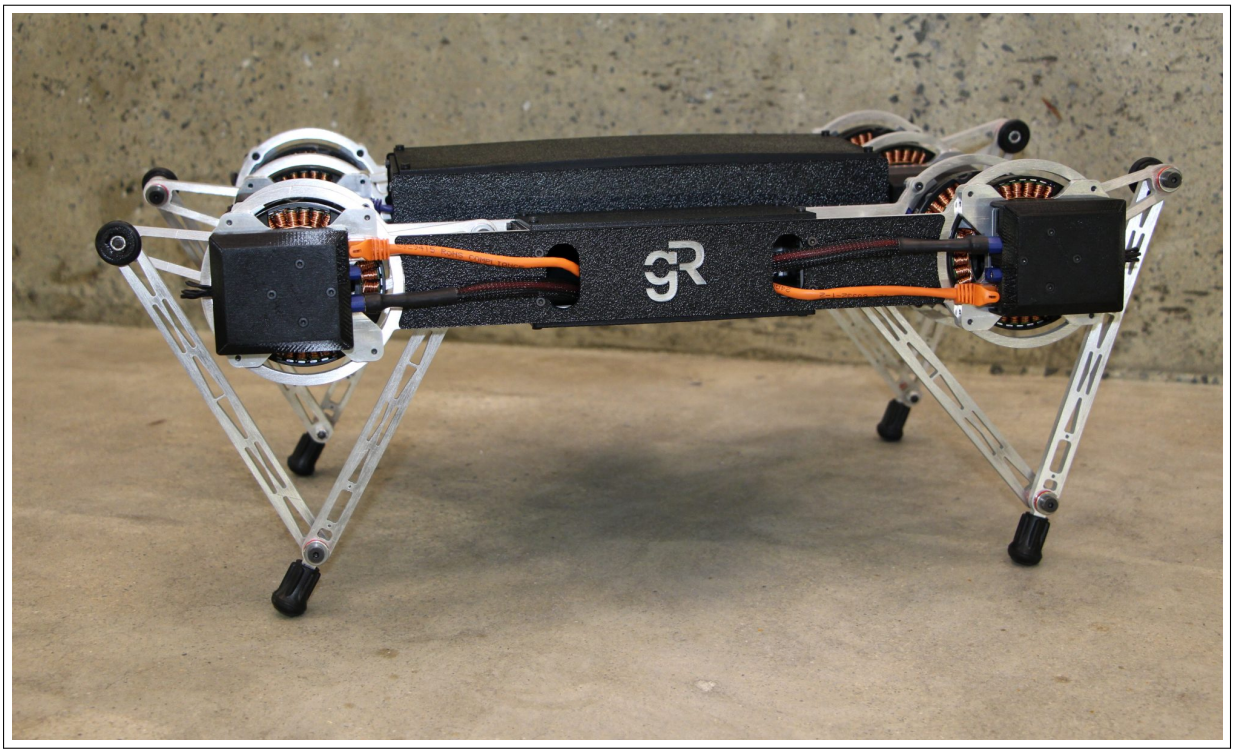

Figure 2.3: The Minitaur robot [33] with a clear view of its direct drive motors attached to the legs.

2.2.2. Geared Motor Actuation

To allow electric motors to operate in their ideal high speed and low torque conditions, high gear ratios can be used. This increases the torque density of the actuator, allowing it to execute high forces at lower speeds. Precise position control is possible with geared motors, since the actuator usually has the necessary power to move to its intended position.

With the high gear ratio comes some disadvantages. The first is geared motors have very little backdrivability caused by the high gear ratio that limits movement coming from external torque applied to the load. Backdrivability is the ability of a system of gears to operate in a reverse direction. The lack of backdrivability causes gears to wear out due to high impact external forces such as a leg hitting the ground. This reduces the robustness of these actuators [35] [36].

Acceleration capabilities of geared motors are another limiting factor. Legged robots need high acceleration in their legs to execute movements like running or balancing. Due to the high gear ratios used in the geared motor actuation, acceleration is traded off for high torque density. Backlash is another problem that is caused by the inclusion of gears, this limits the control performance of geared motor actuators. Backlash models have been developed to overcome this problem [37].

Even though direct drive actuators struggle with the high forces of impacts on the leg, and geared motors struggle with the acceleration needed for adequate leg movement; a compromise in the form of quasi-direct drive actuation can be used.

Qausi-Direct Drive Actuation

Geared motors of ratios less than 10 are referred to as quasi-direct drives (QDD). These low ratios usually require only one planetary gear. Having additional gears will result in extra backlash so one planetary gear minimises the backlash while still having the advantages of higher torques caused by the gear ratio. Backdrivability is also maintained and studies have shown that an applied external force can be sensed by analyzing the motor current [38].

QDD actuators have been used in robots such as the Stanford Doggo [38] and an unnamed exoskeleton robot in [39]. This shows that geared actuators, when used correctly, can be beneficial to electric high performing motor actuators for legged robotics.

2.2.3. Elastic Actuation

Actuators experience high impact events when they are used in legged robotics. This is due to the abrupt nature of a leg touching down on the ground. When looking at animals when they walk or run, one of the notable characteristics is the use of elasticity to soften the impact of the ground contacts. Elasticity can be incorporated in legged robots as well. With elastic actuators a spring can be placed between the electric motor and the load, this is known as a series elastic actuator (SEA). A spring can also be placed in parallel with the load and the motor, known as a parallel elastic actuator (PEA). Figure 2.4 illustrates the difference in spring placement between SEA and PEA.

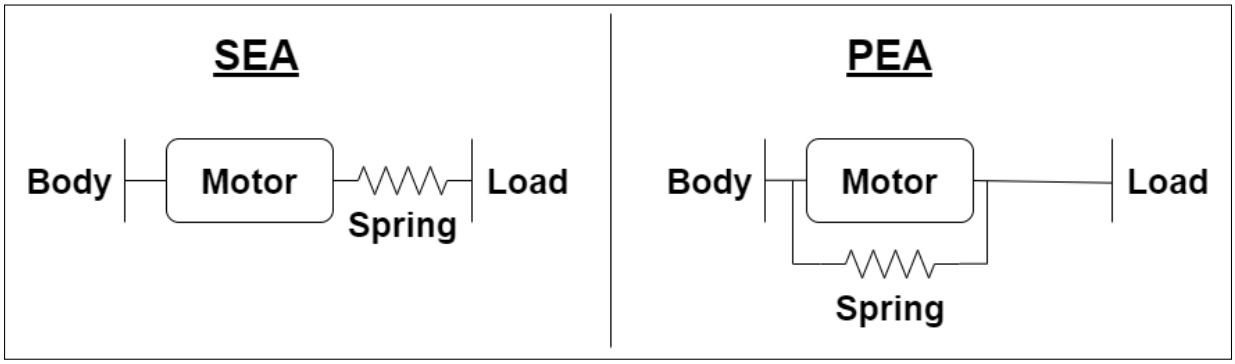

Figure 2.4: This figure shows that with SEA the spring is placed between the motor and the load, and with PEA the spring is placed between the body and the load.

Elastic actuators reduce the high impact on geared motors, improving overall robustness of the actuator. They also act as an energy storage device that improves the efficiency of the robot. The elasticity in the actuator contradicts the "stiffer is better" rule for legged robotics, and less precise position control can be executed when there is elasticity in the actuator. But the elasticity introduced by adding a spring can be used as a form of force sensing in a control system and can replace the need for position control [40]. The force applied to the leg can be sensed by observing the displacement of the spring and knowing what the spring constant is. This is shown in equation 2.1.

$$F = K_{spring} d_{spring} \tag{2.1}$$

Several legged robot platforms such as the MIT Mini Cheetah, the Boston Dynamics BigDog and the ETH Zurich AnyMal use elastic actuators [25] [27] [41]. The payoff of less precise movement control seems to not deter decision makers from using elastic actuators. When more precise actuator movements are needed, elastic actuators will not be a good choice. Another disadvantage of elastic actuators is the inability to change spring rate dynamically. There will always be a design compromise between a low spring rate that eliminates linear friction and impedance, and a high spring rate that increases force bandwidth [29]. And a spring rate that works for steady-state movements might not work well for acceleration or deceleration.

2.2.4. Pneumatic Actuation

Pneumatic actuation uses compressed gas as its source of energy. It usually consists of a chamber, piston and solenoids that control the compressed gas flow. Figure 2.5 shows that when a valve opens up, the compressed gas flows into chamber B and since the piston acts as one of the sealing walls of the chamber it moves inward as gas expands into the cylinder [42]. Once the compressed air flows into chamber A the piston moves outward. This movement is used as actuation in many mechanical applications including robotics [43] [44]. Other forms of pneumatic actuation are available, like the McKibben artificial muscles specifically developed for humanoid robots [30]. One advantage of pneumatic actuators are their relative light weight, since they use gas as a source of power. They are also mechanically simple, making them robust and capable of handling high impact forces. The gas used is also compressible, which aids in absorbing the impulsive forces experienced during legged locomotion.

One of the biggest disadvantages of pneumatic actuators is their control bandwidth. Due to the dynamics of gas and the delay in the switching of solenoid valves ($20ms$) [45],

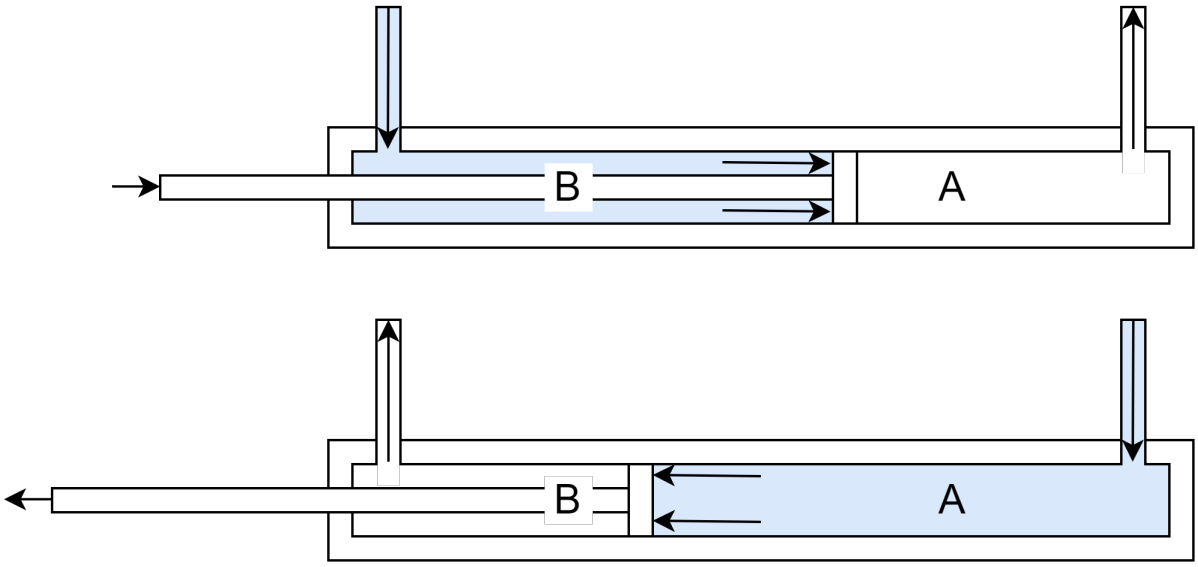

Figure 2.5: Pneumatic actuation explained visually, 2 chambers (A and B) are filled with air or drained of air to enable the shaft to move.

position control or force control is nearly impossible. Another disadvantage is the energy source being compressed gas, requiring most robots to have an off-board supply which limits the mobility of pneumatically actuated robots.

Since the robot we are designing is a mono-pod, it will only execute movement in the sagittal plane and a support rig will be necessary. The design and development of the support rig is the next important step.

2.3. Robotic Support Rigs

Most robotic platforms need some form of support during initial testing in the laboratory. The robots that are built to stand alone in the real world are attached to some sort of support rig initially for safety. The design of the support rig is important since it needs to allow the necessary leg movements for efficient testing. In some cases, like the planar hoppers developed by MIT [4] [18], the support rig needs to limit the dimensional movement of the robots. We will discuss some previously designed support rigs and what kind of robotic movement they support.

For robots such as Baleka [46] and the robot used in [9] the main goal of the support rig was to limit the movement of the robot to the sagittal plane. The design choice to achieve this was using a boom support rig. The boom was attached on the one side to a fixed point that can swivel around, and the robot was attached to the other side. A boom design helps conserve space by turning the robot in a circular motion as it moved forward

or backward. Although the circular motion does not represent a true sagittal plane, if the circle that the robot moves in is big enough it is seen as an approximation of the sagittal plane. Similar support rigs were used on the initial sagittal plane robots developed by MIT [18].

Another method to limit movement to the sagittal plane is building a support trolley around the legged robot. The Kenji Hashimoto Lab at the Meiji University has a few trolley support rigs to support several legged and wheeled robots [47]. A trolley with fixed directional wheels move with the robot in the sagittal plane and supports the robot to not fall over in the other directions of movement.

Robots that are meant to move in 3-dimensional space do not need support rigs to limit movement. However, a mechanical robot falling during testing can cause damage to expensive equipment and to the environment it is moving in. To test the initial stability of free moving robots, they are usually tethered to a moving platform. Examples are on the Atlas robot developed by Boston Dynamics [26]. A large structure covering the test area of the robot was built with a rail at the top. A tether was attached to the rail on the one side and the robot on the other side. If the robot fails to balance itself and falls, the tether will prevent it from hitting the ground at a high velocity and causing damage. Once a certain level of trust in the stability of the robot is reached, the tether is no longer necessary. The different support rig designs can be seen in Figure 2.6.

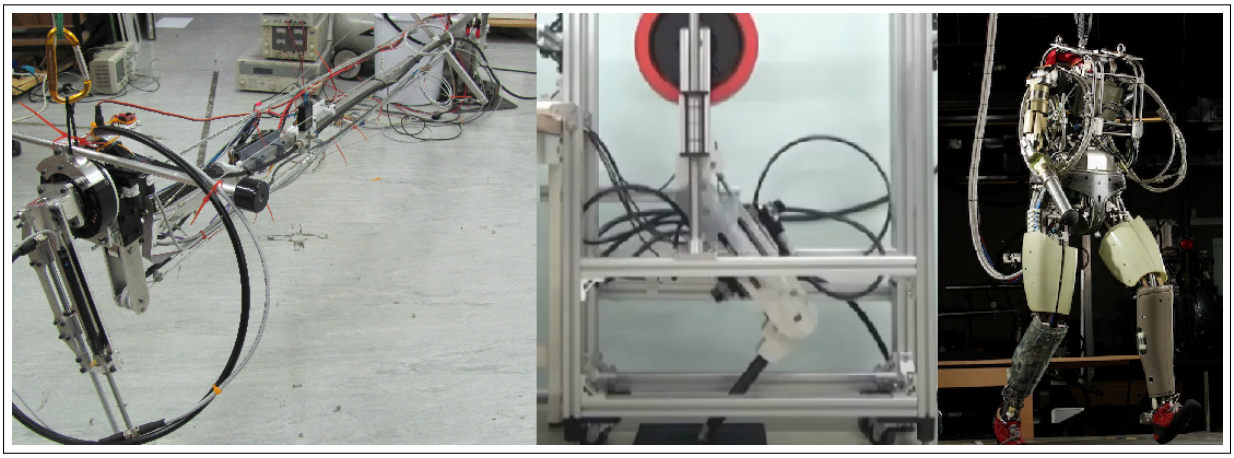

Figure 2.6: The different support rigs discussed in this section, on the left is the boom support rig used in [9], in the middle is the support rig used by the Kenji Hashimoto Lab [47] and on the right is an older version of the Atlas robot [26] that is tethered.

We will look at different ways to overcome the difficulties of real world legged locomotion next.

2.4. Dealing with Terrain

To design and develop legged robots that perform well in the real world, we need to create accurate models. These models have to represent the internal and external forces that will be applied to the robot. A major challenge in legged robot simulation is the accurate modelling of the ground reaction force (GRF) applied to the leg when it touches the ground. This is because it is a very impulsive and discontinuous force that instantly changes the dynamics of the model and introduces complex interactions into the model such as friction and slipping.

In Kanakis et al. [48] the rigid terrain was included by prescribing a contact order and changing the dynamics to include a spring force that represented the terrain. A contact order means the position in time and space where the foot will make contact with the ground. This forces the model into to a fixed contact order and can lead to infeasible or sub-optimal solutions. The spring force is also a soft contact where energy is lost during contact. In Posa et al. [8] the contact forces are defined as variables in a trajectory optimization problem. These variables were subject to complementarity constraints. This allowed the solver to apply a contact force whenever contact was needed in the trajectory. In Patel et al. [49] an epsilon relaxation method is used to make the complementarity constraints in Posa et al. [8] more tractable. The epsilon relaxation method has only been implemented in simulation and has not been tested on a robot platform.

Another challenge that compounds onto the modelling of the GRF is when the GRF is a dynamically changing force due to compliance in the terrain. This will drastically change how the robot interacts with the ground and the robot has to compensate for it. We will now look at ways compliant terrain has been dealt with in the past. First we will discuss the modelling and simulation techniques used, then we will look at practical implementation on robotic platforms.

2.4.1. Compliant Terrain in Simulation

Compliant terrain is terrain that is non-rigid. Research such as Hubicki et al. [1], and Drnach and Zhao [50] have attempted ways to compensate for compliant terrain. In Hubicki et al. [1] an added-mass description of collective grain motion was used to describe compliant terrain successfully, but only on a robot constrained to the vertical plane. In Drnach and Zhao [50] uncertainties in terrain were modelled using stochastic complementarity constraints. This has not been tested on a robotic platform, and is purely a simulation based study therefore none of these studies showed a clear path to successful execution of

their methods on a physical platform.

Once the simulation describes the GRF accurately, dealing with dynamic changes in the terrain in real life is the next challenge.

2.4.2. Compliant Terrain in Real Life

Several different approaches to dealing with compliance in terrain have been presented and tested. General necessities in the literature are a form of sensing whether the terrain has changed, and accurately adjusting to the changing terrain in a timely manner.

One of the methods to overcoming rough terrain is proposed in Aso et al. [51]. This research proposes a passive approach without the need for unnecessary control to react to the terrain. This is done by creating a highly stable robotic platform called Shinayaka that has flexibility in the legs when moving forward, but is stiff when moving vertically to be able to lift the body up. This platform is good for crawling maneuvers and is a great way to remove computational complexity. However rapid movements are not possible with this robot.

Reinforcement learning is used in Lee et al. [52] to train a controller that uses a stream of on-board signals as its input. The model is trained in simulation on several different types of terrain. This method has been used in the real world on the ETH Zurich AnyMAL robot [25]. The model has been able to perform in environments that were not encountered during training showing its robustness.

Kalakrishnan et al. [53] decompose the control problem into multiple subsystems. They use optimal foothold choice by using terrain templates, a body trajectory optimizer and a floating base inverse dynamics optimizer. This approach is tested on the LittleDog quadruped robot and multiple unknown terrains are traversed succesfully by the robot. The different robots can be seen in Figure 2.7.

As can be seen by the variety of approaches to overcome rough terrain, there is no standout way to walk and run on unknown terrain for legged robots. One approach to rigid terrain trajectory solutions that is used in Posa et al. [8] is trajectory optimization.

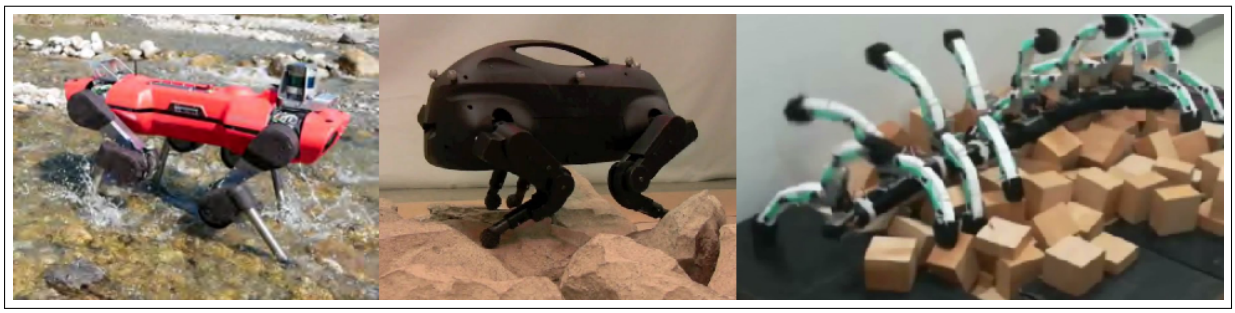

Figure 2.7: The robots that tested the different approaches to overcoming rough terrain. ETH Zurich AnyMAL on the left, the littleDog in the middle and the Shinayaka robot on the right.

2.5. Trajectory Optimization

Trajectory Optimization is a numerical technique that is commonly used to solve open-loop solutions to optimal control problems. There are two methods used in trajectory optimization, direct and indirect methods [54]. We will focus only on direct methods since our research revolves around it.

A continuous optimal control problem is discretized and constraints are used to describe the physical rules and limitations of the environment the problem has to be solved in. Variable bounds are defined to limit the search space of the optimizer. The problem is optimized according to a predefined cost function. The cost function is usually either energy based or time based [55] [56]. The optimizer generates a feasible solution by starting with a seed of variables and using gradient descent to find a local minimum.

For complex systems like legged robots it is nearly impossible to describe the physical parameters of the robot through calculations. In the past computers were too slow to use trajectory optimization to describe a robot doing a specified task. With increased computational power the use of trajectory optimization in legged robotics has started to become a valuable addition. It is especially valuable when describing non-linear rapid movements such as acceleration and deceleration since these movements cannot be predefined by calculations.

In Posa et al. [8] trajectory optimization is used to solve the complex problem of contacts in robotic simulation. Complementarity constraints were added to the optimal control problem to model the interaction between the foot of the robot and the ground. Complementarity constraints enforce two variables to be complementary to each other for example the

following conditions should hold for A and B.

$$\begin{aligned} A &\geq 0, \\ B &\geq 0, \\ AB &= 0 \end{aligned} \tag{2.2}$$

This enables the solver to choose the optimal nodes where contact should happen without the use of prescribed contact order.

In Carius et al. [57], the ETH Zurich Robot AnyMAL [25], that has been discussed earlier, is modelled in a trajectory optimization problem. In the research they are able to generate movements that do not need a prescribed contact order and can allow the foot to slip when necessary. With traditional quadruped modelling methods this is not possible since the robot will fail on unknown terrain. A very basic PD controller is used to execute the trajectories on the robot platform successfully. Their results show that the use of trajectory optimization unearthed robotic movements that was not previously considered.

In Steenkamp et al. [58], trajectory optimization is used to analyse the acceleration of greyhounds. A 2D model is set up with two legs and a body, and the model is required to start from rest and accelerate to a set velocity. This shows how well trajectory optimization solves non-linear movements. This study does not prescribe periodicity or foot contact order, allowing the optimizer to decide what the optimal order and periodicity is. The results show similar movements to that of greyhounds and will inspire the design of feedback controllers for these kind of movements.

Another use of trajectory optimization is to generate trajectories for legged robots that can be analyzed to inspire controller design and controller gain tuning [9]. In this work a long time horizon task is optimized. From the movement and torque behaviour a Raibert controller emerges and is implemented on the platform. It is shown that the implementation of the Raibert controller on the robot produces similar trajectories to the ones generated in simulation by the optimizer.

2.6. Summary

This chapter covered the background literature that was needed for this thesis. Firstly general legged robots was discussed, as well as different approaches taken to develop successful legged robot designs. Thereafter important subsections of the design and development of legged robots was covered. The actuation methods used to power robot legs was covered, focusing on the advantages, disadvantages and general functionality of

these actuators. Following this, the design of different support rigs to assist in laboratory experiments of the robots was looked at. The different approaches to overcoming compliant terrain in legged robotics was also covered. Lastly the use of trajectory optimization in legged robotics was discussed.

Chapter 3

Methodology

The work done in this thesis addressed two main problems. Firstly, can compliant terrain be modelled accurately in a trajectory optimization problem. Secondly, can trajectory optimization be used to inspire controller design for a hybrid pneumatic-electric mono-pod robot running on compliant terrain. A method was developed to present compliant terrain in trajectory optimization in a way that allowed feasible trajectories to be generated. Once these trajectories were generated they were analysed to inspire controller design.

A drawing of the hybrid pneumatic-electric mono-pod robot is shown in Figure 3.1. It is a one legged robot with a torque actuator at the hip and a linear actuator along the leg. The body can rotate freely, meaning balancing the mono-pod to stay upright would be necessary. Inertial weights were added to the body to stabilize it.

