Universidade do Minho
Escola de Engenharia

Diogo Alexandre Pires da Silva

Prevision, control and optimization of a hexapod robot posture in inclined surfaces

October 2023

Universidade do Minho
Escola de Engenharia

Diogo Alexandre Pires da Silva

Prevision, control and optimization of a hexapod robot posture in inclined surfaces

Dissertação de Mestrado
Mestrado Integrado em Engenharia Mecânica

Trabalho efetuado sob a orientação do:
Professor Doutor João Paulo Flores Fernandes
Professor Doutor Pedro Filipe Lima Marques

October 2023

ACKNOWLEDGEMENTS

For all the people that helped, directly and indirectly, the development of this dissertation I have many thanks, but some of them deserve a special recognition and acknowledgement.

First, I would like to thank Professor Filipe Marques and Joana Coelho, without their support, this dissertation would not have been possible. Both of them demonstrated patience and a willingness to assist me. Through their guidance, I acquired effective work methodologies and rectified numerous errors in my work. A single thanks is not enough to acknowledge all the support they both gave.

To my dissertation advisor, Professor Paulo Flores, a deep gratitude for all the knowledge he shared with me during my university years.

To Professor Gil and Professor Fernando Ribeiro, although they were not directly involved with my project, they were always eager to help.

To Manuela and Manuel, for giving me the most basic name in the Portuguese language, inspiring me to work on my personality. I would not be who I am, nor would I strive to reach for the stars if it wasn't for your untiring guidance and love. Pedro, Rui, Luís and Marco, the only people in the world I love to hate, thank you for being part of my life. Also, to Generosa and Maria, my favourite century-old woman, for always supporting me.

To everyone at LAR, thank you for taking care of a minority of mechanical engineering students and making them feel at home. To Tiago, and everyone involved in all the LAR projects I can only express my gratitude for giving me an amazing last year of university. To Marco and Remelgado, it is an honour to call you friends.

To everyone that I encountered during my years in the course of Mechanical Engineering, especially the ones from the group "Cães do Monte", I am eternally grateful for the countless thanks I owe you. The memories, the challenges, the laughter, the moments that forged deep connections, and the experiences that contributed to my personal growth - I will forever cherish these in my heart.

To those who have earned my unwavering love and respect and left a permanent mark my soul— Chapter, Toucas, Massas, Mafalda, Caixas, Cristiano, Cordeiro, Rento, Primata, Paiva, Diana, Magda, André, Eduardo, Cedric, Roleira, Rui, Bárbara, Cachetas, Fella, Lexa e Diogo— you each deserve a special mention for keeping my hope alive in discovering the beauty of life.

STATEMENT OF INTEGRITY

ABSTRACT

Prevision, control and optimization of a hexapod robot posture in inclined surfaces

In a world marked by natural and man-made disasters, the imperative of deploying mobile autonomous robots to replace human involvement in hazardous environments is evident. With this in mind, this dissertation focuses on developing posture control techniques that allow the robot to safely navigate these environments.

Among legged robots, hexapod robots distinguish themselves as exceptional performers. Their capabilities extend to climbing, functioning with damaged limbs, and exhibiting remarkable static balance and gait movement. To comprehend the significance of hexapod robots in contrast to other legged counterparts, an extensive analysis is conducted, studying the structural attributes of hexapods, such as body composition, leg and joint arrangements, actuator and sensor configurations, thereby exposing the advantages and disadvantages intrinsic to this type of robots.

The groundwork for this research is firmly established as it delves into the realm of posture adjustment. In pursuit of enhanced adaptability for the hexapod robot across various terrains, five leg path algorithms were compared, namely: triangular function, parabola function, 3rd-degree spline function and 3rd and 4th-degree Bézier curves. Compared along four different environments the preferred choice for this purpose is the 3rd-degree Bézier curve algorithm. This exploration, focusing on posture adjustment, provides a foundation for the understanding of the ATHENA hexapod model, encompassing its kinematic principles and gait generation strategies.

The application of Q-Learning aided with integration of proprioceptive and exteroceptive sensors and simulation frameworks form a robust foundation for the posture adjustment problem. Through simulations with diverse control parameters in different slope environments, optimal control parameters for each slope were identified. These findings were then applied to simulate the robot navigating terrain with various slopes. A simulation lacking height control parameters resulted in failure, while the controlled simulations successfully adapted to variable slopes.

Keywords

"ATHENA"; HEIGHT CONTROL; HEXAPOD ROBOT; POSTURE ADJUSTMENT; Q-LEARNING

RESUMO

Previsão, controlo e otimização da postura de um robô hexápode em superfícies inclinadas

Num mundo marcado por desastres naturais e provocados pelo homem, é evidenciado a necessidade de implementar robôs móveis autónomos para substituir intervenções humanas em ambientes perigosos. Com isso em mente, esta dissertação foca-se em desenvolver técnicas de controlo de postura que permitem o robô navegar com segurança nestes ambientes.

Entre os robôs com pernas, os robôs hexápodes destacam-se excecionalmente. Estes robôs têm um equilíbrio estático notável e são capazes de escalar e operar mesmo com pernas danificadas. Para compreender a importância dos robôs hexápodes em contraste com outros tipos de robôs com pernas, é realizada uma pesquisa extensiva, explorando as características estruturais dos hexápodes, como a estrutura corporal, a disposição das pernas, configuração das articulações, atuadores e sensores, revelando assim as vantagens e desvantagens inerentes a este tipo de robôs.