To generate feasible trajectories to implement on the robot, a mathematical model describing the robot was required. Specific trajectory optimization manipulations were used to describe the discontinuous ground reaction forces (GRF) and the pneumatic actuator forces. The methods used to create this model is discussed in this chapter.

3.1. Trajectory Optimization Setup

Initial ideas of how the mono-pod robot will work, what actuation methods it will use and what movement restrictions need to be applied, made it possible to setup a mathematical model that will be used in trajectory optimization. The mono-pod was constrained to the sagittal plane and consisted of a leg that can extend and rotate relative to a freely rotating body. The leg was modelled as a pneumatic cylinder that extends and contracts in line with the angle of the leg.

Previous research revealed that the pneumatic cylinder can only extend or contract at a constant force [59]. Therefore the pneumatic cylinder was modelled as a bang-bang

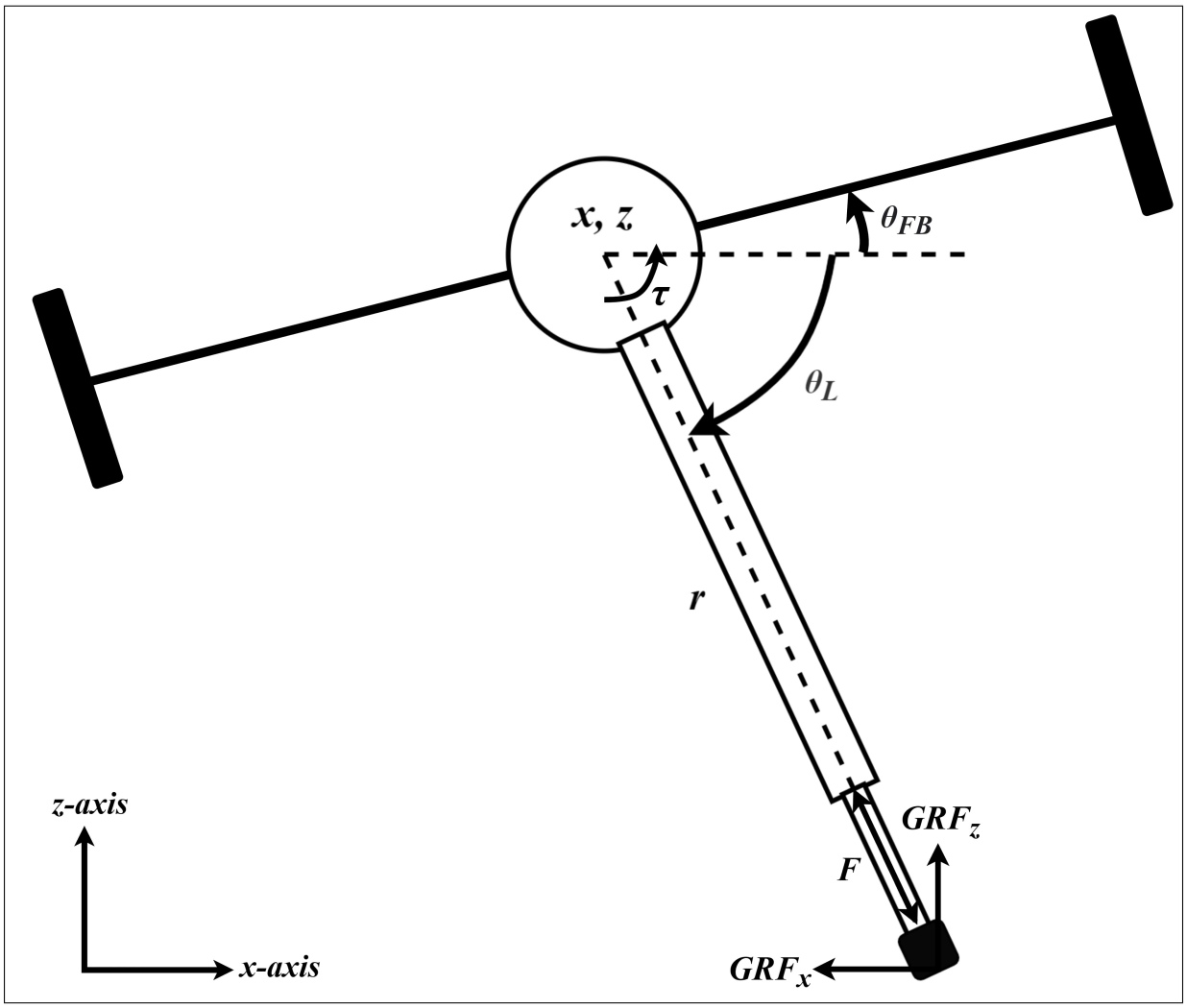

Figure 3.1: Model of mono-pod robot showing the generalized coordinates and the basis vectors as well as the applied forces and torques. The black rectangles in the image are the inertial weights added to stabilize the body.

actuator with a damper [9]. The bang-bang actuator could fully extend and fully contract allowing for a push-pull motion.

The system can be described by five generalized coordinates: x, z - the position in the planar field of the body, θ_L, θ_{FB} - the angle of the leg and the body clockwise from the positive x-axis, and r - the length of the leg measured from the centre of the body to the foot of the leg. The complete model is illustrated in Figure. 3.1.

Trajectories were solved using the open-source Python package, PYOMO [60]. In PYOMO a selection of solvers are available. The IPOPT solver was used to solve the optimal control problem. To ensure realistic trajectories were generated that could be used to execute realistic robot movement, certain constraints had to be placed on the model. The most important constraint to ensure feasible real world movements were the equations of motion.

3.1.1. Equations of Motion

Initially a fixed body robot was designed and built, this meant that the model of the robot had no body pitch movement. After successful experiments were done on the fixed body robot, a free body robot was designed and built for the final experiments. With the free body model, the inertial forces acting on the leg counteracted that of the free body. The mass of the free body and the leg, m_{fb}, m_l, were defined and used in the equations of motion (EOMs). At the body there was a torque, τ_{hip}, acting on the leg that is applied to rotate the leg to a desired angle. The force, F, extended and contracted the leg and included any hard stop forces that limited the linear movement of the prismatic actuator. The force , F, was also limited by a damping force, $C\ddot{r}$. The damping force is the product of the speed of the leg, $\ddot{r}$, and a damping coefficient, C, that represents the damping of the pneumatic cylinder. There were also ground reaction forces (GRF), λ, acting on the foot of the robot. The size and direction of the GRFs depended on the angle and velocity of the foot. The forces and torques are shown in Figure 3.1.

Using the Euler-Lagrange method, the manipulator equation was used to calculate the EOMs of the model [61]:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}\boldsymbol{\tau} + \mathbf{A}\boldsymbol{\lambda} \tag{3.1}$$

Where $\mathbf{M}$ is the positive definite inertia matrix, $\mathbf{C}$ is the coriolis matrix, $\mathbf{G}$ is the gravitational potential matrix and $\mathbf{B}$ and $\mathbf{A}$ mapped any of the external forces and torques applied to the system to the generalized coordinates. The generalised coordinates, $\mathbf{q}$, for the fixed body model (Chapter 5) is the following:

$$\mathbf{q} = [x, z, \theta_{leg}]^{\mathbf{T}} \tag{3.2}$$

The generalized coordinates for the free body model (Chapter 6) is shown below:

$$\mathbf{q} = [x, z, \theta_{leg}, \theta_{body}]^{\mathbf{T}} \tag{3.3}$$

The applied forces are defined in $\boldsymbol{\tau}$. Our robot will have a prismatic actuator attached to a hip actuator as follows:

$$\boldsymbol{\tau} = [\tau_{hip}, F]^{\mathbf{T}} \tag{3.4}$$

Where $F = F_{extend}^{applied} - F_{extend}^{reaction} - F_{contract}^{applied} + F_{contract}^{reaction} - C\ddot{r}$. The variables, $F_{extend}^{applied}$ and $F_{contract}^{applied}$ represent the prismatic actuation force of the pneumatic cylinder. The force is separated into extension and contraction variables to enable complementarity constraints to be applied to it. The variables, $F_{extend}^{reaction}$ and $F_{contract}^{reaction}$ represent the prismatic reaction force of the cylinder. Once the cylinder reaches its maximum length a contracting reaction force, $F_{contract}^{reaction}$, has to be applied to it to create a hard stop. The same principles are applied to the extension force, $F_{extend}^{reaction}$.

Since we only worked in the sagittal plane, the external forces, $\boldsymbol{\lambda}$, were the ground reaction forces and were represented as follows:

$$\boldsymbol{\lambda} = [\lambda_x^+ - \lambda_x^-, \lambda_z] \tag{3.5}$$

λ_x^+ and λ_x^- are the ground reaction forces applied in the x-axis. The force is defined as two variables, a positive and negative variable, to enable complementarity constraints to be applied to the variables.

Once the EOMs are described as an equation, it could be implemented as a constraint in trajectory optimization. But before this could be done a model for a specific trajectory optimization solver should be created and defined.

3.1.2. Creating a trajectory optimization model

As shown in the literature review, Chapter 2, trajectory optimization is a numerical technique that optimizes a trajectory according to a predefined cost function as well as bounds and constraints. Direct methods of optimization were used, and therefore models had to be created and defined for the solver to use.

The number of nodes in a model is defined as N, these nodes represent a collection of decision variables and are linked using splines. N is usually chosen to be a value between 40 and 300 to keep the solve time reasonable while still allowing for detailed solutions.

A time constant was defined as, $h_m = T/N$, where T represented an estimated time that the trajectory would take to run. A time variable, h_n, for each node (n) was also defined and constrained to, $0.2 < h_n < 1.5, n \in (1, N)$. Equation 3.6 shows how the time interval between nodes, $t(n)$, was calculated.

$$t(n) = h_m h_n, n \in (1, N) \tag{3.6}$$

A variable time integrator was needed in the model to enable the optimizer to enforce a contact to land on a node point [49]. If time variation was not possible, and a contact occurs between node points, the solver would not be able to find a feasible solution.

The amount of nodes were defined as N for the optimal control problem. The optimizer only solves discrete approximations of continuous trajectories since software lives in a discrete domain. The amount of nodes determines the time interval between each point in the trajectory. If a small number of nodes are chosen, the interval between the nodes would be too large leading to inaccuracies in the trajectory. The complexity of the problem increases non-linearly with the number of nodes. If too many nodes are chosen the accuracy would be unnecessarily precise and solving the trajectory would take too long due to the limits in computational power.

There is no definite optimal amount of nodes for trajectories of specific times. Through studying past trajectory optimization problems, and experimenting with smaller trajectories, the range of nodes to use for different time ranges and trajectory types become clearer [7].

Collocation

An additional technique that was used to increase the accuracy of the trajectories, without drastically increasing the complexity of the problem, was three-point collocation [55]. Once the optimal control problem was discritized into N nodes, three point collocation used Runge-Kutte integration schemes to approximate two additional trajectory points between nodes [62]. This increased the accuracy of the trajectories while decreasing computational time that comes with an increase in nodes.

A Runge-Kutta basis matrix, $\mathbf{a}$, was used to approximate the collocation points to the trajectory [49]. The values in the matrix are as follows:

$$\mathbf{a} = \begin{bmatrix} 0.19681547722366 & 0.39442431473909 & 0.37640306270047 \\ -0.0655354258502 & 0.29207341166523 & 0.51248582618842 \\ 0.02377097434822 & -0.041548752126 & 0.11111111111111 \end{bmatrix} \tag{3.7}$$

This matrix was implemented in the following equations to solve the dynamics at the nodes and the approximate collocation points:

$$
\begin{aligned}
q(i,j) &= q_0(i) + h(i) * \sum_{k=1}^{3} \mathbf{a}(k,j)\dot{q}(i,k) \\
\dot{q}(i,j) &= \dot{q}_0(i) + h(i) * \sum_{k=1}^{3} \mathbf{a}(k,j)\ddot{q}(i,k) \\
q_0(i) &= q(i-1,3) \\
\dot{q}_0(i) &= \dot{q}(i-1,3)
\end{aligned}
\tag{3.8}
$$

In Equation 3.8 q_0 represents the node points of the trajectory and q represents the more accurate trajectory that includes collocation node points. The first two parts of Equation 3.8 implements the $\mathbf{a}$ matrix to ensure the collocation points accurately represent the trajectory. The last two equations ensure that a continuous trajectory is upheld by equating the 3rd collocation point of a node to the next node in the trajectory.

Cost Function

For the optimiser to generate optimal solutions, a cost function has to be defined. For transient motions there is no intuitive cost function. An energy based cost function is the most common cost function for steady-state motions, however this does not work well for explosive movements needed for rapid acceleration and deceleration. For all the trajectory optimization models in this thesis a time based cost function was used [9]. Equation 3.9 shows how the cost function was defined.

$$
J = \sum_{i=1}^{N} t(i) \tag{3.9}
$$

3.1.3. Complementarity Constraints

Any form of rigid contact event is difficult to simulate in a trajectory optimization model. If an object makes contact with another object, a discontinuous force would appear between them. If this force did not appear the two objects would go straight through each other. The discontinuity of the force is what makes it challenging to simulate. One method is to force the two objects to make contact at certain nodes and make use of hybrid dynamics to actively change the EOMs of the system to handle the contact event, this is called the prescribed contact methods.

For a more robust solution, where the optimizer could decide when the objects touch each other, complementarity constraints were used. Complementarity constraints also allow

for different contact types such as sticking or slipping. The principle of complementarity constraints use two variables α and β respectively, and they adhere to the following constraints [50]:

$$\begin{aligned} \alpha &\geq 0, \\ \beta &\geq 0, \\ \alpha\beta &\leq 0 \end{aligned} \tag{3.10}$$

The first two constraints of equation 3.10 ensured α and β are positive variables. The last constraint ensured that when α was larger than zero, β had to be equal to zero, and once β was larger than zero α was equal to zero [50]. The complementarity constraints were used to model several hard contact events in our trajectories. This is explained below.

Rigid Terrain

To generate realistic and usable trajectories our robot had to make contact with the ground. Initially the terrain on the ground was made rigid, since the initial experiments were going to be on rigid terrain. To generate robust trajectories where contact with the ground is selected by the optimizer, complementarity constraints were applied to model these contact events.

The foot height of our robot, z_{foot}, and the GRF in the z-direction, λ_z, replaced α and β from Equation 3.10. This meant that once the foot is on the ground with its height equal to zero, the appropriate GRF could be applied to the robot. Once the foot was in the air the GRF had to be zero to adhere to the constraints. The constraint is shown below.

$$\begin{aligned} z_{foot} &\geq 0, \\ \lambda_z &\geq 0, \\ z_{foot}\lambda_z &= 0 \end{aligned} \tag{3.11}$$

Once the ground reaction force in the z-direction was calculated, the ground reaction force in the x-direction was calculated. The ground reaction force in the x-direction is dependant on the velocity of the foot of the robot and the friction coefficient of the ground (assumed to be $\mu = 1$ for all simulations in this thesis). A friction cone was enforced using

the following constraint [63]:

$$\mu\lambda_z - \lambda_x^+ - \lambda_x^- \geq 0, \tag{3.12}$$

This ensures that the reaction force in the x-direction should always be equal to the reaction force in the z-direction times the friction coefficient.

To get a variable that represents the magnitude of the tangential velocity of the foot the following constraints were applied to the model:

$$\begin{aligned} \dot{\mathbf{x}}_{foot} + \dot{x}_{foot} &\geq 0, \\ \dot{\mathbf{x}}_{foot} - \dot{x}_{foot} &\geq 0, \\ \dot{\mathbf{x}}_{foot} &\geq 0 \end{aligned} \tag{3.13}$$

Where $\dot{\mathbf{x}}_{foot}$ is the magnitude of the velocity of the foot, the speed of the foot, and $\dot{x}_{foot}$ is the velocity of the foot. Next, to ensure the friction force lies on the friction cone when the foot slides, the following complementarity constraint was applied:

$$(\mu\lambda_z - \lambda_x^+ - \lambda_x^-)\dot{\mathbf{x}}_{foot} = 0, \tag{3.14}$$

Finally the following two complementarity constraints were enforced to ensure the ground reaction force in the x-direction will oppose the foot sliding in the positive or negative directions.

$$\begin{aligned} (\dot{\mathbf{x}}_{foot} + \dot{x}_{foot})\lambda_x^+ &= 0, \\ (\dot{\mathbf{x}}_{foot} - \dot{x}_{foot})\lambda_x^- &= 0, \end{aligned} \tag{3.15}$$

Epsilon Relaxation

The complementarity constraints would have been complex constraints to adhere to for the optimizer. To have found an optimal and feasible solution where $\alpha\beta$ was equal to 0 throughout all the nodes in the trajectory would have taken too long and might have been impossible. To assist in finding a solution that will be more solvable an ϵ-relaxation method was used [55].

A variable ϵ replaced the 0 in the third part of Equation 3.10. ϵ was initialized to be

equal to 1000, this is a sufficiently large number to start with. The trajectory was solved several times. After each time the value of ϵ was decreased by a factor of 10. This slowly tightened the tolerance of the complementarity constraint and allowed for an easier path to a feasible solution. Eventually ϵ was equal to 10^{-4} and from previous work done this was accepted as a low enough value to solve accurate trajectories [9]. The last part of Equation 3.10 now changed to the following:

$$\alpha\beta \leq \epsilon \tag{3.16}$$

Prismatic Reaction

Within the prismatic actuator there was a limit to how far the actuator could extend and contract. Once it reached these limits a hard contact event had to occur to prevent it from moving past these limits. Complementarity constraints were applied when the cylinder was either at its minimum or maximum length to ensure these hard contact events took place.

Two separate complementarity constraints were applied for the actuator extension and contraction. Two separate reaction forces were defined for these constraints as well. $F_{contract}^{reaction}$ would be applied when the actuator is at maximum contraction, and $F_{extend}^{reaction}$ would be applied when the actuator is at maximum extension.

For extension the maximum actuator length minus the current actuator length, $(r_{max} - r)$, and the reaction force, $F_{extend}^{reaction}$, represented α and β respectively from Equations 3.10 and 3.16. This resulted in the following constraints.

$$\begin{aligned} (r_{max} - r) &\geq 0, \\ F_{extend}^{reaction} &\geq 0, \\ (r_{max} - r)F_{extend}^{reaction} &\leq \epsilon \end{aligned} \tag{3.17}$$

For contraction the actuator length minus the minimum actuator length, $(r - r_{min})$, and the reaction force, $F_{contract}^{reaction}$, represented α and β respectively from Equations 3.10 and

3.16. This resulted in the following constraints.

$$\begin{aligned} (r - r_{min}) &\geq 0, \\ F_{contract}^{reaction} &\geq 0, \\ (r - r_{min}) F_{contract}^{reaction} &\leq \epsilon \end{aligned} \tag{3.18}$$

Bang-Bang Actuation

With the prismatic actuator that was used on this robot no force regulation was implemented. The actuator could either extend at a constant force ($F_{extend}^{applied} = F_{max}$) apply no force ($F_{extend}^{applied} = F_{contract}^{applied} = 0$) or contract at a constant force ($F_{contract}^{applied} = F_{max}$). Since this can also be seen as a discontinuous event, a modified version of a complementarity constraint was used to model this actuator force. The following Equations show how complementarity constraints were used to model these forces.

$$\begin{aligned} 0 \leq F_{extend}^{applied} &\leq F_{max}, \\ (F_{max} - F_{extend}^{applied}) F_{extend}^{applied} &\leq \epsilon \end{aligned} \tag{3.19}$$

The extension and contraction forces were seperated as variables, this made it possible to apply separate constraints to them. The extension force, $F_{extend}^{applied}$, was constrained to be a value between zero and F_{max}. Then F_{max} minus the extension force, $F_{extend}^{applied}$, represented α and $F_{extend}^{applied}$ represented β in Equations 3.10 and 3.16. This ensured that $F_{extend}^{applied}$ could either be equal to F_{max} or equal to zero. The following equations show how these constraints were implemented.

$$\begin{aligned} 0 \leq F_{contract}^{applied} &\leq F_{max}, \\ (F_{max} - F_{contract}^{applied}) F_{contract}^{applied} &\leq \epsilon \end{aligned} \tag{3.20}$$

The same constraints were implemented for the contraction force. It is important to note that the extension and contraction forces are both positive variables but are implemented in the opposite direction in the model.

An additional constraint was added to this modified complementarity constraint to ensure that $F_{extend}^{applied}$ and $F_{contract}^{applied}$ cannot be equal to F_{max} at the same time. The ϵ-relaxation method is also used with this constraint.

$$F_{extend}^{applied} F_{contract}^{applied} \leq \epsilon \tag{3.21}$$

3.1.4. Node Bucketing

The time between nodes in the trajectories that where generated would typically range between $3ms$ and $18ms$. The prismatic actuator on the real robot was a pneumatic cylinder controller by a solenoid. The solenoid had a switching delay, making it impossible to switch rapidly to mimic force regulation. This switching delay ($20ms$) had to be modelled for the trajectories to be accurate [45].

A novel node bucketing technique was developed to reduce the resolution of the prismatic actuator force and prevent rapid extension and contraction. The model has N nodes that discretize the continuous trajectory. For this technique K nodes were also generated that represent a collection (or bucket) of nodes N. Each bucket had N/K nodes in it that had to represent the same value. The amount of K nodes generated depended on how rapidly the specific solenoid valves were able to switch compared to how long the trajectory took to execute.

The value of K had to be a factor of N to ensure division would result in an integer. Constraints were used to implement this technique. K variables that represented the prismatic actuator force at each node k were initialized. These variables were initially constrained by the complementarity constraints in Equations 3.19 and 3.20. Thereafter the buckets of n variables were constrained to be equal to their corresponding k variable. These constraints were implemented for an extension force, $F_{bucket,ext}(k)$, and a contraction force, $F_{bucket,cont}(k)$. The constraints are shown below.

$$\begin{aligned} C &= N/K, \\ F_{ext}(n) &= F_{bucket,ext}(k), n \in (C(k-1), Ck), k \in (0, K) \end{aligned} \tag{3.22}$$

For the contraction variable the same constraint was implemented as in Equation 3.22. These constraints ensured that the force value at any node, n, was equal to the force value of the bucket of nodes that it was a part of. This technique allowed the trajectories to have a large amount of nodes for a detailed solution, while ensuring that the solenoid switching delay was accounted for. Figure 3.2 further illustrates how the node bucketing worked.

3.1.5. Bounds

For each variable without specified bounds, the solver could search through any value from positive infinity to negative infinity. Let us use the height of the robot body, z_{body},

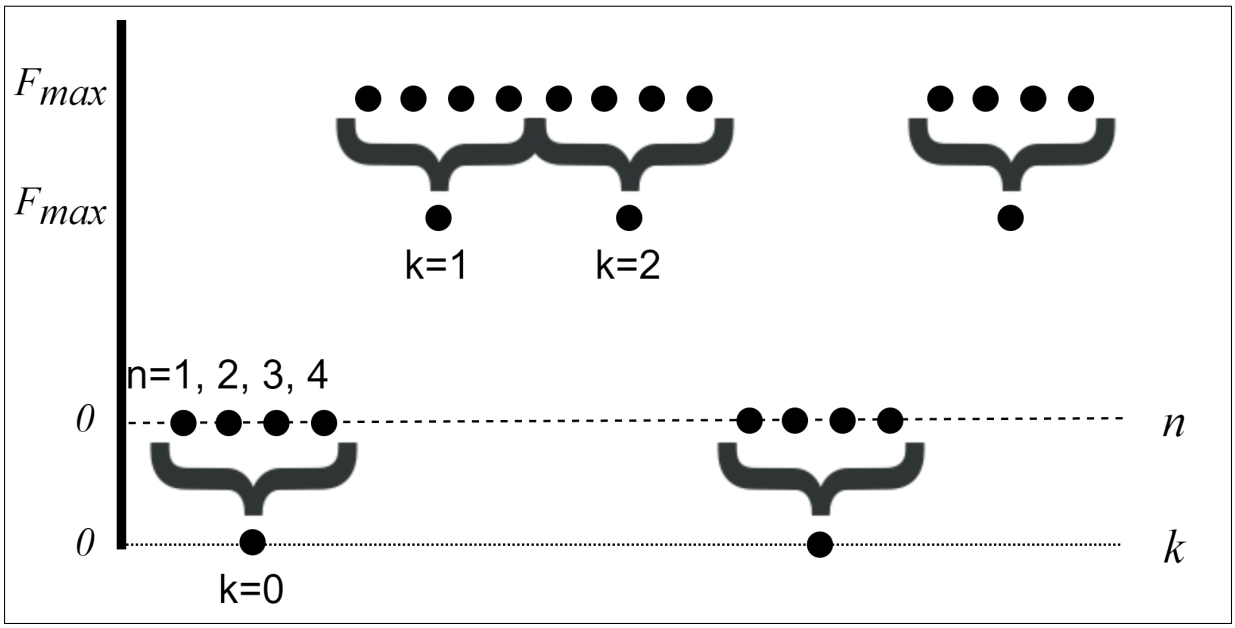

Figure 3.2: An example of node bucketing with $N = 20$ and $K = 5$. The visual representation of the technique shows how a single k-node is equated to a number of n nodes.

as an example; there is a realistic range of values that the height could be. The minimum height is a few centimeters below the minimum leg length, and the maximum height is limited by the acceleration capabilities of the prismatic actuator. If there were no bounds the solver could start at an infeasible value and the solver might never be able to find an optimal solution. Adding bounds minimizes the search domain and eases finding an optimal solution while ensuring feasible trajectories. Certain bounds were implemented on variables where the realistic range was known due to physical and actuation limitations, or in the case of a body angle where the range of values from 0 to 2π provide the same answer as the range from 2π to 4π.

For the robot body height we used the minimum leg length minus 5cm as our lower bound, and $50cm$ as our upper bound. The configuration of the support rig limited the height to $0.5m$. The x-position of the robot body, x_{body}, had a lower bound of $0m$ since all our trajectories would start at $0m$. The support rig of the real robot also limited the movement in the x-direction to $1m$, making $1m$ our upper bound. The leg length bounds were defined by the pneumatic cylinder model that we used. A Festo DSNU-16-125-P-A pneumatic cylinder was used with a minimum length of $0.235m$ when contracted, and a maximum length of $0.38m$ when fully extended. These bounds are shown below.

$$\begin{aligned} z_{body} &\in (0.185, 0.5)m \\ x_{body} &\in (0.0, 1.0)m \\ r_{leg} &\in (0.235, 0.38)m \end{aligned} \tag{3.23}$$

The leg and body angles could also be bound. Instead of any physical limitations enforcing the bounds, they were bounded on the premise that realistic trajectories would not exceed certain angle and angular velocities. If the leg angle goes to 180 degrees, it would not be able to push the robot vertically off the ground, therefore it exceeds realistic trajectory angles. A balanced body was one of the goals of the trajectories and therefore it should not exceed certain angle bounds. The leg angular velocity was bounded by the torque actuator used, a DS5160 servo motor. The body's angular velocity was bounded by inertial forces. These bounds are shown below.

$$\begin{aligned} \theta_{body} &\in (-\pi/4, \pi/4) rad \\ \dot{\theta}_{body} &\in (-1.0, 1.0) rad/s \\ \theta_{leg} &\in (\pi/3, 2\pi/3) rad \\ \dot{\theta}_{leg} &\in (-8.06, 8.06) rad/s \end{aligned} \tag{3.24}$$

Other bounds that were implemented are the reaction forces obtained by the optimizer through complementarity constraints. How these were obtained will be explained in the next sub-section. The GRFs, λ_z, λ_x and the reaction forces on the end of the leg, $F_{extent}^{reaction}$, $F_{contract}^{reaction}$, were bounded. To allow realistic value ranges they were bounded to not exceed values more than 10 times the weight of the robot (mg). They were bounded as follows.

$$\begin{aligned} \lambda_z &\in (0, 10 m_{robot} g) N \\ \lambda_x &\in (-10 m_{robot} g, 10 m_{robot} g) N \\ F_{extent}^{reaction} &\in (0, 10 m_{robot} g) N \\ F_{contract}^{reaction} &\in (0, 10 m_{robot} g) N \end{aligned} \tag{3.25}$$

Once these bounds are defined in the model, a faster and feasible solution becomes possible. The following subsections show how ground contacts, reaction forces and prismatic actuators were modelled using additional constraints.

3.1.6. Initial and Terminal Conditions

To generate specific types of trajectories, for instance acceleration or steady-state trajectories, initial and terminal conditions should be set that define the velocity and position of generalized coordinates of the robot. A combination of fixed value conditioning and constraints were used to implement these conditions.

For convenience a mono-pod robot executing a steady-state hop was conditioned to start

and end its velocity in the z-direction at zero. This condition ensured the apex of the hop took place at the start and the end of the trajectory and also ensured a realistic apex height. The hopping distance in the x-direction was implemented by fixing the terminal x variable to a predefined value, $x_{distance}$.

To ensure a steady-state hop occurred, initial conditions had to be constrained to be equal to the terminal condition, with the exception of the x-position as the robot hopped forward. The constraints and fixed value conditions for a steady-state trajectory can be seen below in Table 3.1.

Table 3.1: Conditions for a steady-state trajectory.

Steady-state Conditions		
Condition	Fixed	Constrained
$z(1) = z_{apex}$	X	
$z(N) = z_{apex}$		X
$\dot{z}(1) = 0.0$	X	
$\dot{z}(N) = 0.0$		X
$x(1) = 0$	X	
$x(N) = x_{distance}$		X
$\dot{x}(1) = \dot{x}(N)$	X	
$\theta_{leg}(1) = \theta_{leg}(N)$		X
$\dot{\theta}_{leg}(1) = \dot{\theta}_{leg}(N)$		X
$\theta_{body}(1) = \theta_{body}(N)$		X
$\dot{\theta}_{body}(1) = \dot{\theta}_{body}(N)$		X

An acceleration trajectory for a mono-pod robot had to start at complete rest with the robot in a neutral position. The terminal conditions for an acceleration trajectory had to follow the initial and terminal conditions from the accompanying steady-state trajectory. Below the significant initial and terminal conditions for an acceleration trajectory are shown in Table 3.2.

The opposite is needed for a deceleration trajectory. All the terminal conditions had to allow the robot to come to rest in a neutral position. The initial conditions had to be constrained to the resulting steady-state trajectory to create a full long time horizon trajectory. The significant deceleration trajectory conditions can be seen below.

The initial and terminal conditions in Tables 3.1, 3.2 and 3.3 show the importance of generating a steady-state trajectory first. Without the steady-state trajectory it was impossible to generate the accompanying acceleration and deceleration trajectories. Once the separate trajectories were generated, they had to be stitched together and interpolated to enable detailed analysis.

Table 3.2: Conditions for an acceleration trajectory.

Acceleration Conditions		
Condition	Fixed	Constrained
$z(1) = z_{min}$	X	
$z(N) = z_{apex}$		X
$\dot{z}(1) = 0.0$	X	
$\dot{z}(N) = 0.0$		X
$x(1) = 0.0$	X	
$\dot{x}(1) = 0.0$	X	
$\dot{x}(N) = \dot{x}_{steady-state}$		X
$\theta_{leg}(N) = \theta_{leg\ steady-state}$		X
$\dot{\theta}_{leg}(N) = \dot{\theta}_{leg\ steady-state}$		X
$\theta_{body}(N) = \theta_{body\ steady-state}$		X
$\dot{\theta}_{body}(N) = \dot{\theta}_{body\ steady-state}$		X

Table 3.3: Conditions for a deceleration trajectory.

Deceleration Conditions		
Condition	Fixed	Constrained
$z(1) = z_{apex}$	X	
$z(N) = z_{min}$		X
$\dot{z}(1) = 0.0$	X	
$\dot{z}(N) = 0.0$		X
$x(1) = 0.0$	X	
$\dot{x}(1) = \dot{x}_{steady-state}$	X	
$\dot{x}(N) = 0.0$		X
$\theta_{leg}(1) = \theta_{leg\ steady-state}$	X	
$\dot{\theta}_{leg}(1) = \dot{\theta}_{leg\ steady-state}$	X	
$\theta_{body}(1) = \theta_{body\ steady-state}$	X	
$\dot{\theta}_{body}(1) = \dot{\theta}_{body\ steady-state}$	X	

3.1.7. Trajectory Post Processing

Stitching

Since trajectory optimization is a computationally expensive process and often struggles to come to a feasible solution for complex trajectories, a technique called trajectory stitching was implemented [9]. As shown in Section 3.1.6, the acceleration, steady-state and deceleration stages of the trajectory was executed as separate optimization problems. Once feasible and optimal trajectories were generated for the separate stages, the results were put together to form a long time horizon trajectory from rest, to steady-state motion, and then back to rest, while travelling a fixed distance. The amount of repeated steady-

state hops were chosen manually since an infinite amount is possible. Seeing the results of a full trajectory stitched together allowed for better insight into the behaviour of the trajectory and could be used to inspire control strategies. The equation below shows how the trajectories were stitched together.

$$\begin{aligned} q &= q_{acc}(1:N_{acc}) + q_{ss}(1:N_{ss}) + ... + q_{ss}(1:N_{ss}) + q_{dec}(1:N_{dec}) \\ \dot{q} &= \dot{q}_{acc}(1:N_{acc}) + \dot{q}_{ss}(1:N_{ss}) + ... + \dot{q}_{ss}(1:N_{ss}) + \dot{q}_{dec}(1:N_{dec}) \end{aligned} \tag{3.26}$$

If the three separate trajectories were generated optimally, an optimal trajectory is possible for the full trajectory [64]. The trajectory stitching technique decreased the solving time of the trajectories, as smaller problems are less complex to solve, and eased the process of finding optimal trajectories for a problem.

Interpolation

Once the trajectory was stitched together a full trajectory from start to finish could be displayed. A problem with the stitched trajectories were the varying time intervals between nodes. These intervals varied on a scale of $1:7.5$ as mentioned in section 3.1.2. This variation made the analysis of the trajectories difficult since there was not an even distribution of data attached to time.

To overcome this problem interpolation was used to fill in gaps between node points and create data with a fixed time interval. The trajectory data were interpolated to a time interval of $5ms$. The interpolation allowed for time realistic animations of the trajectory and eased analysis of the trajectory as well as allowing these trajectories to be compared to data sampled at $200Hz$ by a microprocessor.

Linear interpolation was used to represent the unknown data points at fixed intervals. The linear interpolation method assumes that if an unknown data point, x, y, exists between two known data points, $x_0, y_0; x_1, y_1$, the gradient at the unknown data point will be the same as the gradient of the known data points. The following equation is the formula used in linear interpolation.

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \tag{3.27}$$

Although linear interpolation is considered inaccurate, there was no need for more accurate interpolation methods since these trajectories were used for analysis and direct implementation on a fixed bodied robot only.

3.2. Compliant Terrain in Trajectory Optimization

Rigid terrain applies an instantaneous force to the model when contact is made, this was modelled using a complementarity constraint as seen in Section 3.1.3. For compliant terrain the force applied during contact increases as the terrain deforms and decreases as the relative velocities of the objects become less. For the compliant terrain used in this project, gravel, an additional characteristic causes the terrain to become rigid after deformation.

To model compliant terrain in trajectory optimization a combination of prescribed contact order and complementarity constraints were used [1] [50]. At the nodes where contact order was prescribed a spring and damper system represented the force applied during contact. As the foot reached the bottom of its maximum penetration, the complementarity constraint for rigid terrain was implemented. This mimicked the compression the foot would apply to a gravel surface. Even with the time variation allowed in the model, Section 3.1.2, it was important to accurately prescribe contact order to generate feasible trajectories.

Initially a full model with only rigid terrain was implemented with the same objective as the desired compliant terrain model. As seen in Figure 3.3, the node and x-position at which the foot hit the ground, $n_{penetration}$, and left the ground, n_{exit}, during the rigid terrain trajectory was used to prescribe a contact order for the compliant terrain model. The node where it was necessary to switch from a compliant terrain to a deformed rigid terrain, n_{nadir}, was selected to be halfway between the penetration and the exit node. To prescribe a contact order, additional constraints were required to ensure the foot was in contact with the terrain at certain node points. For the compliant terrain model, a virtual ground was described as a specified distance, d, above zero. This allowed penetration of the ground to take place in the positive z-domain for reasons that will become clear later in this section. For the first half of the prescribed contact nodes, $(n_{penetration} : n_{nadir})$, a compliant terrain was modelled.

Spring and damper forces were implemented as the ground reaction force on the foot. This allowed the foot to penetrate past the specified distance, d. The foot position, z_{foot}, was also forced to be equal to or less than the specified distance to ensure penetration of the

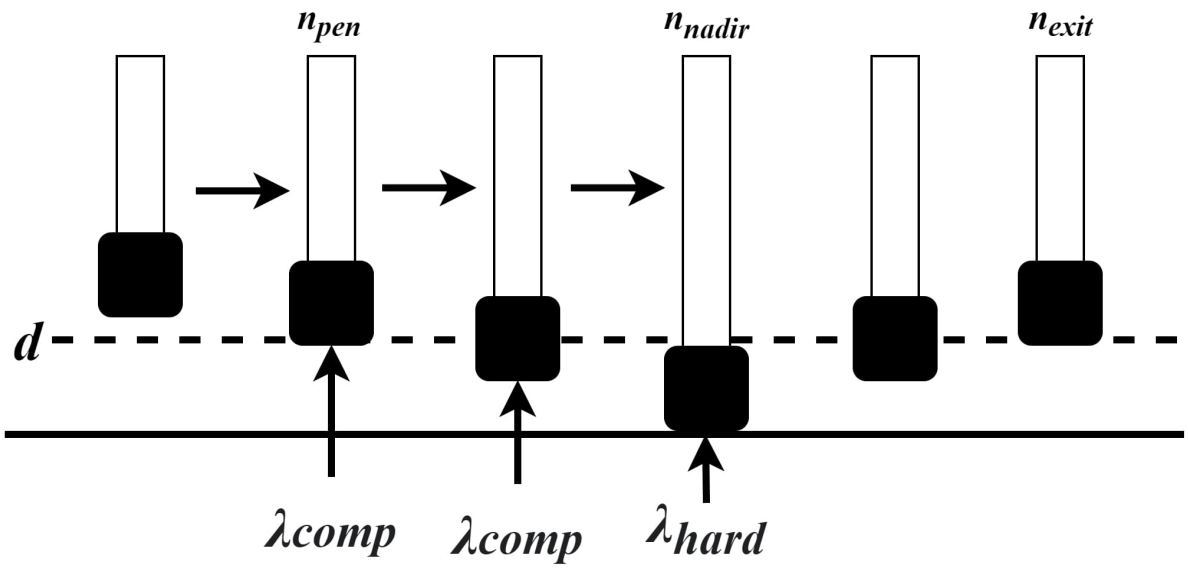

Figure 3.3: Image showing the progression of the ground reaction force on compliant terrain as the foot penetrates below d

compliant terrain took place. The equations below show the constraints implemented to model the compliant terrain.

$$\begin{aligned} z_{foot}(n) &\leq d, \\ \lambda_z(n) &= z_{foot}(n)k_{ct} + \dot{z}_{foot}(n)c_{ct}, \\ n &\in (n_{pen}, n_{nadir}) \end{aligned} \tag{3.28}$$

The spring and damper coefficients, k_{ct} and c_{ct}, were chosen by analyzing a similar rigid terrain trajectory. The velocity of the foot just before it hits the ground in the rigid terrain trajectory was used as an approximate penetration velocity at distance d for the compliant terrain trajectory. The amount of force, F_a, needed to let the foot come to rest at $z_{foot} = 0$ and $n = n_{nadir}$ was calculated and acceptable coefficients were calculated [65].

$$\begin{aligned} c_{ct} &= F_a/\dot{z}_{rigid\ terrain\ \ foot}, \\ k_{ct} &= F_a/d \end{aligned} \tag{3.29}$$

For the second half of the prescribed contact, $(n_{nadir} : n_{exit})$, a rigid terrain complementarity constraint was used to generate the ground reaction force. The rigid terrain complementarity constraint is an approximate model of the terrain after deformation has taken place on the gravel. At this point no more deformation was possible and the terrain was effectively a hard surface.

At the exit node, n_{exit}, the foot was forced to be above the ground ensuring that no contact

would take place any further in the trajectory. This technique for modelling deforming compliant terrain could be used for multiple contact trajectories as well. Figure 3.3 shows how the dynamics of the ground reaction force changed as the foot penetrated below the specified distance, d.

3.2.1. Drop Test

The specified distance, d, that the foot will penetrate into the ground had to be obtained for different compliant terrain types. The apex height, z_{apex}, was a predefined condition and could therefore be used to find the penetration distance, d, by dropping the physical robot from the apex height and measuring the penetration distance.

This method was called a drop test and was done at least 20 times on each terrain type to find an average penetration distance for each terrain type. The penetration distance was then defined in the trajectory optimization model and used to model the effect of the compliant terrain.

3.3. Trajectory Inspired Control

To execute trajectories on a robotic platform using the raw actuation values generated by the optimizer was a challenging task. While methods were used to model the robot as accurately as possible, discrepancies between the robot model and real world robot still existed. This was caused by approximations used to simplify the model such as friction coefficients, weight distributions and the inertial influence of a support rig. It was therefore needed to implement a method of control that would make adjustments to attempt to adhere to the specified trajectory.

Looking at certain significant events in the generated trajectory assisted in finding the best control method for our robot. Certain repeated phases in the robot movement became apparent. Firstly the height at which the pneumatic cylinder expands and contracts remained consistent to achieve a consistent apex height for the robot. Secondly the angle of the leg as it touched down on the ground and launched off the ground was important to achieve the desired robot velocity in the x and z direction.

Two separate, basic controllers were implemented. One to control the pneumatic cylinder and the other to control the servo actuator at the hip.

3.3.1. Pneumatic Control

To ensure the robot reached the desired apex height as it hopped, the pneumatic cylinder had to expand and contract at the right time. If the pneumatic cylinder starts expanding as the robot is at a low height the force of the pneumatic cylinder will act between the robot and the ground for a longer time. This will increase the acceleration of the robot in the positive z-direction and launch it high in the air. If the cylinder expands when the robot is higher in the air, the acceleration will be less and the apex height of the robot will be lower.

The generated trajectories provide the nodes at which the pneumatic cylinder is commanded to expand and contract. From these nodes the robot height where the pneumatic cylinder expands and contracts can be identified.

As mentioned in Chapter 2.2 there is a $20ms$ switching delay in the solenoids that controls the pneumatic cylinder. For the cylinder to expand and contract at the correct height, this delay has to be accounted for. Once again the generated trajectory is used to find the height $20ms$ before the pneumatic cylinder has to expand. Once the height sensor detects that the robot is at a height $20ms$ before the cylinder should expand it commands the solenoid valve to open, and expand the leg.

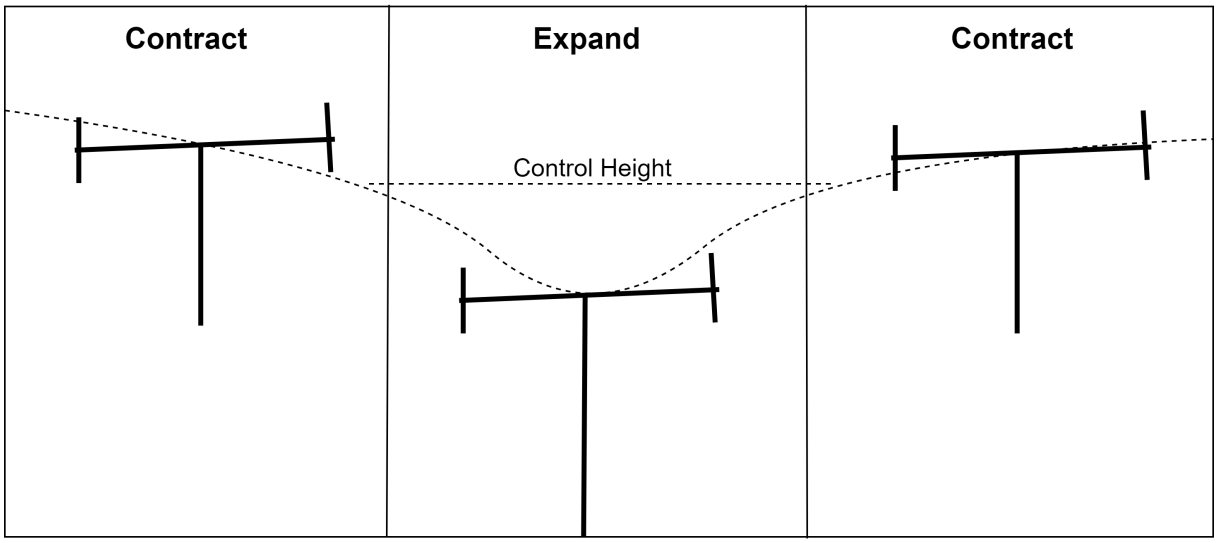

Figure 3.4: The process of the pneumatic control.

3.3.2. Servo Control

The generated trajectories provide the leg and body angle of the robot as it hops. To simplify the control of the servo, and ensure the robot remains upright during hopping,

the controller focused on the angle of the leg and body as the foot of the robot touched the ground and launched off the ground. These two events influence the speed and acceleration of the robot, therefore, ensuring the angles are correct at touchdown and launch is important.

Accelerating, moving at a constant velocity and decelerating required different touchdown and launch angles. A state machine was implemented to allow the launch and touchdown angles to change. As with the pneumatic control, the robot height at which the leg started moving towards the touchdown or launch angles were observed in the generated trajectories. These heights were used as state switching points. When the robot was moving in an upwards direction the servo command would move the leg angle to the current state's touchdown angle. When the robot was moving in a downwards direction the servo command would move the leg angle to the current state's launch angle as well as moving the controller to the next state.

Moving to a touchdown angle when moving upwards and a launch angle when moving downwards seems counter intuitive but it is necessary to account for the speed at which the robot is moving. As the robot is moving downwards it will soon touch the ground and already needs to start moving the leg towards the launch angle since the impact event is very fast, and as the robot is launched into the air it already needs to start moving the leg towards the upcoming touchdown angle. Figure 3.5 shows a visual explanation of the state switching.

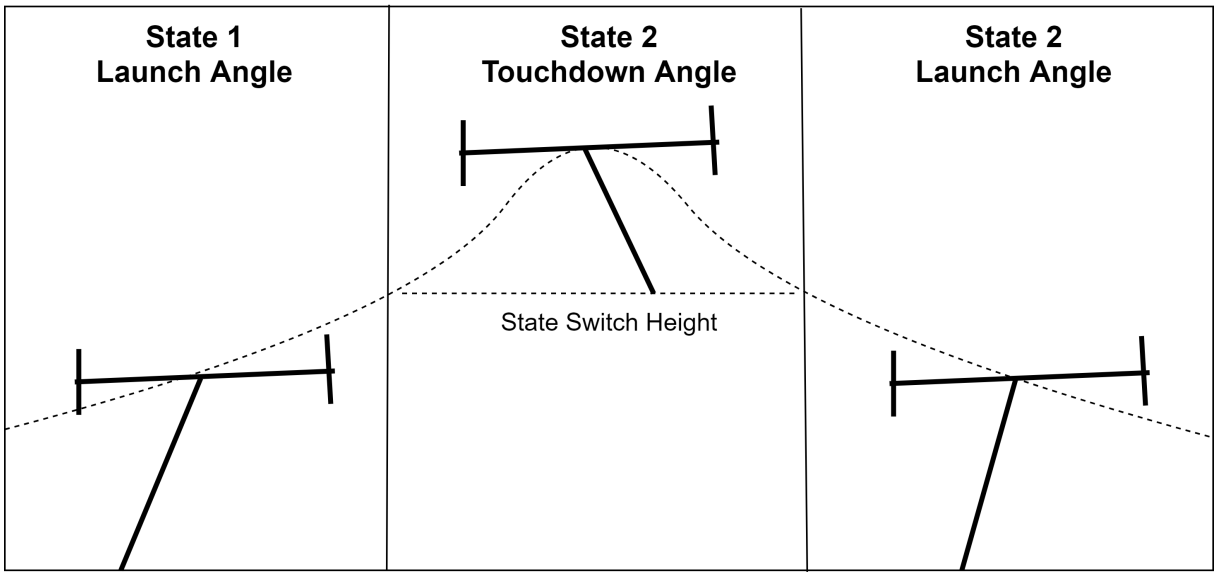

Figure 3.5: The process of the pneumatic control.

Since the robot has a free moving body, and disturbances can influence the body angle, deviation from the body angle of the generated trajectory was expected. To compensate for these deviations the relative leg angle was adjusted as the body angle changed. The

adjustment ensured that the leg angle, when measured clockwise from the x-axis, was correct. The following equation shows how the control was implemented.

$$\theta_{leg}(n) = \theta_{body}(n-1) + \theta_{touchdown}, \tag{3.30}$$

This control allows the leg angle to adjust to the movement of the body and still execute successful consecutive hops.

To test and use the methods detailed in this chapter, a physical robot and support rig had to be built. Chapter 4 discusses how this was done.

3.4. Summary

This chapter covered the necessary methods to be able to used trajectory optimization for legged locomotion simulation and control inspiration on compliant terrain. The setup of a trajectory optimization model was discussed looking specifically at the equations of motion, what it takes to create a trajectory optimization model, how to implement complementarity constraints, how to use node bucketing to mimic a switching delay in a solenoid, how to set up bounds in a trajectory optimization model, initial and terminal conditions that ensure a smaller solve space and trajectory post processing to enable easy data use. After this the method of modelling compliant terrain in trajectory optimization was discussed, including how a drop test was used to find foot penetration depths. And finally the method of using trajectories to inspire the creation of control systems was discussed.

Chapter 4

Robot and Support Rig

An important goal of this project was to validate trajectories and controllers on a physical robotic platform. The trajectory optimization problem had to represent the robotic platform. A mono-pod robot, actuated by a pneumatic cylinder at the leg and a servo motor at the hip was designed and built. Since the model was purposefully restricted to move only in the sagittal plane; a support rig was designed and built around the robot to ensure the movement of the robot was constrained. Additionally the support rig had to supply the robot actuators with the necessary power and air supply to operate as well as determining the states of the robot using encoder sensors. A micro controller was used to execute control and log data to compare to the simulation results.

4.1. Robot Design

As described previously in the methodology above, the robot has a torque actuator at the hip and a prismatic actuator at the leg. This allows the robot to rotate the leg to a desired angle and actuate the leg for contraction or extension at a desired time. In this section the different actuators used will be discussed. The sensors used to collect data and act as input to a control strategy will also be detailed, as well as the electronics that controlled the robot.

4.1.1. Torque Actuator

Turning the hip of the robot fast and accurately is important for transient locomotion. In addition to speed and accuracy the actuator has to withstand high impact torques since the robot makes contact with the ground at different angles [66].

Using the knowledge gathered in our literature review, Section 2.2, a geared servo motor was used to actuate the robot at the hip. The high positional accuracy and lack of

backdrivability of geared motors made them ideal for the positional control. The high impact torques experienced when making contact with the ground had little influence on the angle of the hip, and the positional accuracy ensured the right angles were reached at the right time. Table 4.1 below shows a comparison of the angular range, angular velocity and torque of different available servo motors. All tough backdrivability is desirable in legged robotics to be able to handle the large impact forces, the use of the linear actuator discussed in the following section compensated for the lack of backdrivability in the geared servo motor.

Table 4.1: Comparison between different types of high torque servo motors

Servo Motor Comparison			
Servo Motor Model	Angular Range	Angular Velocity $[rad/s]$	Torque $[kg.cm]$
Feetech FT5335M	120°	5.8	40
DS5160	270°	8.06	60
DS3235	270°	8.73	35

To successfully execute control on the robot, the servo motor had to be able to move to a desired angular position fast and withstand high torque impacts from the foot hitting the ground. This is why the DS5160 servo motor was used since it has the best combination of a angular range, angular velocity and torque.

4.1.2. Prismatic Actuator

There are a number of different prismatic actuators such as hydraulic and electric screw thread, but these are all too heavy and slow to execute rapid robot movements. A pneumatic cylinder was used as our prismatic actuator. This method of prismatic actuation is ideal for rapid acceleration and explosive movement. Since it is a true prismatic actuator the approximations made to model the pneumatic cylinder is minimal. The only major approximation is made when modelling the dynamics of the pneumatic cylinder as a bang-bang force with damping, a delay and hard stops. This is done to simplify the model and has been confirmed to be sufficient [9].

As the solenoids open and close the cylinder expands or contracts at a maximum force, the bang-bang force model is an approximation of this behaviour. As the cylinder expands and contracts it comes to a complete stop when it reaches the ends of the chamber, a hard stop was modelled using complementarity constraints to mimic this behaviour. The damping in the model is added to approximate the compression and flow of the air in the chamber. A delay is added as the solenoids take time to open once it is switched and the air takes time to compress.

The power of the cylinder came from compressed air stored in a compressor. The power output of the compressed air was consistent but could not be regulated dynamically. The only electronic component in the prismatic actuator system was the solenoid valves that controlled the flow of air to the cylinder. The pneumatic cylinder could contract the piston by letting air flow into the bottom of the cylinder chamber, and could expand the piston by letting air flow into the top of the cylinder chamber with the other chamber vented.

A Festo DSNU-16-125-P-A pneumatic cylinder was used as the leg actuator. This pneumatic cylinder was chosen because it had the best size to power ratio for the robot application. A very large pneumatic cylinder would need too much space to execute long time horizon trajectories, and smaller pneumatic cylinders would move too fast and be diffcult to control. The maximum force that this cylinder can apply is 160 N at 4 bar and was powered by a 100 l air compressor [67]. Two Festo CPE10-M1BH-5L-QS-6 solenoid valves were used to switch between cylinder contraction and expansion.

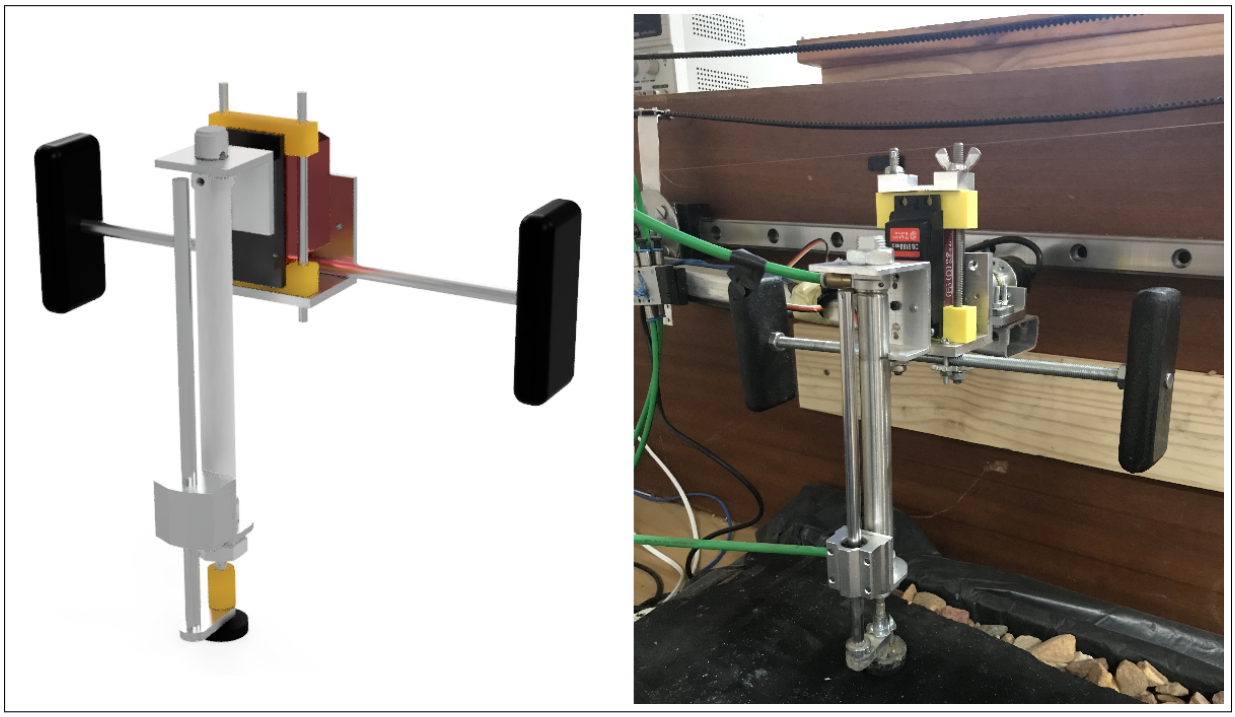

Figure 4.1: CAD model of the robot including the linear support rail compared to the fully built robot

4.1.3. Electronics

To execute the desired trajectories and implement the designed controllers, a Teensy 4.0 micro-controller was used. The Teensy 4.0 has an ARM Cortex-M7 processor that runs at 600MHz with 40 digital IO pins and 14 analog input pins. Serial communication was used to send the necessary data from the Teensy 4.0 to the computer.

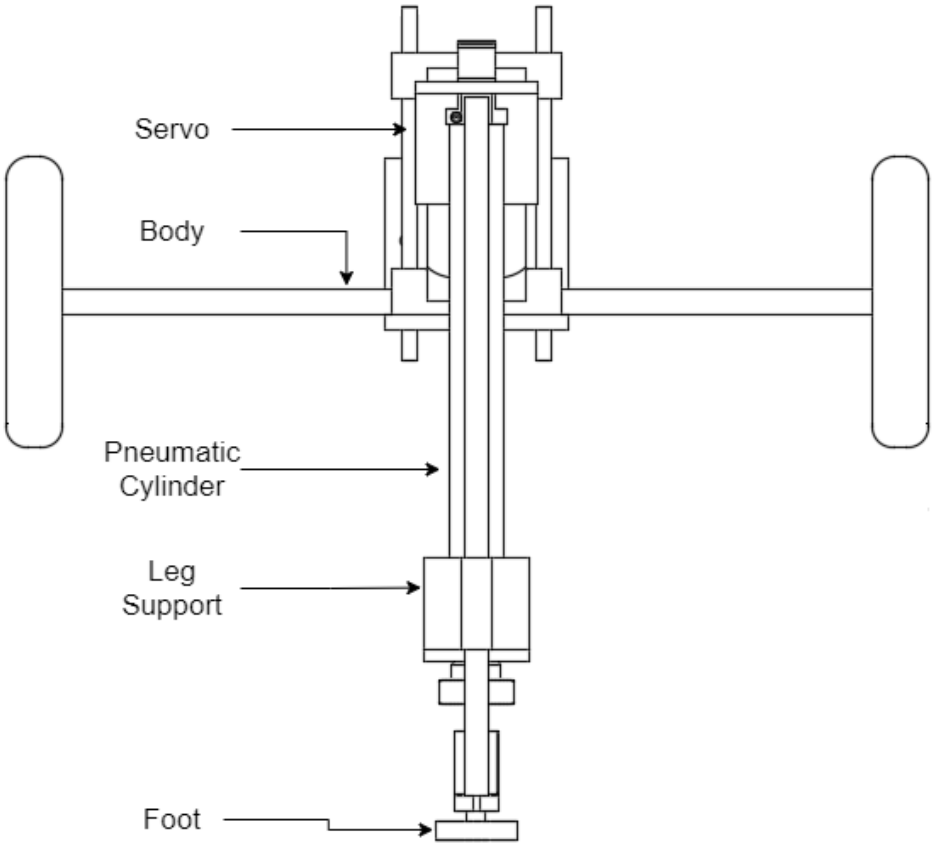

Figure 4.2: A sketch of the final robot design.

All the sensors and actuators were connected to the Teensy 4.0 but ran on separate power supplies as the voltage ratings were different. Below is a table showing the different voltage rating of the different electronic devices used.

Table 4.2: Voltage Ratings of the electronic devices used on the robot

Device Voltages	
Device	Supply Voltage $[V]$
Feetech FT5335M Servo Motor	8.4
Teensy 4.0 MCU	3.3
Festo CPE10 Solenoid	24
Generic Optical Encoders	$4.7 - 24$

Figure 4.3 shows the Teensy connected to the custom designed printed circuit board. The circuit board was designed to connect at least three optical encoders, two MOSFETS and one servo motor to the Teensy GPIO pins. Extra connection points were made available as well to allow for future applications. Since the solenoids and servo motor needed separate power supplies, the PCB was designed to allow 24 V and 8.4 V power distribution to these components respectively. Two $100\mu F$ capacitors were used as decoupling capacitors to keep the voltage supply stable. Transistors were needed to allow the Teensy micro controller to switch the solenoids on or off. Two 2N5191 MOSFETS were connected to the

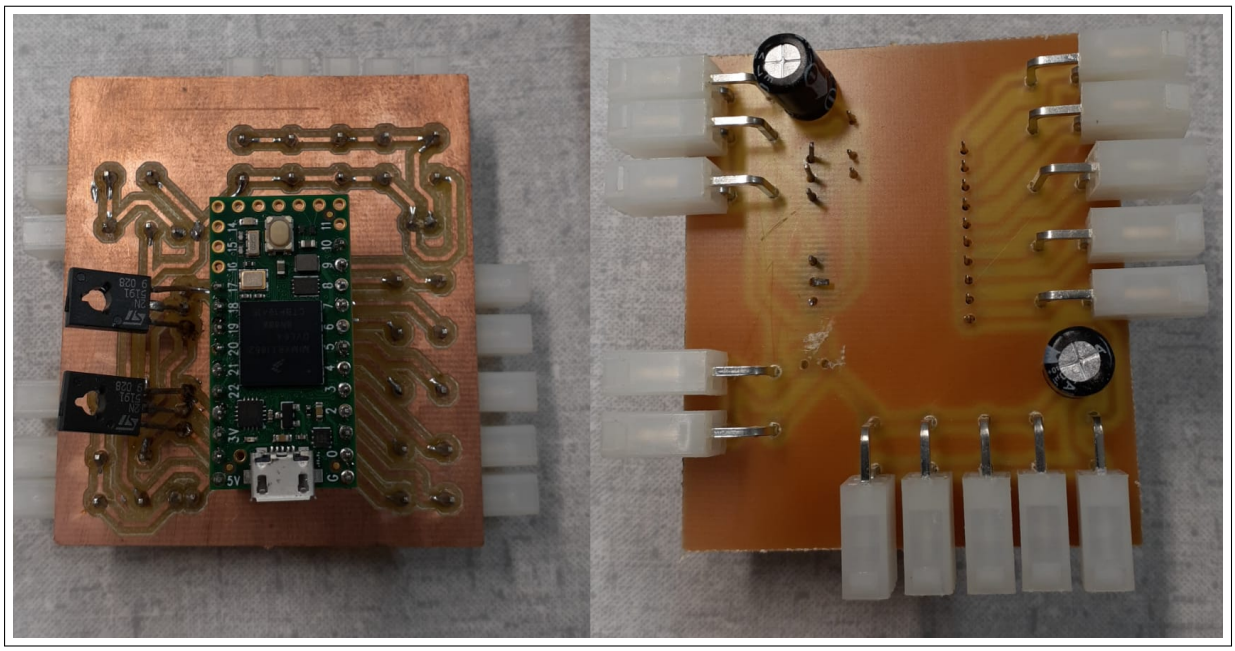

Figure 4.3: Top and bottom view of the designed PCB with the Teensy connected to it.

solenoids and the Teensy micro controller. The base pins were connected to GPIO 22 and 23 of the Teensy micro controller. The emitter pins were connected to ground and the collector pins were connected to the negative connector of the respective solenoids.

Sensors

When the robot was in operation it was important to know the position and orientation of the robot. This ensured that it remained on the desired trajectory and was also important for data collection and executing control.

A combination of optical rotary encoders were implemented to observe the body angle, z-position and x-position of the robot. Generic 1000PPR optical encoders were used to take these readings at a frequency of 200 Hz and the readings were used to execute the desired control strategy. The encoders were placed at specific positions on the support rig mentioned in the design subsection below (4.3.2). These encoders use AB two-phase orthogonal pulse signals to indicate the angle at which it has been turned. This means each encoder needs two GPIO pins pins from the Teensy micro controller. GPIO pins 4-9 were used to implement communication with three rotary encoders on the Teensy micro controller.

4.1.4. Safety Support

As mentioned in Section 4.1.1 when the foot of the robot hits the ground a large impact force was applied to the robot. As compressed air was used at high speeds safety was a concern and something to consider while designing and building the robot.

A linear support rail was developed for the pneumatic piston. The piston itself is 5 mm in diameter and is susceptible to breaking under high lateral force. The goal of the support system was to remove the lateral load from the piston and in case of failure to prevent a projectile from flying into the air. A mechanical fuse was added Figure 4.1 shows the design of the linear support rail in a CAD model, as well as the actual linear support rail that was implemented on the robot.

4.1.5. Mechanical Specifications

To generate and execute feasible trajectories the dimensions and weight of the robot had to be scaled correctly. The fixed dimensions of the pneumatic cylinder and servo motor was used as a starting point to design the rest of the robot.

The maximum length of the leg was specified as 360 mm, which includes the stroke of the pneumatic cylinder, 125 mm, plus the length of the pneumatic cylinder, 235 mm. The length of the body had to scale with the length of the cylinder, if the body was too short the robot would have been too unstable to control, if the body was too long there would have been flex on the body that would lead to instability. The robot body was 300 mm with weighted ends to increase inertia and create stability.

Figure 4.2 shows a front view of the robot and indicates the important parts of the robot. Figure 4.1 compares a rendered CAD model of the robot to the actual robot.

The full specifications of the robot are listed in Table 4.3 . The mass of the robot body, m_{body}, included the effective mass that the support rig added to the robot.

Table 4.3: Key specifications of the robot

Robotic Platform	
Specifications	Value
l_{body}	300 mm
l_{leg}	$271 - 396$ mm
m_{body}	1.6 g
m_{leg}	0.35 kg
F_{max}	160 N
τ_{max}	40 kg/cm
ω_{max}	5.82 rad/s

4.2. Fixed Body Support

Initial experiments were done on a fixed body robot. This was done to confirm the feasibility of initial trajectories without the use of a control strategy. A fixed body robot is essentially a leg with a prismatic actuator attached to a fixed hip motor. The body has no inertial reaction to leg movement, this means no balancing is needed to execute successful trajectories.

It was important to limit the movement of the robot to the sagittal plane to enable the robot to execute the 2-dimensional trajectories. The support rig for the fixed body robot was a boom structure. The one side of the boom was attached to the body of the robot, and the other side was attached to a 2-degrees of freedom pivot. The support rig allowed the robot to move up and down freely, as well as in a circular side to side motion. The circular side to side motion was seen as an approximation of lateral side to side motion since the boom length, $2m$, was long enough. Figure 4.4 shows a side on view of the fixed body support rig and robot. Two optical encoders, mentioned in Section 4.1.3, were

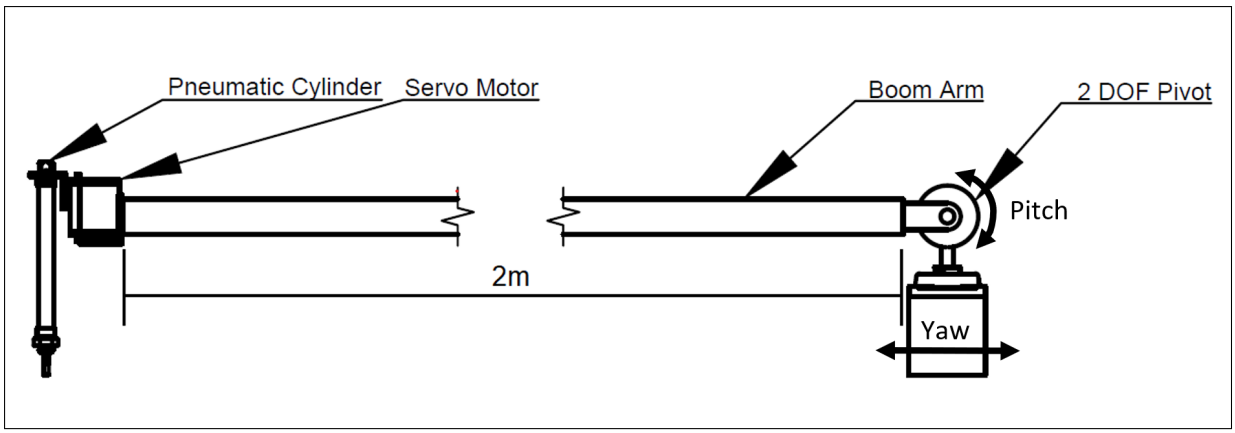

Figure 4.4: Side view drawing of robot platform and fixed body rig indicating degrees of freedom and important components

attached to the pivot on the one end of the boom. The first optical encoder recorded the

z-position of the robot. A 164 tooth gear was attached to the pitch axis of the boom, and the encoder had a 10 tooth gear attached to its shaft. This allowed the encoder to record the z-postition movement of the robot at a gear ratio of 16.4:1. The same was done on the yaw axis to record the x-position of the robot, this time a 107 tooth gear was attached to the axis and allowed the optical encoder to record the x-position at a gear ratio of 10.7:1.

This support rig worked sufficiently for a fixed body setup. Good experimental results were acquired using the support rig. However the weight of the boom, as well as the oscillation of the boom as the robot moved was not ideal for a free body robot. This lead to a redesign of the support rig to better support the free body robot.

4.3. Free Body Support

The final support rig design had to support a free moving body that would react to leg movement. This meant the robot has to balance itself to prevent failure. The balancing of the robot was a challenging problem to overcome, having a very stable and lightweight support rig is important. The problems and limitations of the fixed body support rig assisted in making better design choices for the free body support rig.

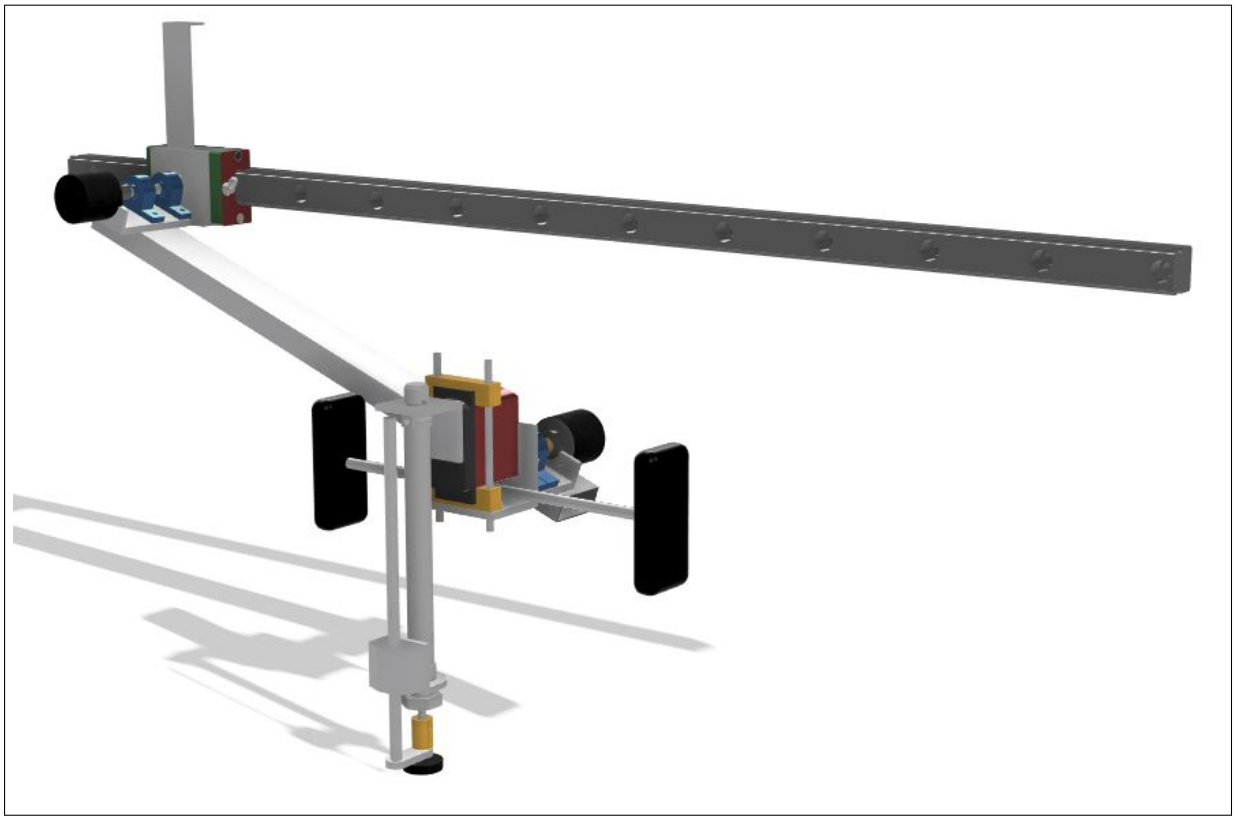

Figure 4.5: 3D rendering of the designed Robot and support Rig

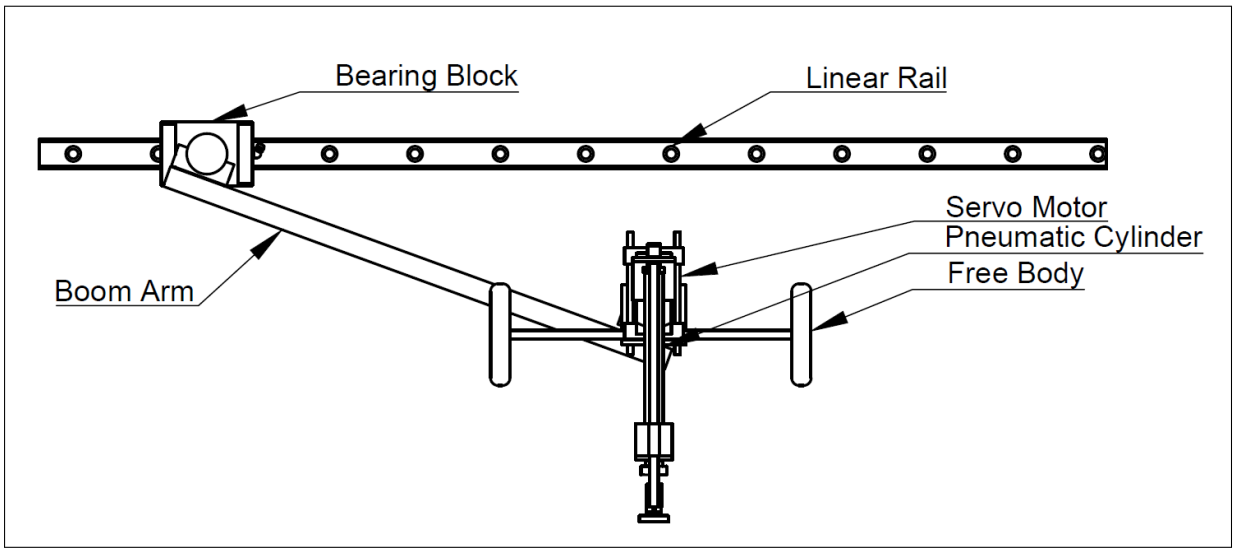

Figure 4.6: Front view drawing of robot platform and free body rig

Figure 4.5 and 4.6 show the designed robot and support rig. The 3D design was done using Autodesk Fusion 360. For the components that were bought 3D models were imported or created using the component measurements. The custom components were 3D printed or machined from aluminium.

4.3.1. Design Priorities

Since the trajectories that were generated using the methods in Chapter 3 never exceeded an x-position of 1 m measured from the start position of the trajectory, the support rig could limit the movement of the robot to 1 m. This helped with space saving and assisted in the design of the support mechanism.

A problem with the fixed body support was the length of the boom. The boom had to be at least $2m$ in length to minimise the circular movement of the robot and to stop the foot from slipping, but this lead to flex in the boom that influenced the movement of the robot. Due to this problem a priority in designing the free body support rig was to shorten any support boom or eliminate the need for a support boom.

When the robot was converted from a fixed body to a free body, one mechanical feature that was added was the weights on the free body, seen in Figure 4.2 and Figure 4.1. The addition of the weights increased the inertia of the body and helped with stability in the robot, but increased the overall weight of the robot and prioritised eliminating any extra weight added by the support rig.

The angle of the body became an important data point to observe with the free body

robot. Designing the support rig with space for 3 optical encoders, one to observe the robot height, one to observe the distance the robot travels and one to observe the angle of the body was also important.

4.3.2. Design

As seen from the design priorities above, the design had to be a light weight, stable support rig that allowed free movement in the sagittal plane.

The final design that was chosen was a boom and linear guide hybrid design. This design had a bearing block on a linear rail guide and a boom attached to a shaft on the bearing block on the one side and a shaft on the robot on the other side. The following subsections describe the design of each component.

Linear Guide

A 1 m long linear guide was used as the main support. The guide was attached horizontally to a wall. A bearing block was attached to the linear guide and allowed robot movement in the x-direction. The linear guide had to be large enough to withstand high lateral forces.

Initially a Machifit MGN12 linear guide and bearing block was considered but proved to be too small to handle the lateral impact forces that the attached shaft would exert on the bearing block. Eventually a larger NSK NH30 linear guide and bearing block was used. It could easily handle the lateral impact forces. The main disadvantage of the larger linear guide was the addition of inertial mass in the x-direction. Table 4.4 shows a comparison of the linear guide and bearing block combinations considered.

Table 4.4: Comparison between different types of linear guides and bearing blocks

Linear Guide and Bearing Block Comparison			
Model	Linear Guide Dims [mm]	Bearing Block Dims [mm]	BB Weight [g]
Machifit MGN12	12x8x1000	27x10x45	50
NSK NH30	30x26x1000	60x30x80	700

Boom Arm

On the bearing block of the linear guide a swiveling boom arm was attached. The boom arm was 50 cm long and allowed movement in the z-direction. The length of the boom

arm when compared to the fixed body support boom removed a lot of oscillation and mass from the system.

Initial 20x20 *mm* aluminium tubing was considered for the boom arm, but it proved to be too thin and had too much compliance. 20x40 *mm* Aluminium tubing was then chosen as the boom arm. Aluminium is light weight and does not add unnecessary amount of weight to the robot, but it is still strong enough to deal with the high impact force and eliminate oscillations. The robot was attached to the one end of the boom arm using two rotary bearings. This allowed the body of the robot to move freely.

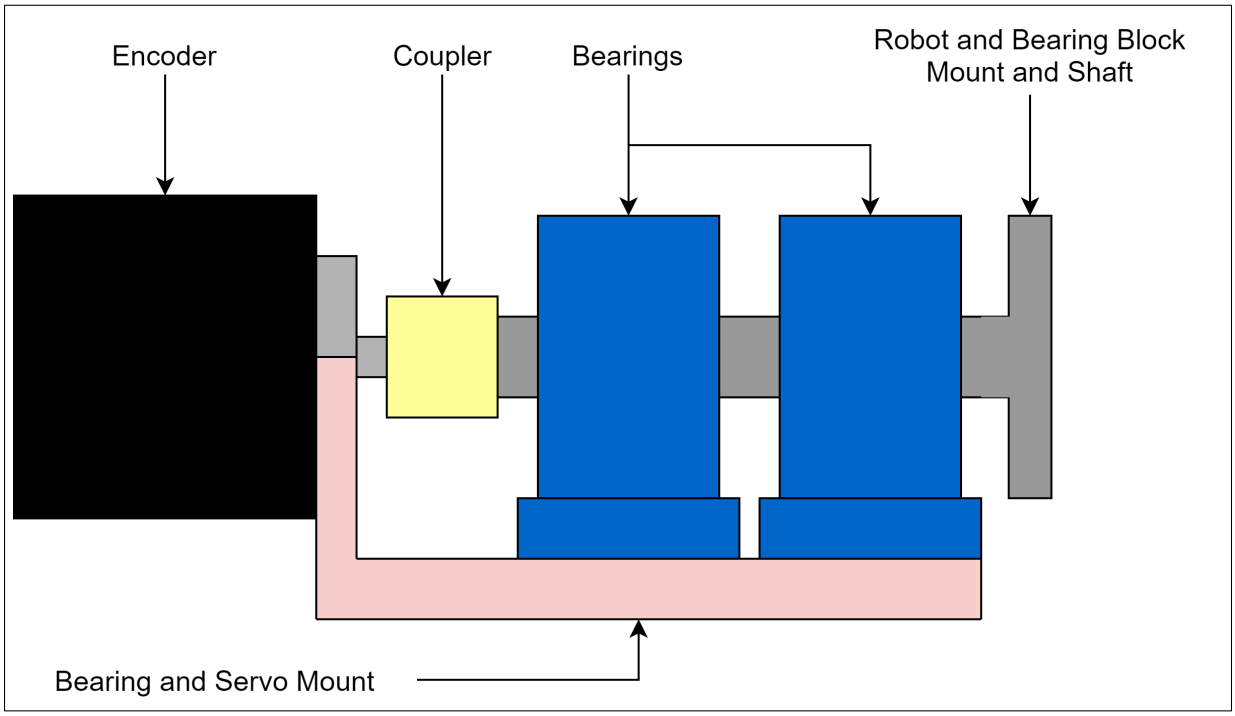

Figure 4.7: Side view of encoder and bearing layout

Attachments

For the robot to move freely in the sagittal plane, the attachment point between the boom arm and the bearing block; and the attachment point between the robot body and the boom arm had to allow free rotational movement. The rotational movement between the body and the boom and the boom and the bearing block, as well as the movement of the linear rail allowed free body movement in the sagittal plane.

Pillow block bearings were used as attachment points. Initially a 15 *mm* pillow block bearings were considered, but the strength of such a bearing was unnecessary and the added weight could have been detrimental to the robot success. 8 *mm* pillow block bearings were used eventually, with reinforced 8 *mm* shafts that attached the boom arm and robot body to these bearings.

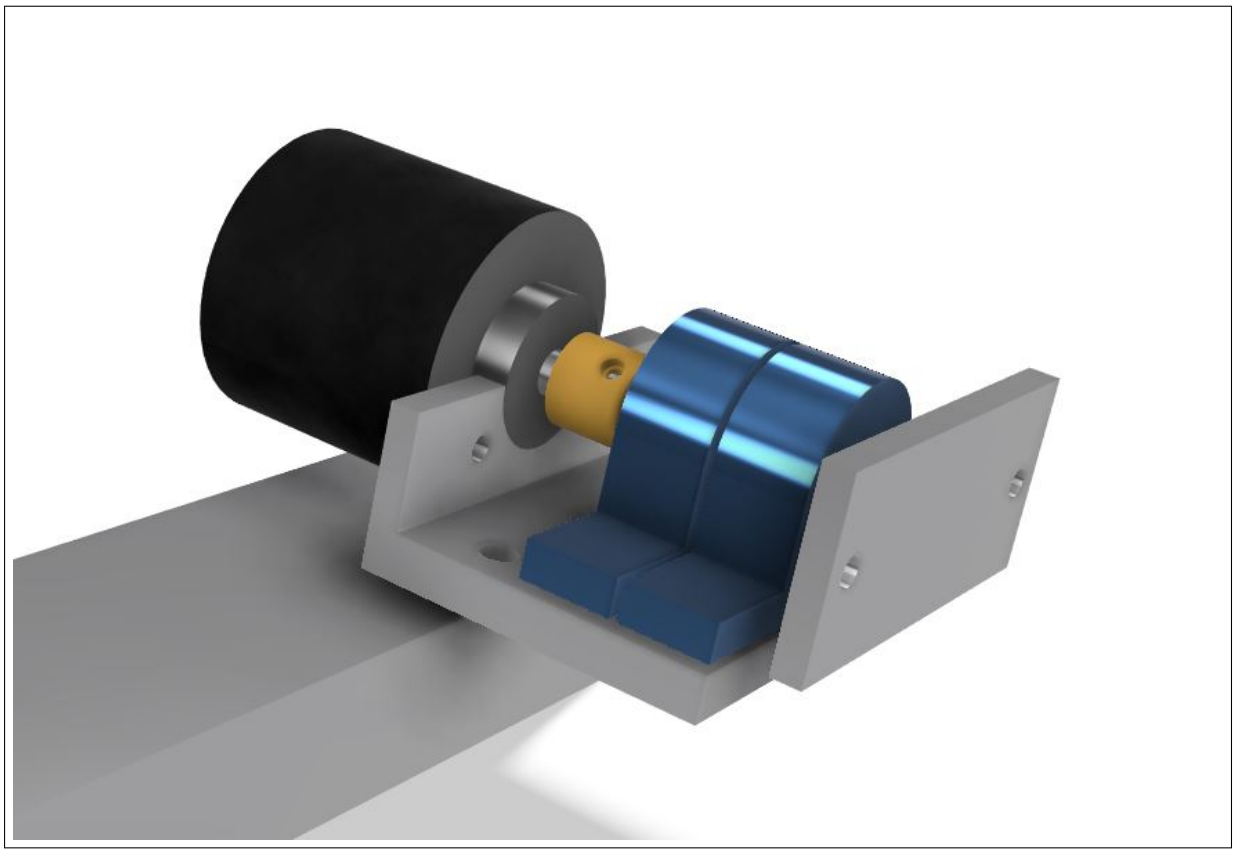

Figure 4.8: Render of the design of the bearing mounts and encoder attachments.

Encoder Attachments

The design included attachment points for optical encoders. The first encoder was attached at the axle where the boom arm met the bearing block. This allowed for the height of the robot (z_{body}) to be measured by measuring the angle of the boom arm. The second encoder was attached at the axle where the robot and the boom arm met. This allowed for the body angle of the robot to be measured. Figure 4.7 show how the encoders at the bearing block and the body of the robot were attached to the support rig using a coupler, a mount and the pillow block bearings. Figure 4.8 shows a rendering of the CAD design done for the encoder attachments at the pivots. The bearing and servo mount and the robot and bearing block mount and shaft were machined from aluminium and the coupler was 3D printed.

A third encoder measured the x-position of the bearing block. This was achieved by designing a belt and pulley system that attaches to the bearing block. The encoder was attached to the axle of the pulley and as the bearing block moved the pulley rotated and the encoder could measure the distance that the bearing block moves.

4.4. Experimental Setup

No control was added to the robot to ensure it stayed on the desired trajectory. Instead the actuation timing of the solenoid valves and servo of the generated trajectories were used. Since the trajectory optimization generated time varying intervals between nodes, the trajectories were interpolated to ensure servo and solenoid actuation instructions were sent at a frequency of 200 Hz. All trajectories used in the experiment started with the mono-pod leg at rest, the pneumatic cylinder fully retracted and the leg vertical.

For all experiments on rigid terrain, a wooden floor was used, while the foot of the mono-pod leg was rubber to minimize slipping. It was assumed that the compliance of the rubber foot was negligible. For experiments on gravel, a gravel pit, shown in Figure 4.9, was build for the robot to execute the trajectories on. Two types of gravel was used, also shown in Figure 4.9. A rough gravel with an average rock diameter of 27.4 mm, and a fine gravel with an average rock diameter of 5.2 mm. All trajectories travelled less than 1 m in the x direction to avoid exceeding the width of the gravel pit. An average steady-state velocity of 0.6 m/s (2.55 $leg\ lengths/s$) was achieved on all trajectories.

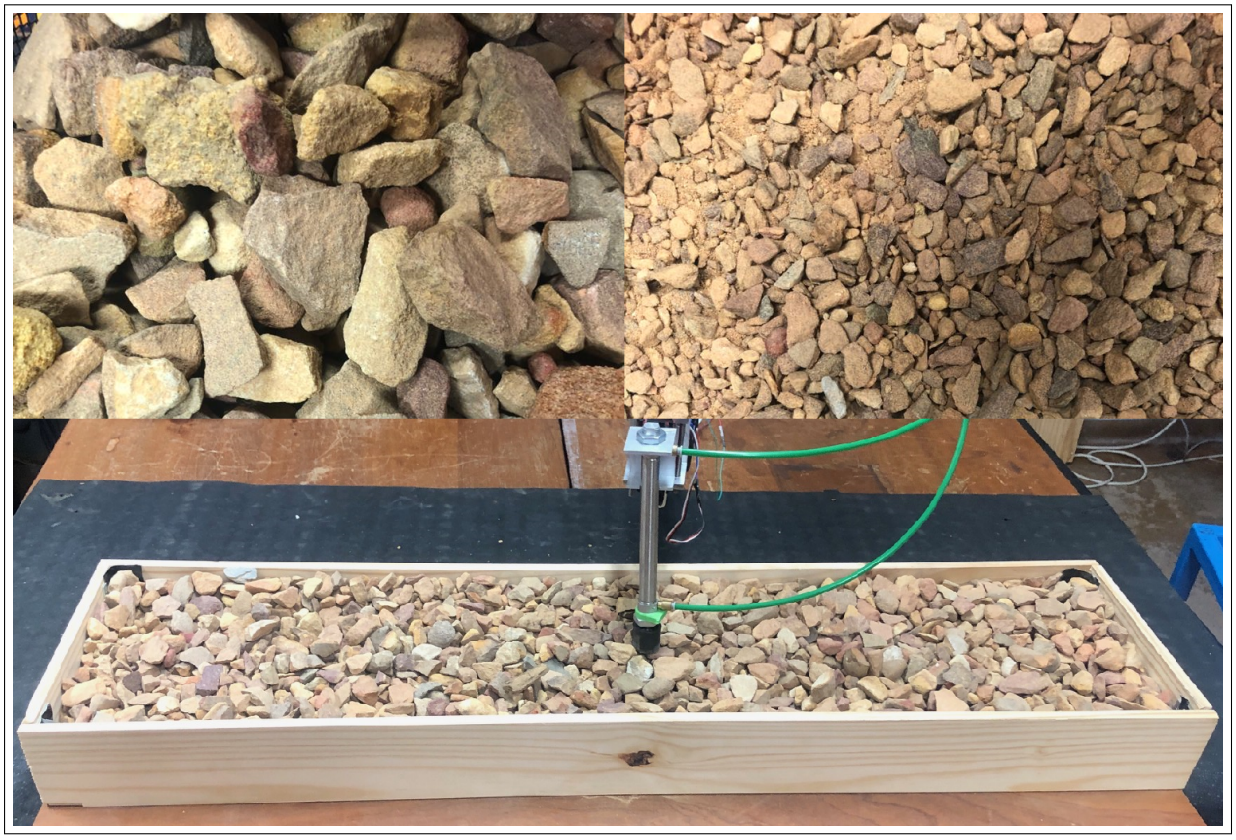

Figure 4.9: Image of gravel pit with robotic platform and the two types of gravel used

4.5. Summary

This chapter started with a discussion of the design and implementation of the mono-pod robot. It provided details of the design choices made around the robot including which torque actuator and prismatic actuator was used and what type of sensors and micro controller was used. It went into further design details discussing the different support rigs that were designed and mechanical features surrounding these support rigs. The design priorities was discussed and details surrounding the components of the final design was discussed. Information about the experimental setup and different contact surfaces was also covered.

Chapter 5

Fixed Body

As a preliminary to the final outcome of this thesis, the process of generating trajectories and confirming their viability on a fixed body mono-pod was done. A fixed body improves the stability of the robot and the reduction in generalized coordinates simplified the trajectories and controllers. This chapter will discuss what needs to be set up to generate fixed body trajectories for rigid terrain and compliant terrain surfaces. The trajectory results will be used to confirm that the different methods such as the description of rigid terrain (Section 3.1.3), compliant terrain (Section 3.2) and modelling the pneumatic actuator (Section 3.1.3 and Section 3.1.4) work. Lastly the generated trajectories will be executed on a fixed body robot and will be compared to the corresponding generated trajectory to confirm the accuracy of the trajectories. These experiments will confirm that the generated trajectories can be applied to a robotic platform successfully.

5.1. Optimizer Model Setup

To generate an accurate trajectory for a desired movement a trajectory optimization problem need to be set up. The following sections discuss the different variables of the optimization problem that needs to be constrained and fixed to generate the desired motions. This provides a more detailed look at the optimization problem that was discussed in the methodology chapter.

5.1.1. Time and Nodes

The different trajectory types (acceleration, steady-state and deceleration) will run at different time lengths (T) to achieve the desired motions. As the time length increases the amount of nodes needed to acquire an accurate trajectory also increases. The following table shows the time values and amount of nodes that were used for specific trajectories.

These time values and amount of nodes were used on every terrain type.

Table 5.1: Node ranges for different trajectory types.

Node Selection Range		
Trajectory	**T** (s)	Nodes
$Steady - State$	1.0	70
$Acceleration$	1.5	120
$Deceleration$	0.8	40

The values showed in table 5.1 were found by looking at other trajectory optimization problems and eventually by trial and error [9].

5.1.2. Initial and Terminal Conditions

As discussed in Section 3.1.6 to generate separate steady-state, acceleration and deceleration trajectories, different initial and terminal conditions are needed.

For all of the fixed body trajectories the apex height (z_{apex}) was chosen to be equal to 0.45 m. This height allows for significant pneumatic actuation but still remains within the range of the support rig. A steady-state hopping distance ($x_{distance}$) of 0.2 m was used, this allowed several steady-state hops within the 1 m x-position limit of the support rig.

The initial and terminal conditions discussed in Section 3.1.6, such as the acceleration velocities starting at $\dot{q}(1) = 0$ and the deceleration velocities ending at $\dot{q}(N) = 0$, were implemented. For more detail about the initial and terminal conditions of the separate trajectories refer to Section 3.1.6.

5.1.3. Drop Test

To generate rough gravel and fine gravel trajectories, a penetration depth, distance d, was defined by performing a drop test. Distance d represented an average depth that the foot penetrates the different types of gravel from its original average height (Section 3.2). The robot platform was dropped from a height of 45 cm at different positions in the gravel pit. The 45 cm distance was an expected apex height derived from rigid terrain trajectories. The z-position of the robot foot was measured after every drop and a penetration depth average for each type of gravel was found. Table 5.2 shows the results of the drop test.

Due to the foot moving and penetrating the ground at different speeds and angles during actual hopping the distance d was not seen as a precise representation of the surface but

Table 5.2: Average penetration depth on two types of gravel

Drop Test Results		
Gravel Type	Avg Gravel Diameter $[mm]$	Avg Pen Depth d $[mm]$
Rough	27.4	21.8
Fine	5.2	39.2

rather a good estimate validated by testing on the robot platform.

5.1.4. Additional Variables

Table 5.3 below shows all the additional values that need to be declared and defined in the model to generate feasible trajectories.

Table 5.3: Additional values that are declared in the model.

Additional Values	
Description	Value
Body Weight	1.65 kg
Leg Weight	0.35 kg
Pneumatic Damping Coefficient	36.8

5.2. Trajectory Analysis

Trajectories are generated for three different terrain types, a rigid terrain that uses the complementarity constraint to describe the surface, and rough and fine gravel terrains that both use the method described in Section 3.2. Once these trajectories are generated they can be used to confirm whether the methods described in Chapter 3 generate realistic and usable trajectories.

As shown in Section 3.1.7 the generated variable time-step trajectories for each terrain type are interpolated to a fixed time step of 5 ms and stitched together as needed. The stitching creates a long time horizon trajectory that starts and stops with the robot at rest. The interpolation allows for easier analysis by creating time realistic graphs and animations. The interpolation also allows the necessary data to be uploaded to a micro controller for execution at a fixed time step.

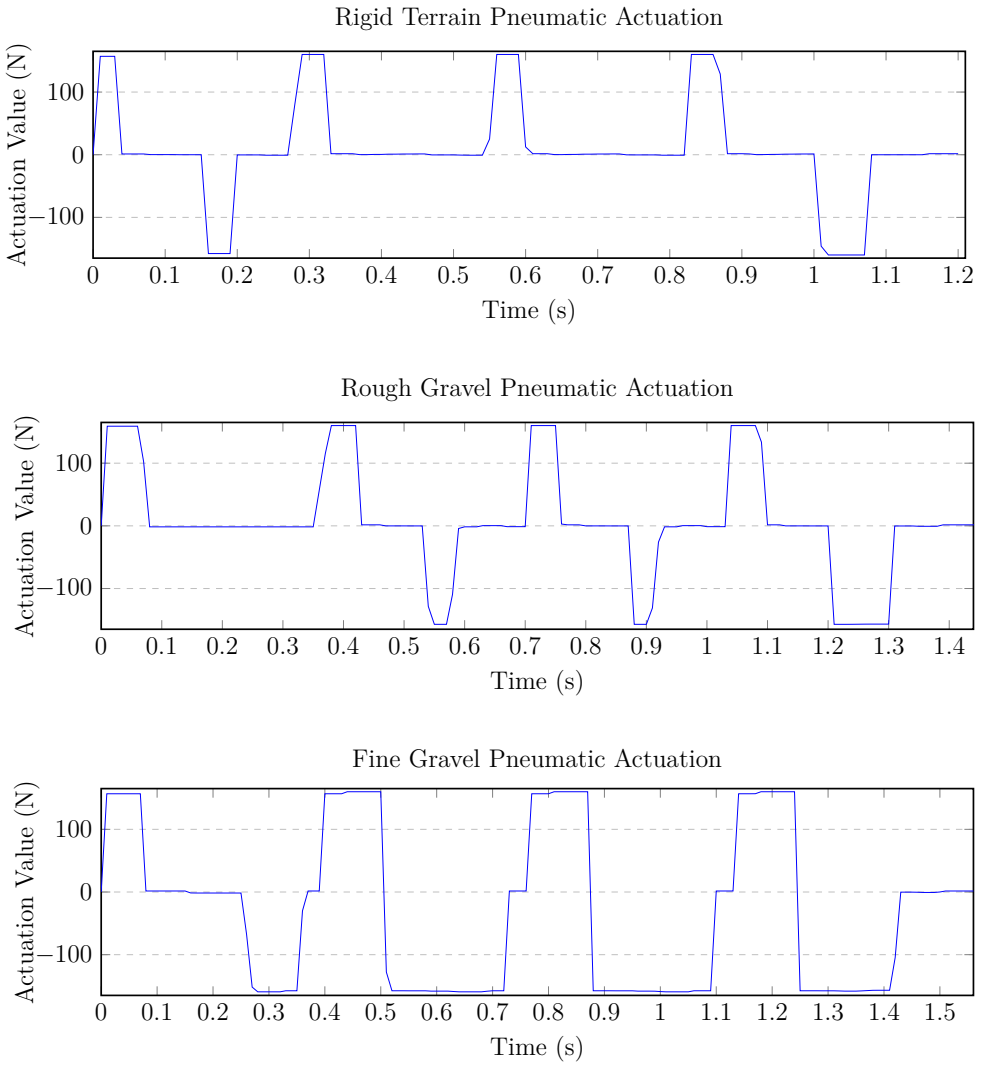

Figure 5.1: The interpolated and stitched long-time-horizon pneumatic actuation data generated by the optimizer for rigid terrain, rough gravel and fine gravel trajectories.

5.2.1. Pneumatic Actuation

The graphs in Figure 5.1 show the stitched and interpolated actuation data of the pneumatic cylinder for the rigid terrain, rough gravel and fine gravel. From these graphs we can see that the implementation of the pneumatic actuator as a bang-bang force with damping works (Section 3.1.3). The actuation force is either close to 160 N, 0 N or -160 N. The graphs also show that the node bucketing technique works. There is never rapid switching smaller than 20 ms and that accounts for the switching delay in the solenoid valves (Section 3.2).

An additional insight gained from these graphs is the ability of the optimizer to adjust the timing and time duration of applying a pneumatic actuation force when the ground contact becomes softer. On the rough gravel the optimizer uses the pneumatic contraction

to add more speed to the foot when the pneumatic expansion happens. This can be seen on the graph as the negative spikes. On the fine gravel the optimizer applies the forces for longer to adjust for lost energy in the contact. Table 5.4 shows the percentage of contraction and expansion time within a steady-state hop for the different terrain types.

Table 5.4: Percentage time the pneumatic actuator contracts and expands within a steady-state hop.

Steady-State Pneumatic Actuation		
Terrain Type	%-Contract	%-Expand
Rigid Surface	0	15.3%
Rough Gravel	15.15%	18.18%
Fine Gravel	61.1%	27.8%

This also shows how the optimizer uses contraction and subsequent expansion to add speed to the foot as it penetrates the compliant terrain. On the rigid terrain the optimizer frequently applies no force since it makes it easier to solve the hard stop complementarity constraints.

5.2.2. Foot Position

The graphs in Figure 5.2 show the effects of modelling the different terrain types. The graphs display the z-position movement of the foot as it makes contact with the ground during the steady-state trajectory. The first graph shows the hard contact steady-state trajectory and the second shows the fine gravel steady-state trajectory.

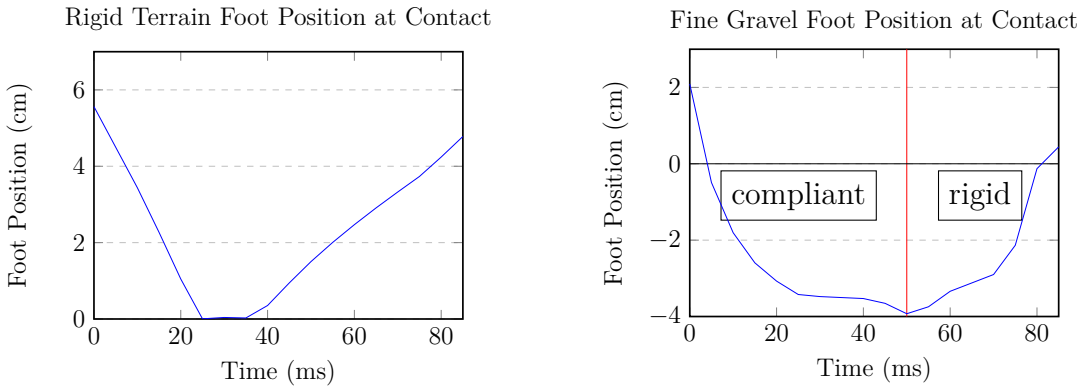

Figure 5.2: Foot position data generated by the optimizer for rigid terrain and fine gravel trajectories.

These graphs clearly show the different effects the different terrain types have on the foot contact. The rigid terrain graph shows the foot moving towards the ground linearly. Once it hits the ground it comes to a complete stop indicating a hard stop. The fine gravel graph shows the foot hitting the start of the ground at $z_{foot} = 0$ and slowing down due to

the compliant terrain effects. Once the foot reaches $z_{foot} = -3.92\ cm$ the terrain switches to hard contact dynamics and the foot can launch off the ground. This indicates that our compliant terrain model changes the dynamics of the foot and ground interaction. The feasibility of trajectory indicates that the drop tests done in Section 3.2.1 produced accurate penetration depths.

5.3. Trajectory Comparison Results

Since our preliminary robot platform was a fixed body robot, the risk of failing was low. The mono-pod had no non-actuated joints and all the movement came from the actuation of the pneumatic cylinder and servo motor. Due to the simplicity of the robot, this allowed for a basic open-loop control strategy.

Since the trajectory interpolation generates trajectory data at 5 ms intervals this data was used to send actuation instructions at 200 Hz. The pneumatic actuation data seen in Figure 5.1 and the positional data of the hip actuation was loaded onto a micro controller. The micro controller executed instructions according to this data at 5 ms intervals. No feedback was used to adjust the instructions. The following section will show the results gathered using this open-loop control method.

To compare the simulated trajectory and the robot execution the x-position and z-position of the robot body was measured by placing the optical encoders at points on the rig mentioned in Chapter 4 Section 4.3.2. The pitch and yaw indicated in Figure 4.4 is measured by the encoders, and from these measurements the x-position and z-position are derived.

5.3.1. Hard Surface

The trajectories generated using the trajectory setup in Section 5.1 were tested on the robotic platform. Initially a rigid terrain trajectory was executed on the robotic platform running on a hard surface. The x-position and z-position of the robot body was compared to confirm that the robot executes conventional trajectories correctly. After this the rigid terrain trajectory was executed on the robot platform running on the different gravel surfaces.

Figure 5.3 shows a z- and x-position comparison between the trajectory and the mono-pod robot platform execution of the trajectory using open-loop control. These results indicate that the robot executed the trajectory successfully on a hard surface without the use of

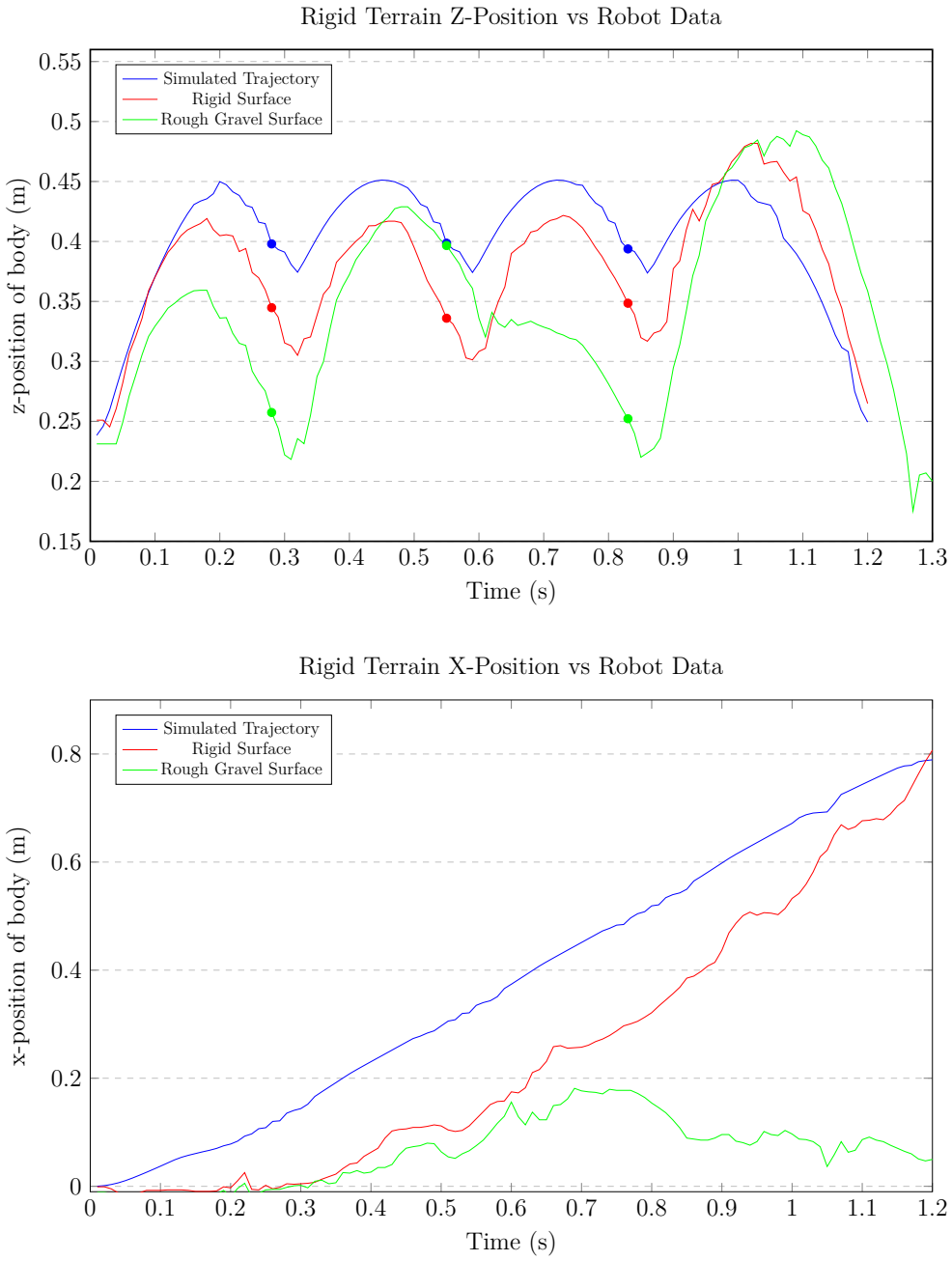

Figure 5.3: z- and x-position comparison between the simulated rigid terrain trajectory, the robot on a rigid terrain, and the robot on a compliant terrain. The dots on the graphs show when the pneumatic cylinder expanded.

a closed-loop controller. The same rigid terrain trajectory was executed on the rough gravel unsuccessfully. This was expected since the trajectory was generated for rigid terrain. The graphs reveal that the rigid terrain trajectory running on a rough gravel surface did not actuate at the right time to adjust to the softer terrain and therefore had inconsistent hopping heights. Due to these inconsistent hopping heights the foot placement was incorrect as well and the robot failed to hop the simulated distance. Next, to confirm whether the rough and fine gravel trajectories can be executed successfully without the

use of a closed-loop controller, the robot platform was tested on rough and fine gravel.

5.3.2. Gravel

Using the results of the drop test, rough and fine gravel trajectories were generated. The trajectories were executed on the robot running on rough and fine gravel respectively, and the x-position and z-position of the robot body was compared to the simulated trajectories. Figure 5.4 and 5.5 show the comparison between the simulated trajectories and the mono-pod robot platform executing the trajectories using open-loop control.

Since the implementation on the robot platform was an open-loop implementation with no feedback, the cylinder actuated at the same time on the platform as it did on the simulated trajectory. This made it impossible to adapt to terrain inconsistencies and lead to behaviour that can be seen in the graphs in Figures 5.4 and 5.5. The initial acceleration hops were too low and too high on the rough gravel and fine gravel respectively. Therefore the subsequent steady-state hops struggled to reach the desired heights. The robot platform still executed a run from rest back to rest successfully but could never recover from the initial inconsistencies. In Chapter 6 the implementation of a controller made it possible to adjust to these heights.

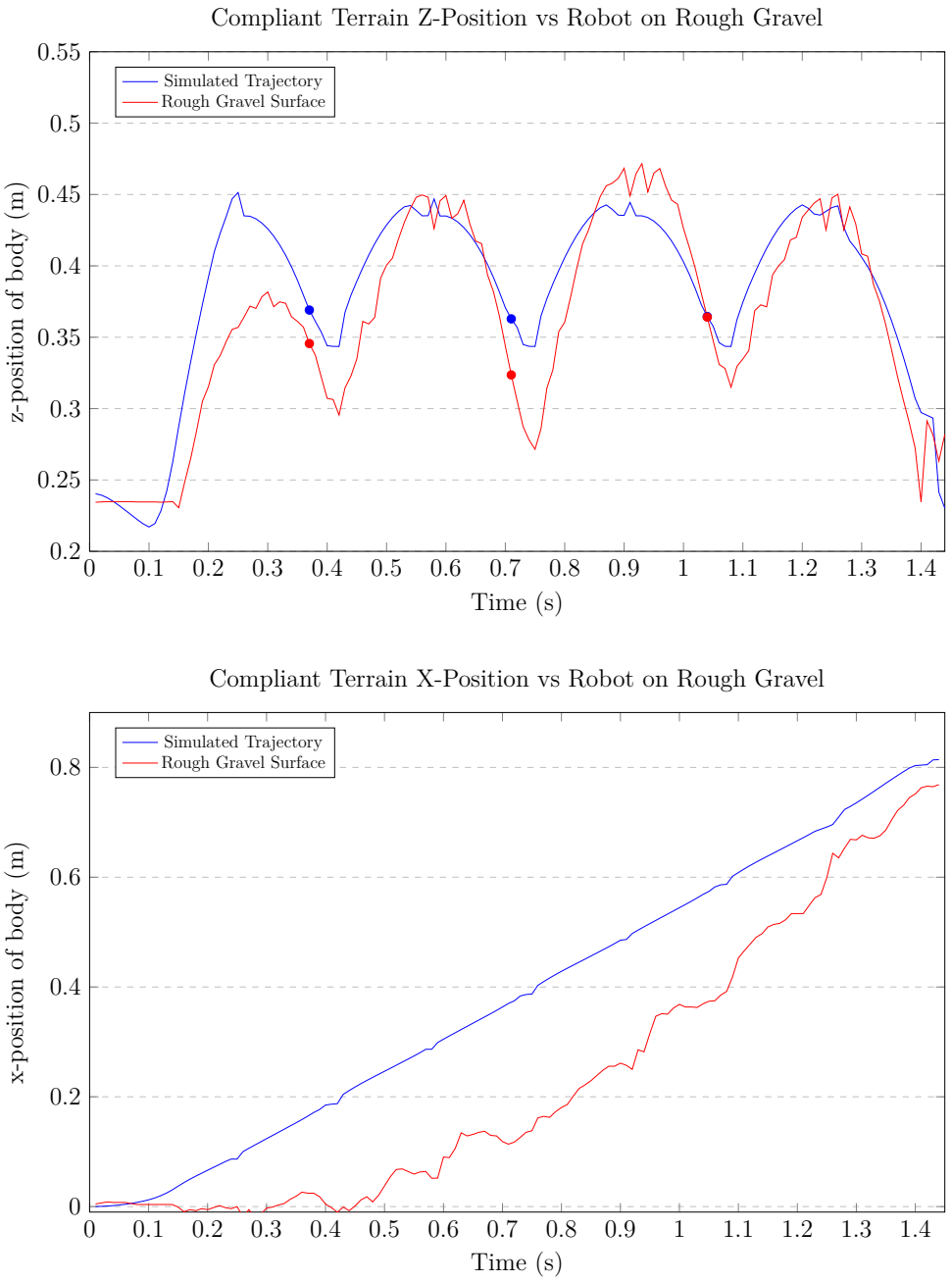

Figure 5.4: z- and x-position comparison between the simulated compliant terrain trajectory and the robot on rough gravel. The dots on the graphs show when the pneumatic cylinder expanded.

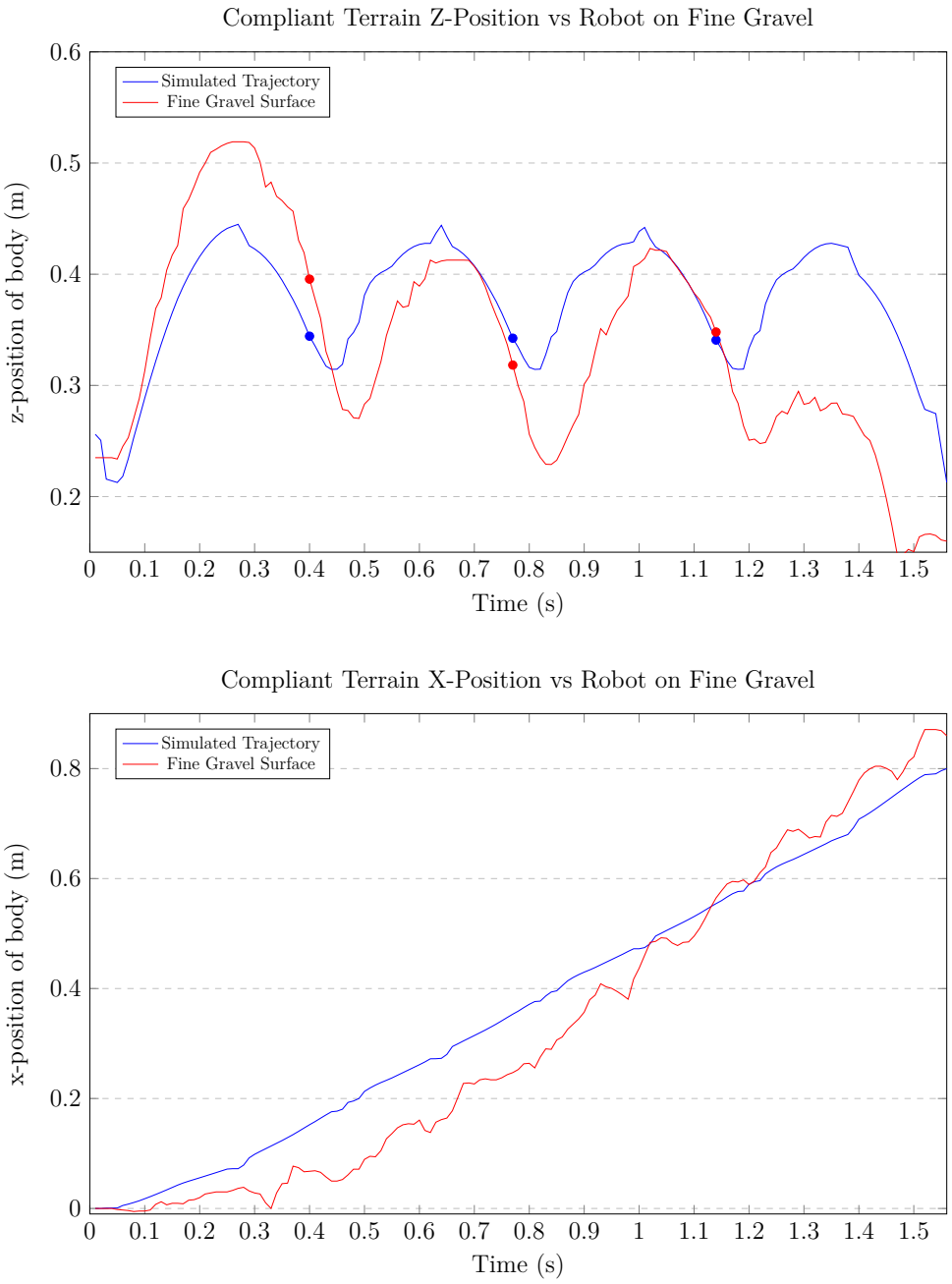

Figure 5.5: z- and x-position comparison between the simulated compliant terrain trajectory and the robot on fine gravel. The dots on the graphs show when the pneumatic cylinder expanded.

5.4. Results Discussion

The goal of building a fixed body robot and generating fixed body trajectories was to confirm whether techniques developed for this thesis would lead to accurate and realistic trajectories that can be implemented on a robot platform. It also confirms that the gravel and hard contact models, as well as the pneumatic model implemented in trajectory optimization works well enough to generated sufficient trajectories.

To confirm the generated trajectories are realistic and accurate the actuation data of the trajectory were executed on the robot platform and the results compared to the generated trajectory. This showed that the generated trajectories represent the dynamics of the robot platform well. Additionally it confirmed the success of the method describing the different compliant terrains.

One important aspect of the pneumatic actuation to note when looking at Figure 5.1 and the graphs in Figures 5.3, 5.4 and 5.5 is that the hard surface trajectory never contracted the cylinder during its steady-state hopping, the foot always touched the ground unactuated. In comparison the gravel surface trajectories contracted the cylinder in the air and expanded the cylinder before the foot touched the ground. This behaviour can be explained as a compensation for the terrain difference. The compliance of the gravel surfaces required the foot to hit the ground at a higher speed. The higher speeds caused a higher initial normal force acting on the foot and consequently higher friction between the foot and the ground reducing slippage. This acutation strategy was not pre-planned and was automatically generated by the optimizer to compensate for the compliant terrain model.

The following chapter, Chapter 6, implements the control methods described in Section 3.3 to allow a free body robot to execute hopping on different terrain types.

Chapter 6

Free Body

Generating and executing fixed body trajectories in Chapter 5 confirmed the pneumatic actuator model, the different terrain models and the general feasibility of the mono-pod trajectories. Future work will apply the techniques developed in this thesis to multi-legged robot that move without the assistance of a support rig. It was therefore necessary to confirm the techniques on a free body robot.

The added complexity of a free moving body led to the implementation of closed-loop control. Due to the assumptions and simplifications in modelling the actuators and terrain, the body movement of the robot could be different from the movement of the generated trajectory. Using feedback control allows the robot to adjust to these differences and still attempt to execute the movements successfully.

This chapter will firstly discuss the main changes in the optimizer model setup. Secondly some trajectory results will be used to inspire the free body and pneumatic control, and thirdly the control will be executed on the robot on different terrain types and the implementation will be compared to the generated trajectories. Lastly the control will be adjusted to execute on two surfaces within one execution and the results will be compared to the generated trajectories.

6.1. Optimizer Model Setup

To allow the model to move with a free body, a few changes need to be made to the model setup. To avoid repetition only the changes will be outline in this section. Section 5.1 covers the optimizer model setup needed for the fixed body setup.

When the equations of motion are set up for the fixed body model, the generalized coordinates discussed in Section 3.1.1 do not include the angle of the body, θ_{body}. The addition of the angle of the body allows the body to move freely around a central axis.

Since the body can move freely it will possess certain inertial forces that act on the whole robot system. These inertial forces are incorporated into the model when the body angle, θ_{body}, is added to the equations of motion. Once these changes were made to the model, and feasible trajectories were generated for the free body robot, these trajectories could be used to assist in setting up the controllers of the robot for the different trajectories. The body angle is constrained to start and end horizontally to ensure stability during hopping.

6.2. Pneumatic Control

The method described in Section 3.3.1 was used to implement pneumatic control. Using the generated trajectories, such as the one shown in Figure 6.1, a z-position trigger height where the solenoid valve should be triggered to expand the pneumatic cylinder, was identified. The trajectory optimization mostly switched from compression to expansion, with nodes applying no force to the pneumatic cylinder only when the foot is in the air. Therefore to simplify the control the controller was designed to expand the pneumatic cylinder once the robot is below the z-position trigger height and contract the pneumatic cylinder once the robot is above the z-position trigger height. This removes the ability to have no force applied to the pneumatic cylinder.

When the optimizer applied no force to the pneumatic cylinder, it was mostly done to ease the hard stop constraints on the cylinder. The ability to apply no force to the cylinder was therefore seen as unnecessary for successful trajectory execution. On the robot platform no additional energy is used to keep the cylinder expanded or contracted.

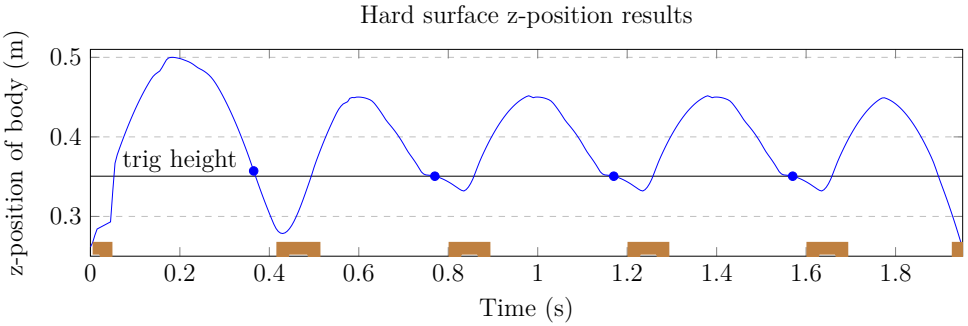

Figure 6.1: z-Position simulated trajectory with blue dots where the cylinder will go from contraction to expansion. The intermittent brown lines at the bottom of the graph indicate when the foot of the robot touches the ground.

The graphs in Figure 6.2 show how the height of the robot was used to control the pneumatic cylinder for the rigid terrain. The node in the trajectory 20 ms before pneumatic expansion takes place was to account for the delay in solenoid switching. The robot body height at

this node was identified and used in the controller as a trigger point to start expanding or contracting the pneumatic cylinder. Table 6.1 show the different trigger point heights for the different terrain types.

Table 6.1: Trigger height for pneumatic controller on different terrain types.

Control Height Trigger	
Terrain Type	z-Position Trigger $[cm]$
Rigid Terrain	34.6
Rough Gravel	34.9
Fine Gravel	36.2

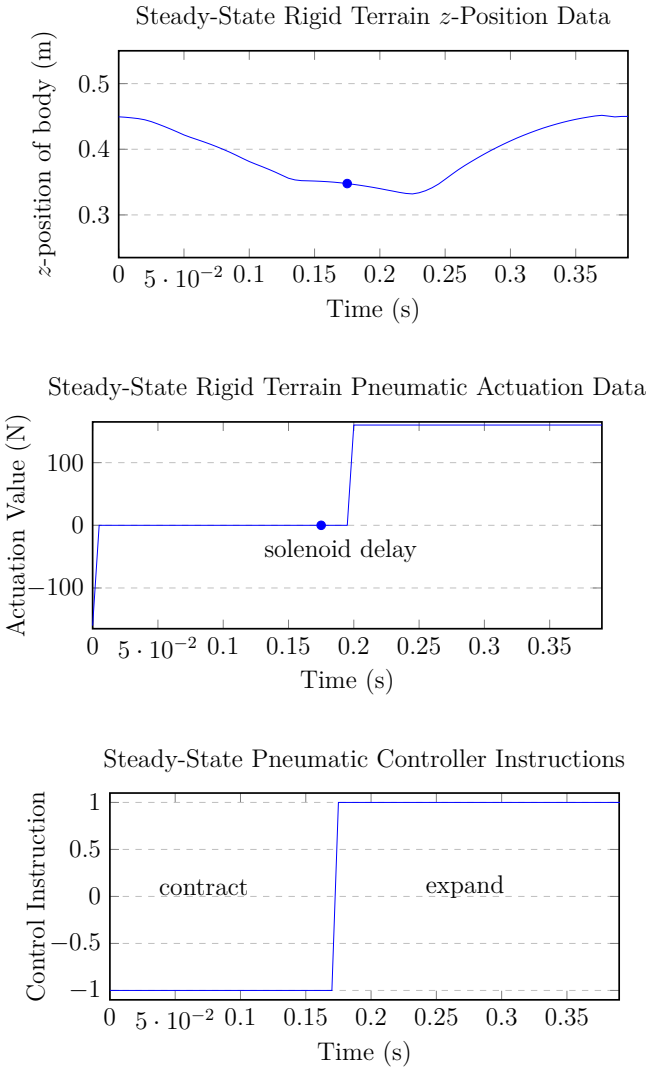

Figure 6.2: A rigid terrain steady-steady trajectory with a blue dot on the node where the solenoid needs to be triggered for pneumatic actuation.

As seen in Figure 6.1 the difference in firing height between the acceleration and steady-state trajectories was almost negligible. Therefore to simplify the controller even further, the difference in pneumatic actuation between acceleration, deceleration and steady-state

trajectories was ignored. The z-position trigger height was taken from the steady-state trajectories for each terrain type. During the execution of the experiments it was concluded that the simplification of the trigger heights was acceptable.

6.3. Servo Motor Control

As shown in Section 3.3.2 the feedback received from the encoder measuring the body angle of the robot was used to adjust the leg angle. This ensured that the leg points in the right direction relative to the x-axis and will allow the robot to execute the hopping accurately even if the body angle changes unexpectedly due to external disturbances.

The graphs in Figure 6.3 shows the change in leg angle of acceleration, steady-state and deceleration trajectories on a rigid terrain. The red part of the graph indicates that the foot of the robot is in the air, while the blue part of the graph indicates when the foot is on the ground. From these graphs the launch and touchdown angles for the different trajectory types can be found.

Table 6.2: Trigger height and specified leg angle for servo motor controller.

Rigid Terrain State Machine				
State	Touchdown Angle $[rad]$	T-Trigger Height $[cm]$	Launch Angle $[rad]$	L-Trigger Height $[cm]$
1 (ACC)	–	–	1.571	–
2 (ACC)	1.74	49.3	1.91	42.9
3, 4 (SS)	1.79	39.1	1.6	36.9
5 (DEC)	1.5	44.8	–	–

The graphs in Figure 6.3 also shows how the leg angle moves back and forth from a maximum to a minimum. This insight allowed simplification of the controller. Instead of following the exact angle at a node, a state machine was implemented that used the z-position of the robot once again as a trigger to move from different touchdown angles to different launch angles. Table 6.2 show the different launch and touchdown angles for the different states for the rigid terrain trajectory. To allow for thorough testing of the controllers, three steady-state hops were incorporated into the state machine. Figure 6.4 visualizes the state machine that was implemented.

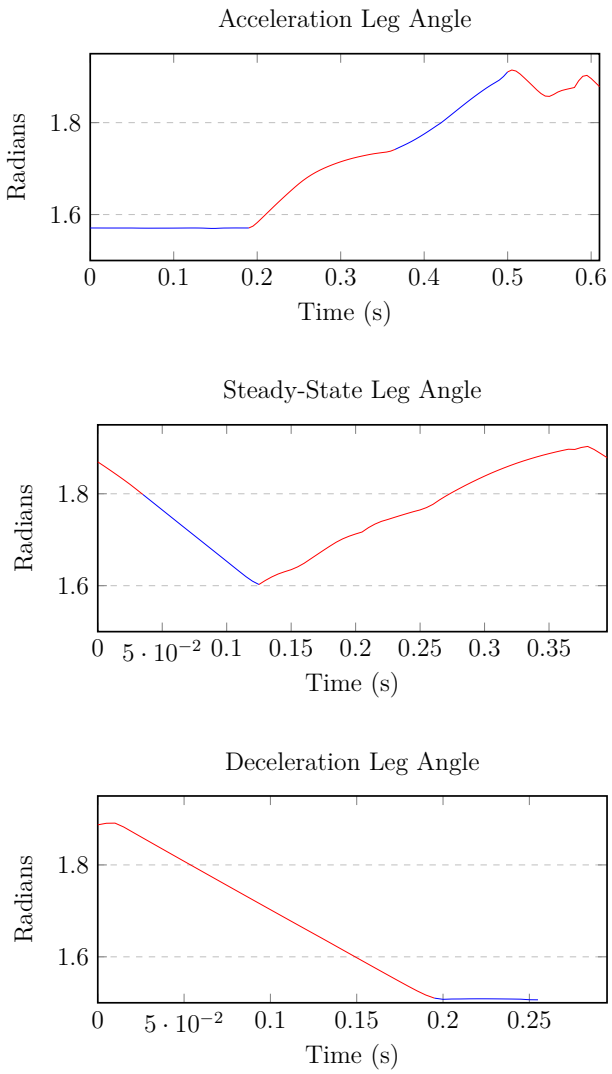

Figure 6.3: The change in leg angle for an acceleration, steady-state and deceleration trajectories on rigid terrain. The red indicates that the foot is in the air, and blue indicates that the foot is on the ground.

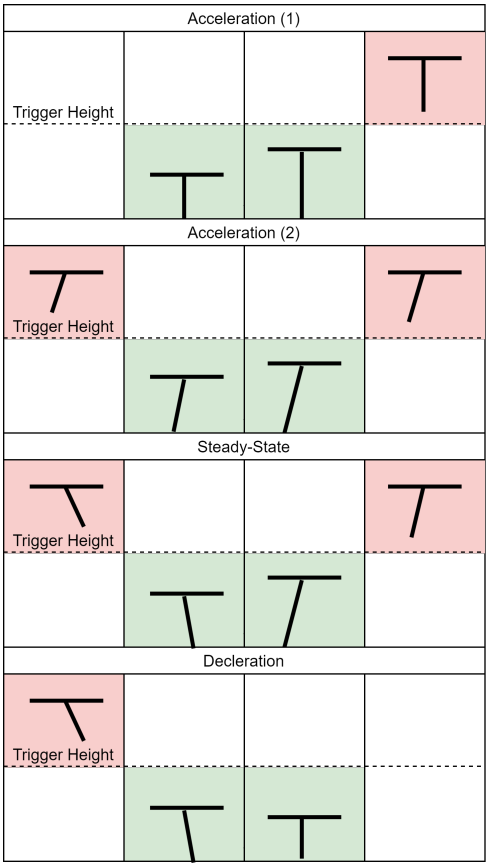

Figure 6.4: Diagram showing what the different states will look like. The green blocks indicate when the pneumatic cylinder should expand and the red blocks indicate when the pneumatic cylinder should contract.

6.4. Trajectory Comparison Results

The trajectory inspired controller was implemented on the mono-pod robot platform. Long-time-horizon trajectories were executed on the robot platform, with the robot starting in the rest configuration, accelerating to a steady-state speed and decelerating back to a rest configuration. Initially the controller was set up for a hard surface and Section 6.2 and 6.3 shows the different trigger heights and leg angles for the different terrains that were acquired from the trajectories. The x-position and z-position of the robot body was compared to confirm that the controller executes the trajectories correctly.

6.4.1. Hard Surface

The graphs in Figure 6.5 shows a comparison of the z- and x-position between the trajectory and the trajectory inspired controller on the free body mono-pod robot. The controller was set up for a hard surface. The results show some discrepancy between the trajectory and the controller execution, particularly on the z-position graph at the bottom of the steady-state hops. This is due to the way the pneumatic controller was simplified to either expand or contract.

When looking at where the pneumatic cylinder converts from contracting to expanding and comparing it to where the cylinder fires on the fixed body robot platform in Chapter 5, it becomes apparent that the controller changes the firing timing as the robot platform reaches different apex heights. This is different from Chapter 5 where the timing of the cylinder firing is fixed making it fire at the incorrect height. This is all due to the controller that was implemented and it allowed the free body robot platform to adjust to inconsistencies in the compliant surfaces in the next section. Overall the results show a successful execution of the trajectory by the controller.

6.4.2. Gravel Terrain

After confirmation that the controller can execute a trajectory on the hard surface, the controller was set up for the two different types of gravel. Once again Section 6.2 and 6.3 shows the different trigger heights and leg angles for the different terrains. The x-position and z-position of the robot body was compared on both types of gravel to confirm that the controller executes the trajectory correctly.

The graphs in Figure 6.6 shows a comparison of the z- and x-position between the trajectory

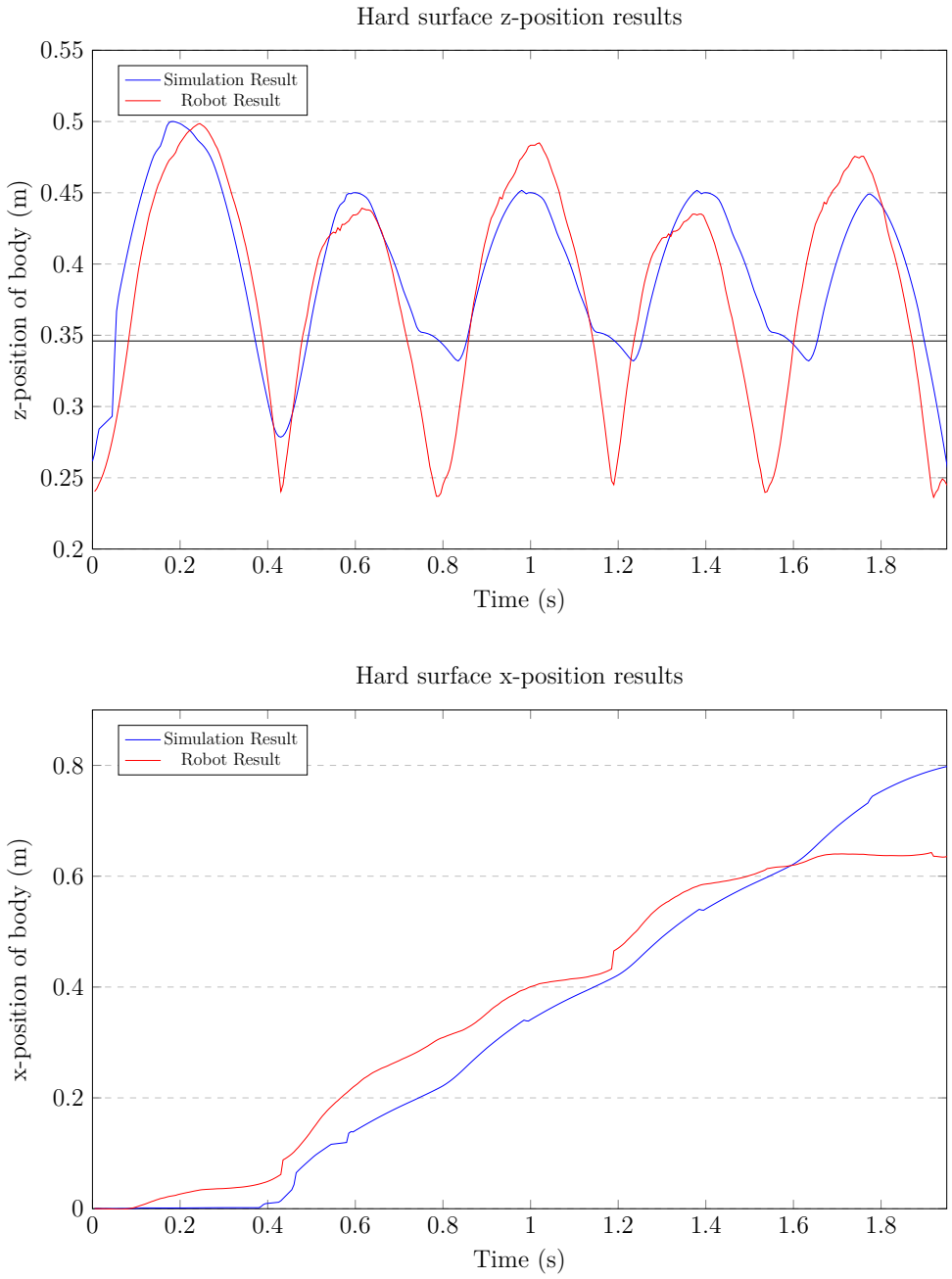

Figure 6.5: z- and x-Position comparison between the hard surface trajectory and the free body robot hopping on a hard surface. The line on the z-Position graph indicate the trigger height for when the pneumatic cylinder goes from contraction to expansion.

and the trajectory inspired controller on the free body mono-pod robot. The controller was set up to execute on the rough gravel surface. The results show some discrepancy between the trajectory and the controller execution, particularly the apex height and x-position velocity of the mono-pod robot. This is most likely due to the inconsistencies of the rough gravel that was not accounted for in the model, but the controller assisted in adjusting to these inconsistencies by firing at a desired height and not at a desired time as in Chapter 5. Overall the results show a successful execution of the trajectory by the controller.

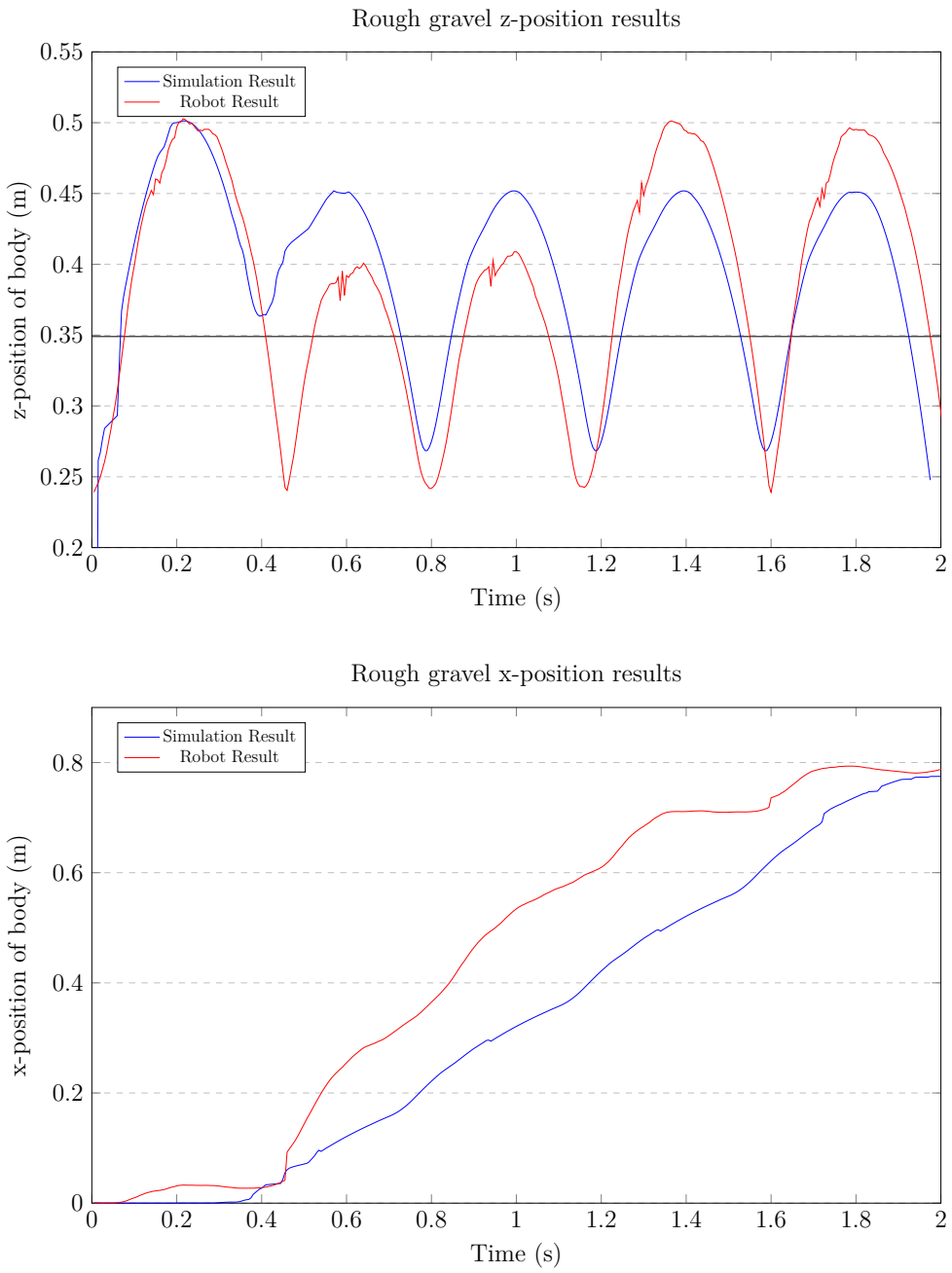

Figure 6.6: z- and x-Position comparison between the rough gravel trajectory and the free body robot hopping on rough gravel. The line on the z-Position graph indicate the trigger height for when the pneumatic cylinder goes from contraction to expansion.

The graphs in Figure 6.7 shows a comparison of the z- and x-position between the trajectory and the trajectory inspired controller on the free body mono-pod robot. The controller was set up to execute on the fine gravel surface. Once again due to inconsistencies in the fine gravel the results show some discrepancy between the trajectory and the controller execution, particularly the apex height and x-position velocity of the mono-pod robot. Overall the results show a successful execution of the trajectory by the controller. The pneumatic actuation indications show how the pneumatic controller used the trigger

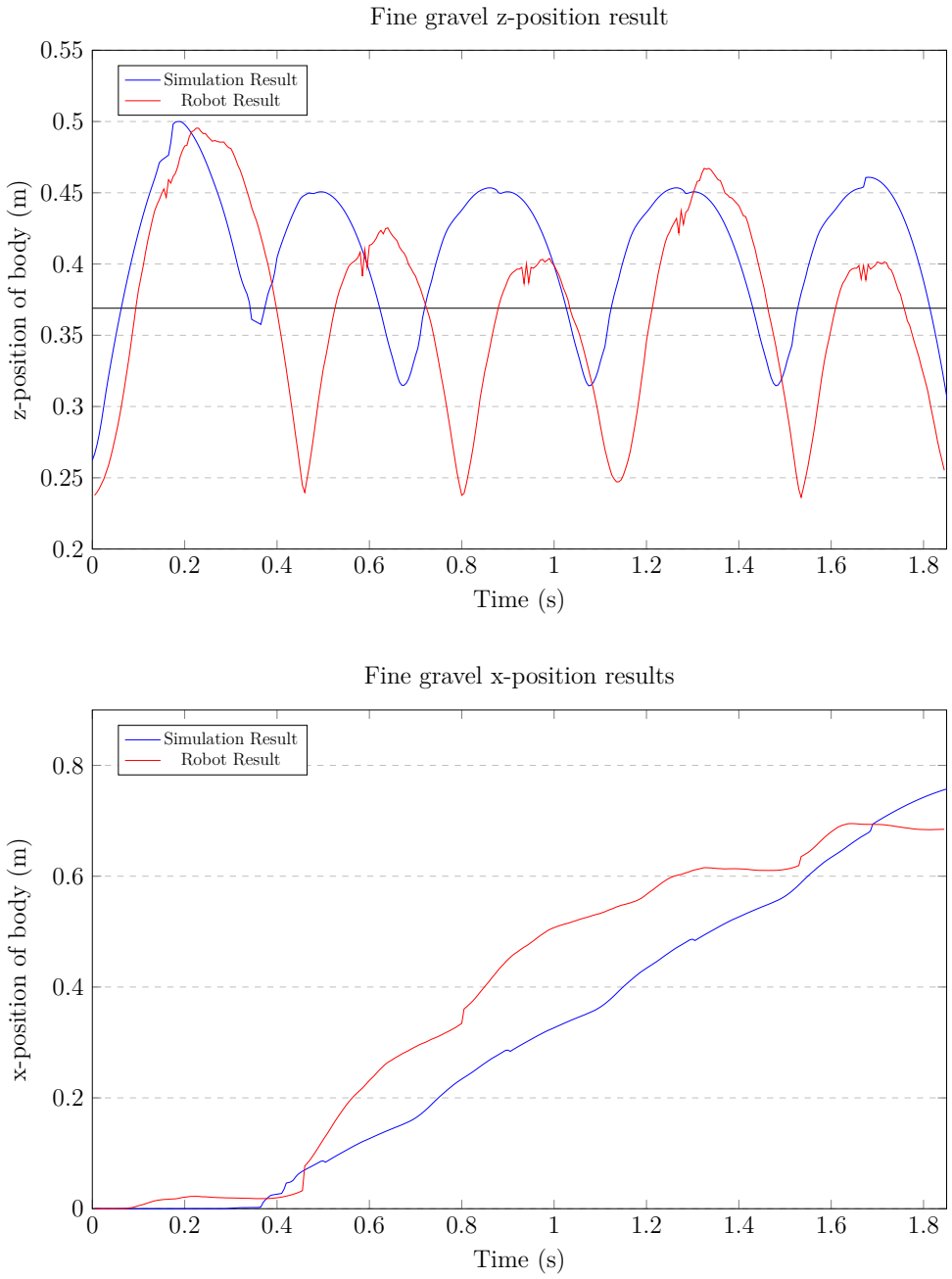

Figure 6.7: z- and x-Position comparison between the fine gravel trajectory and the free body robot hopping on fine gravel. The line on the z-Position graph indicate the trigger height for when the pneumatic cylinder goes from contraction to expansion.

heights from the simulated trajectory to expand the pneumatic cylinder at the right time regardless of the inconsistencies of the terrain and the possible external disturbances.

The successful execution of the long-time-horizon trajectories of the free body robot also indicates that the servo controller kept the free body in a stable horizontal position throughout the execution. Even if the free body rotated to a more unstable position due to external disturbances the servo controller adjusted to this rotation and still executed

the trajectory successfully in most cases.

From the images in Figure 6.8 the expansion of the cylinder before the foot touches the ground is shown. This is once again, as discussed in Chapter 5, to increase the traction of the foot on the ground by digging into the surface.

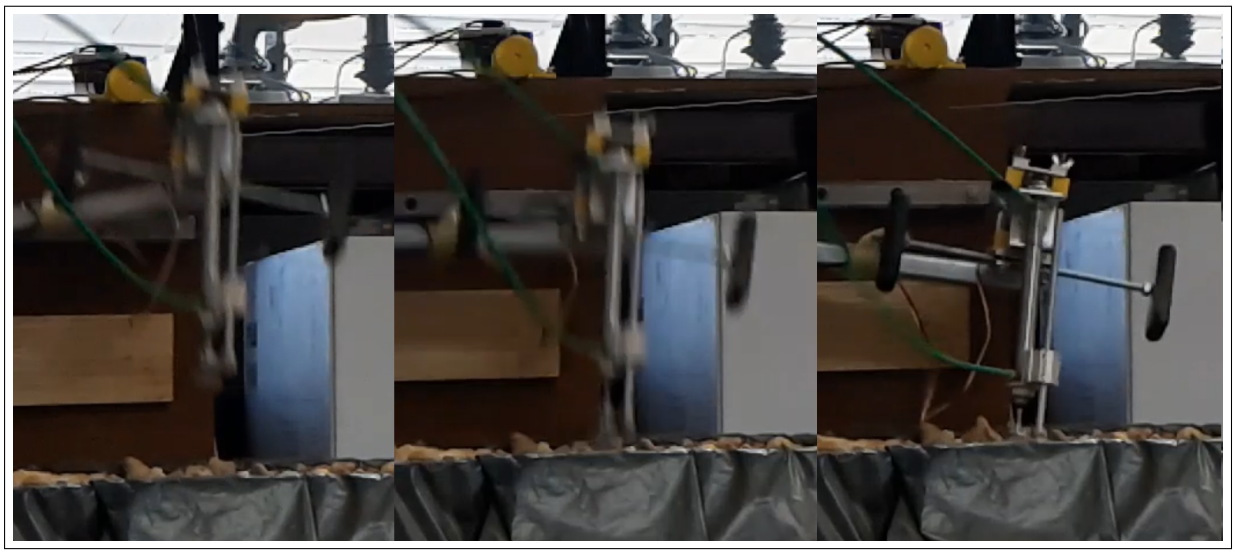

Figure 6.8: Screenshots of the robot showing the pneumatic expansion before the foot touches the ground.

6.4.3. Multi-Surface Execution

The results in Figure 6.5, 6.6 and 6.7 show that the controller implementation can successfully execute multiple mono-pod hops from rest back to rest without complete operations failure. This is a good result and shows the methods that were implemented in trajectory optimization as well as controller design helped with the adjustment to different terrain types. But what happens if the mono-pod robot needs to execute hops on two different terrain types in one execution?

Since all the trajectories for the free-body robot was constrained to an apex height of 45 *cm*, the controller could be adjusted to switch from one surface type to another halfway through execution. The trigger heights and leg angles were changed to the appropriate values at the appropriate state in the state machine.

Two experiments were set up to test the viability of the multi-surface execution. Firstly the controller was set up to start on a hard surface and transition to a rough gravel surface. Secondly the controller was set up to start on a hard surface again and transition to a fine gravel surface. The floor of the robot platform was adjusted to be a hard surface for the first 40 *cm* and a gravel surface for the second 40 *cm*.

The results in Figures 6.9 and 6.10 compare the z- and x-position of the trajectories to the controller execution of the free body mono-pod robot. The trajectories are stitched together, and not re-optimized for this specific experiment.

The graphs in Figures 6.9 and 6.10 show successful controller execution of multi-surface hopping.

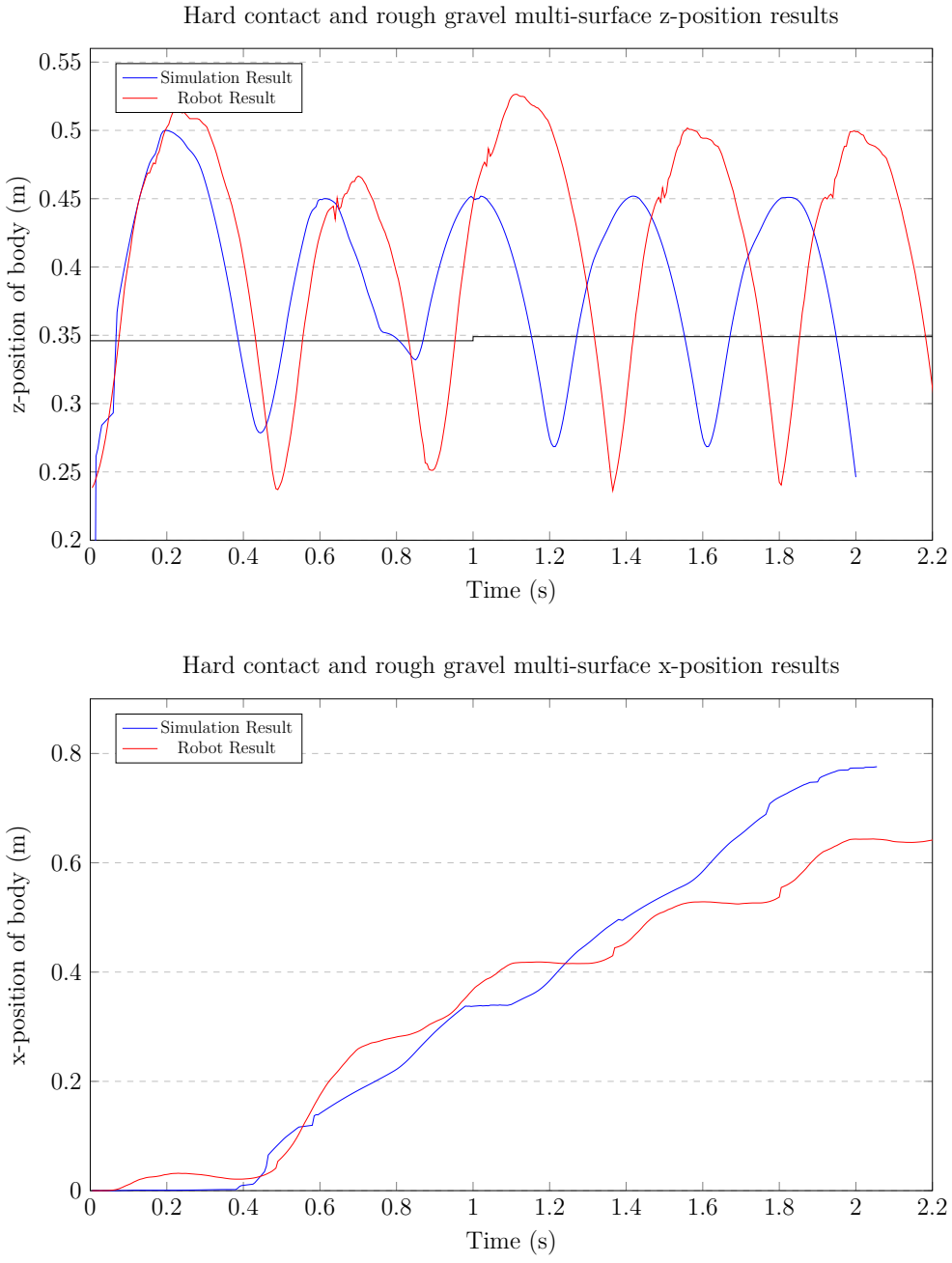

Figure 6.9: z- and x-position comparison between the simulated multi-surface trajectory and the robot hopping from a hard surface to rough gravel. The line on the z-Position graph indicate the trigger height for when the pneumatic cylinder goes from contraction to expansion.

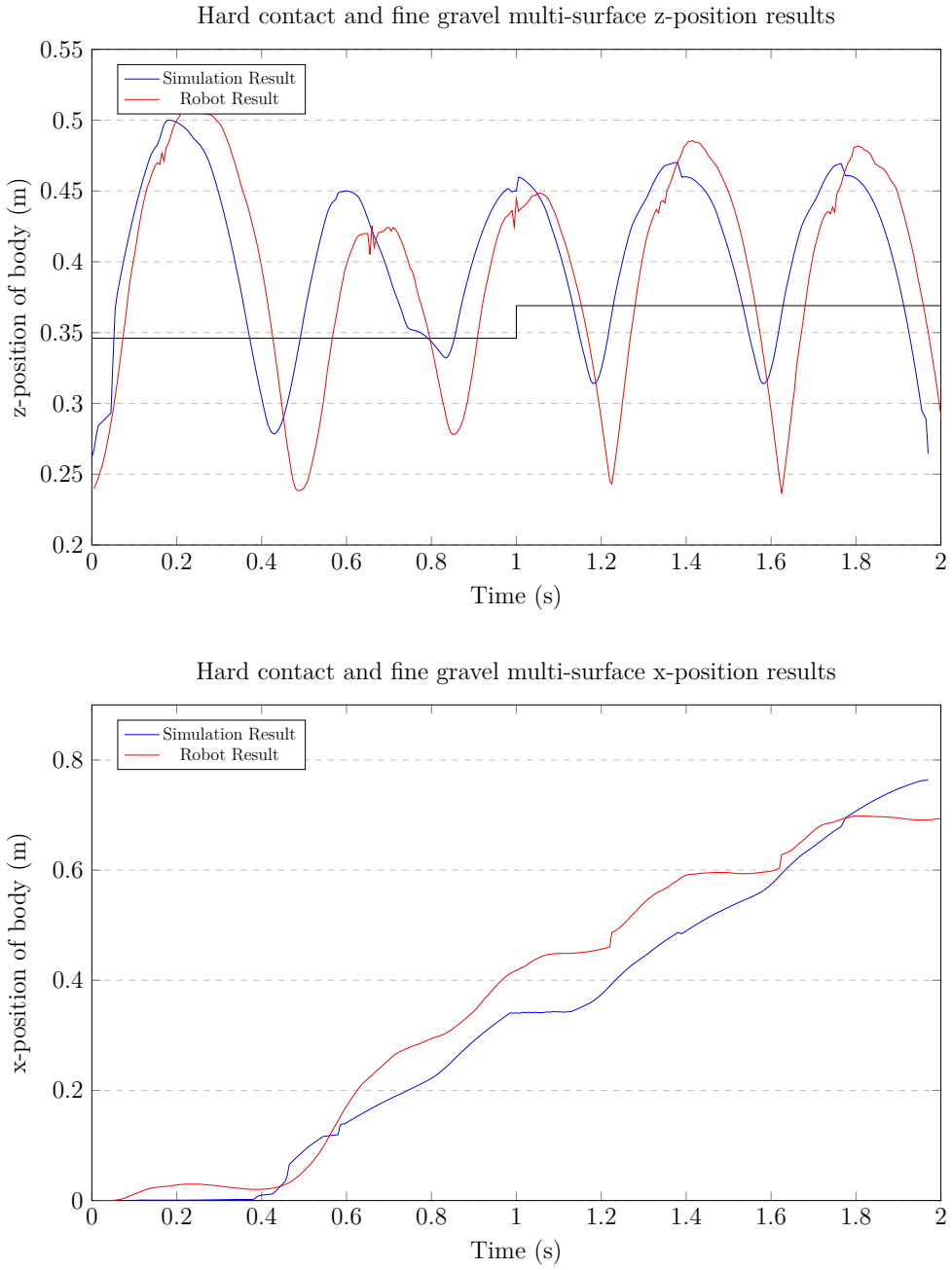

Figure 6.10: z- and x-position comparison between the simulated multi-surface trajectory and the robot hopping from a hard surface to fine gravel. The line on the z-Position graph indicate the trigger height for when the pneumatic cylinder goes from contraction to expansion.

Chapter 7

Conclusion

Legged locomotion in robotics research is a growing field in the literature. Quadrupeds like Boston Dynamics' Spot [5] is proving the commercial viability of legged robots, while ETH Zurich's AnyMAL [25] is pushing the limits of legged robotics research. But many research groups still struggle to create a robotic platform that is capable of working robustly outside of the laboratory environment. To achieve success outside the laboratory environment, a legged robot has to be able to deal with different terrain types and obstacles.

The aim of this research was to use trajectory optimization to inspire basic control design for a mono-pod robot on different terrain types. A model of the the mono-pod robot was created and used in the trajectory optimization to find feasible trajectories that could be executed on a physical robot platform that was designed and built during this thesis. For a rigid terrain type through-contact methods were used to model the interaction between the foot and the ground. This allowed the optimizer to determine the optimal contact order. For a compliant terrain type a hybrid method was designed that used a prescribed contact order as well as through-contact methods. Additionally rigid terrain trajectories were used to inspire the placement of the prescribed contact order in the hybrid method. This allowed for specific changes in dynamics at the contact point between the foot and the ground, while taking the guessing work out of the prescription of contact order. From the different trajectories a controller was designed and implemented on a robot platform.

To confirm whether the trajectories were feasible in the real world a mono-pod robot was designed and built. Additionally a support structure for the mono-pod was built that ensured safe sagittal movement and collected appropriate metrics for comparison to the generated trajectories. An initial boom arm based robot platform was built, but was replaced by a linear rail and shorter boom arm platform as the robot moved from a fixed body to a free body.

Trajectories were generated for three different terrain types, a hard surface, a rough gravel surface and a fine gravel surface. Initially the trajectories were generated for a fixed body robot, and open-loop control was implemented on the fixed body robot platform to execute

the trajectories. These initial experiments confirmed the methods used to describe the different terrain types as well as several other methods explained in Chapter 3.

The trajectories generated for the free body robot platform was used to inspire controller design. Specific z-position values were used to control pneumatic actuation and servo motor actuation. These z-position values were taken from the generated trajectories. For each terrain type the z-position and leg angle values were different. Additionally the servo motor used feedback from the body angle measurement to adjust the leg angle and allow successful execution even if the body was disturbed. A state machine was used to move from an acceleration phase to a steady-state phase and a deceleration phase.

The controller was implemented on the free body robot platform using a micro controller and optical encoder sensors placed at specific positions on the robot platform to allow for the necessary feedback. The robot platform executed the control, and the z- and x-position of the robot was recorded and compared to the generated trajectories. The results showed that the controller executed the trends of the trajectories in a satisfactory way. The adjustments made by the controller due to disturbances in the free body insured successful execution, and the changes in the z-position and leg angle values adjusted appropriately to the different terrain types.

Additional experiments were done to see whether the controller can handle multi-surface hopping. The state machine was adjusted to hop on two surfaces within one execution. A hard surface was placed underneath the robot platform for the first half of the execution distance, and a gravel surface was placed underneath the robot platform for the second half of the execution distance. Half way through execution the z-position and leg angle values changed to accommodate the change in terrain type. These executions were compared to a stitched compliant terrain and rigid terrain trajectory and showed that the controller followed the trends of the stitched trajectory in a satisfactory way. These results opens up the ability to implement a gait library for multiple terrain types. The results confirmed that the use of trajectory optimization to inspire basic control design for different terrain types is a valid technique to use in legged locomotion.

Below is a link to a video showing the robot executing all the different trajectories on the different surfaces.

- https://drive.google.com/file/d/1Fipy6e4fUYPWdV464g-7PXTSvXpus8zK/view?usp=sharing

7.1. Future Work

Recommended future work include expanding this technique to bipeds and quadrupeds. Using sensory feedback to obtain terrain properties and adjust the controller accordingly can be looked at. This will allow dynamic switching from one gait to another without a predefined state machine. Finally, implementing the hybrid method to describe compliant terrain can be tested on more terrain types to confirm its viability.

Bibliography

[1] C. M. Hubicki, J. J. Aguilar, D. I. Goldman, and A. D. Ames, "Tractable terrain-aware motion planning on granular media: An impulsive jumping study," pp. 3887–3892, 2016.

[2] S. Meek, J. Kim, and M. Anderson, "Stability of a trotting quadruped robot with passive, underactuated legs," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 347 – 351, 06 2008.

[3] M. Buehler, "Dynamic locomotion with one, four and six-legged robots," *Journal of the Robotics Society of Japan*, vol. 20, no. 3, pp. 237–242, 2002.

[4] M. H. Raibert, "Hopping in legged systems — modeling and simulation for the two-dimensional one-legged case," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-14, no. 3, pp. 451–463, 1984.

[5] E. Guizzo and E. Ackerman, "$74,500 will fetch you a spot: For the price of a luxury car, you can now buy a very smart, very capable, very yellow robot dog," *IEEE Spectrum*, vol. 57, no. 8, pp. 11–11, 2020.

[6] T. Kamimura, S. Aoi, K. Tsuchiya, and F. Matsuno, "Body flexibility effects on foot loading in quadruped bounding based on a simple analytical model," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2830–2837, 2018.

[7] V. K. J. Carius, R. Ranftl and M. Hutter, "Trajectory optimization with implicit hard contacts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3316–3323, 2018.

[8] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact *," 2013. [Online]. Available: https://groups.csail.mit.edu/robotics-center/public_papers/Posa13.pdf

[9] C. Fisher, "Trajectory optimisation inspired design for legged robotics," Ph.D. dissertation, University of Cape Town, 3 2021, an optional note.

[10] K. Harbick and G. S. Sukhatme, "Controlling hopping height of a pneumatic monopod," 2002.

[11] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[12] R. Blickhan and R. Full, "Similarity in multilegged locomotion: Bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, pp. 509–517, 01 1993.

[13] J. Tenreiro Machado and M. Silva, "An overview of legged robots," 04 2006.

[14] I. Doroftei and I. ION, *Design and locomotion modes of a small wheel-legged robot*, 07 2013, pp. 609–616.

[15] R. MOSHER, "Test and evaluation of a versatile walking truck," *Proceedings of Off-Road Mobility Research Symposium, Washington DC, 1968*, pp. 359–379, 1968. [Online]. Available: https://ci.nii.ac.jp/naid/10029513508/en/

[16] R. McGhee and A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, 1968. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0025556468900904

[17] M. Raibert, M. Chepponis, and H. Brown, "Running on four legs as though they were one," *IEEE Journal on Robotics and Automation*, vol. 2, no. 2, pp. 70–82, 1986.

[18] J. Hodgins, "Legged robots on rough terrain: experiments in adjusting step length," pp. 824–826 vol.2, 1988.

[19] M. Raibert, M. Chepponis, and H. Brown, "Experiments in balance with a 3d one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[20] R. Playter and M. Raibert, "Control of a biped somersault in 3d," vol. 1, pp. 582–589, 1992.

[21] A. B. Filho, P. F. S. Amaral, and B. G. M. Pinto, "A four legged walking robot with obstacle overcoming capabilities," pp. 374–379, 2010.

[22] Y. Chow and R. CHung, "Obstacle avoidance of legged robot without 3d reconstruction of the surroundings," vol. 3, pp. 2316–2321 vol.3, 2000.

[23] M. Yagi and V. Lumelsky, "Biped robot locomotion in scenes with unknown obstacles," vol. 1, pp. 375–380 vol.1, 1999.

[24] Y.-J. Li, W.-C. Chou, C.-Y. Chen, B.-Y. Shih, L.-T. Chen, and P.-Y. Chung, "The development on obstacle avoidance design for a humanoid robot based on four ultrasonic sensors for the learning behavior and performance," pp. 376–379, 2010.

[25] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," pp. 38–44, 2016.

[26] E. Guizzo, "By leaps and bounds: An exclusive look at how boston dynamics is redefining robot agility," *IEEE Spectrum*, vol. 56, no. 12, pp. 34–39, 2019.

[27] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6295–6301.

[28] R. Colbrunn, G. Nelson, and R. Quinn, "Design and control of a robotic leg with braided pneumatic actuators," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 2, 2001, pp. 992–998 vol.2.

[29] D. Robinson, J. Pratt, D. Paluska, and G. Pratt, "Series elastic actuator development for a biomimetic walking robot," in *1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (Cat. No.99TH8399)*, 1999, pp. 561–568.

[30] C.-P. Chou and B. Hannaford, "Measurement and modeling of mckibben pneumatic artificial muscles," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 90–102, 1996.

[31] G. Kenneally, A. De, and D. E. Koditschek, "Design principles for a family of direct-drive legged robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, 2016.

[32] E. Guizzo, "A new spin on robot actuators," *IEEE Spectrum*, vol. 56, no. 1, pp. 40–42, 2019.

[33] D. J. Blackman, J. V. Nicholson, C. Ordonez, B. D. Miller, and J. E. Clark, "Gait development on Minitaur, a direct drive quadrupedal robot," in *Unmanned Systems Technology XVIII*, R. E. Karlsen, D. W. Gage, C. M. Shoemaker, and G. R. Gerhart, Eds., vol. 9837, International Society for Optics and Photonics. SPIE, 2016, pp. 141 – 155. [Online]. Available: https://doi.org/10.1117/12.2231105

[34] A. De and D. E. Koditschek, "The penn jerboa: A platform for exploring parallel composition of templates," *CoRR*, vol. abs/1502.05347, 2015. [Online]. Available: http://arxiv.org/abs/1502.05347

[35] J. W. Hurst, "The role and implementation of compliance in legged locomotion," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, August 2008.

[36] A. Wang and S. Kim, "Directional efficiency in geared transmissions: Characterization of backdrivability towards improved proprioceptive control," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1055–1062.

[37] M. Nordin, J. Galic', and P.-O. Gutman, "New models for backlash and gear play," *International Journal of Adaptive Control and Signal Processing*, vol. 11, no. 1, pp. 49–63.

[38] N. Kau, A. Schultz, N. Ferrante, and P. Slade, "Stanford doggo: An open-source, quasi-direct-drive quadruped," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6309–6315.

[39] S. Yu, T.-H. Huang, X. Yang, C. Jiao, J. Yang, Y. Chen, J. Yi, and H. Su, "Quasi-direct drive actuation for a lightweight hip exoskeleton with high backdrivability and high bandwidth," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1794–1802, 2020.

[40] G. Pratt and M. Williamson, "Series elastic actuators," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1, 1995, pp. 399–406 vol.1.

[41] M. Raibert, "Dynamic legged robots for rough terrain," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*, 2010, pp. 1–1.

[42] "What is a pneumatic actuator and how do they work?" https://www.processindustryforum.com/article/what-is-a-pneumatic-actuator#:~:text=By%20definition%2C%20a%20pneumatic%20actuator,compressed%20air%20into%20mechanical%20motion.&text=Consisting%20of%20a%20piston%2C%20cylinder,linear%20or%20rotary%20mechanical%20motions., (Accessed on 05/27/2021).

[43] M. Schluter and E. Perondi, "Mathematical modeling with friction of a scara robot driven by pneumatic semi-rotary actuators," *IEEE Latin America Transactions*, vol. 18, no. 06, pp. 1066–1076, 2020.

[44] G. Muscato and G. Spampinato, "A multi level control architecture for a pneumatic robotic leg," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, vol. 2, 2005, pp. 7 pp.–779.

[45] *solenoid valve CPE10-M1BH-5L-QS-6*, FESTO, 12 2018.

[46] C. Fisher, A. Blom, and A. Patel, "Baleka: A Bipedal Robot for Studying Rapid Maneuverability," *Frontiers in Mechanical Engineering*, vol. 6, p. 54, 2020.

[47] "Research," https://hashimoto-lab.jp/en/research-en/, (Accessed on 06/03/2021).

[48] G. S. Kanakis, F. Dimeas, G. A. Rovithakis, and Z. Doulgeri, "Prescribed contact establishment of a robot with a planar surface under position and stiffness uncertainties," *Robotics and Autonomous Systems*, vol. 104, pp. 99–108, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889017308266

[49] A. Patel, S. L. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-implicit trajectory optimization using orthogonal collocation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2242–2249, 2019.

[50] L. Drnach and Y. Zhao, "Robust trajectory optimization over uncertain terrain with stochastic complementarity," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1168–1175, 2021.

[51] Y. Aso, K. Aihara, and K. Ito, "Multi-legged robot for rough terrain: Shinayaka-l vi," in *2019 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2019, pp. 136–141.

[52] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986

[53] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2665–2670.

[54] T. Antony and M. J. Grant, "Rapid indirect trajectory optimization on highly parallel computing architectures," *Journal of Spacecraft and Rockets*, vol. 54, no. 5, pp. 1081–1091, 2017. [Online]. Available: https://doi.org/10.2514/1.A33755

[55] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[56] Y. Zhao, H. Lin, and M. Tomizuka, "Efficient trajectory optimization for robot motion planning," *CoRR*, vol. abs/1810.04255, 2018. [Online]. Available: http://arxiv.org/abs/1810.04255

[57] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory optimization for legged robots with slipping motions," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3013–3020, 2019.

[58] N. F. Steenkamp and A. Patel, "Minimum time sprinting from rest in a planar quadruped," pp. 3866–3871, 2016.

[59] J. V. Zyl, "Rapid acceleration of legged robots," 3 2021.

[60] W. Hart, J. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, pp. 219–260, 2011.

[61] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832)," Online, 5 2021.

[62] A. L. Schwartz, "Theory and implementation of numerical methods based on runge-kutta integration for solving optimal control problems," Ph.D. dissertation, EECS Department, University of California, Berkeley, Apr 1996. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1996/2995.html

[63] D. Pretorius and C. Fisher, "A novel method for computing the 3d friction cone using complimentary constraints," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5000–5006.

[64] C. Fisher, C. Hubicki, and A. Patel, "Do intermediate gaits matter when rapidly accelerating?" *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3418–3424, 2019.

[65] P. Edwards, "Mass-spring-damper systems: The theory," 2001.

[66] S. Seok, A. Wang, D. Otten, and S. Kim, "Actuator design for high force proprioceptive control in fast legged locomotion," 10 2012.

[67] N. Weiss, "Developing simple pneumatic actuator models for robotic applications," 2020.