O alicerce desta pesquisa é estabelecido à medida que nos aprofundamos no domínio do ajuste de postura. Em busca de uma maior adaptabilidade para o robô hexápode em vários terrenos, foram comparados cinco algoritmos de trajetória de pernas, nomeadamente: função triangular, função parabólica, função *spline* de 3º grau e curvas de *Bézier* de 3º e 4º grau. Comparando-os em quatro ambientes diferentes, a curva de *Bézier* de 3ºgrau foi a selecionada. Esta pesquisa, centrada no ajuste de postura, fornece uma base para a nossa compreensão do modelo hexápode *ATHENA*, abrangendo a sua análise cinemática e estratégias de geração de marcha.

A aplicação de *Q-Learning*, com a integração de sensores exterocetivos e propriocetivos e um esquema de códigos e simulações, constitui uma base sólida para o problema de ajuste de postura. Através de simulações com diversos parâmetros de controlo em ambientes compostos por diferentes inclinações, identifica-se os parâmetros de controlo ótimos para cada inclinação. Estas descobertas foram então aplicadas para simular a navegação do robô num terreno com vários declives. Uma simulação sem aplicação de parâmetros de controlo de altura resultou em fracasso, enquanto as simulações controladas adaptaram-se com sucesso a inclinações variáveis.

Palavras-Chave

AJUSTE DE POSTURA; "*ATHENA*"; CONTROLO DE ALTURA ; *Q-LEARNING;* ROBÔ HEXÁPODE

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

Abbreviations

ATHENA	All-Terrain Hexapod for Environment Navigation Adaptability
AI	Artificial Intelligence
ANN	Artificial neural network
ARL (LAR)	Automation and Robotics Laboratory
CMEMS	Center for Microelectromechanical Systems
CoM	Center of Mass
CoT	Cost of Transport
CPG	Central Pattern Generator
CTr	Coxa-Trochaterofemur
DGSM	Dynamic gait stability margin
DoF	Degrees of Freedom
DP	Dynamic programming
FTi	Femur-Tibia
GUI	Graphical user interface
IMU	Inertial measurement unit
LSM	Longitudinal Stability Margin
MDP	Markov Decision Process
ML	Machine Learning
RL	Reinforcement Learning
RRT	Rapidly exploring random tree
SLIP	Spring-loaded inverted pendulum
SP	Support Polygon
SP	Supervised Learning
SSM	Static Stability Margin
TC	Thorax-Coxa
TD	Temporal difference
UL	Unsupervised Learning
ZMP	Zero Moment Point

Symbols

θ	Angle	rad
ζ	Cost of transport	–
γ	Discount factor	–
β	Duty factor	–
ε	Explore rate	–
k_h	Height control parameter	–
δ_z	Height displacement	m
α	Learning rate	–
m	Mass	kg
T	Phase time	s
k_β	Pitch control parameter	–
δ_β	Pitch displacement	m
P	Power consumption	W
r	Reward	–
k_α	Roll control parameter	–
δ_α	Roll displacement	rad
S	Stride	m
f	Stride frequency	Hz
λ	Stride length	m
t	Time	s
h	Trajectory maximum height	m
v	Velocity	m/s

1. Introduction

Natural and man-made disasters prompt the necessity for mobile autonomous robots capable of substituting human-performed chores in unsafe environments. The performance of mobile and autonomous robots creates the possibility of safeguarding human life by eliminating the use of humans to access dangerous environments. Therefore, these devices are applied to operate, detect and execute rescue maneuvers(Zhao et al., 2018). Regarding the access to these environments, only half of the land area is accessible by wheel-type vehicles. Wheeled robots need a continuous support point between the wheels and the ground during locomotion, which affects their performance in irregular surfaces. On the contrary, multi-legged biomimetic robots have isolated support points, which optimize support and traction. Thus, the discrete contact points provide, legged robots superiority in terms of mobility. Legged robots are able to perform tasks in complex environments by choosing the best approach to overcome the obstacles or ground irregularities, depending on their movements (Raibert, 1986; Wang et al., 2021; Zhao et al., 2018). Amongst legged robots, hexapod robots are capable of climbing, moving with damaged limbs and have the best overall static balance and gait, which is of paramount importance for navigating complex environments(Deng et al., 2017a; H. Li et al., 2021; H. Zhang et al., 2014).

The gait complexity of hexapod robots is not only related with the mechanical parts, but also the servomotors and electrical components that enable their control (Faigl & Čížek, 2019). The body's posture and gait constantly change while moving through rough terrain because of balance requirements imposed by the interaction between the ground and the contact points. Subsequently, the generation of the leg's trajectory change continuously to adapt to the hexapod's balance to the walking conditions(Bal, 2021; Deng et al., 2017a). Besides, robot stability is equally important to avoid falling and damage. Stability is considered a challenge in terms of locomotion control due to the fact that an ineffective posture control creates torso oscillations (Bjelonic et al., 2016). Since posture control is necessary in the robot's motion across inclined and complex surfaces, the dynamical analysis plays an important role in the control design. Development of the dynamical model requires the feet forces distributions, energy consumption, torque and angular momentum (Mahapatra et al., 2019). Directly reading IMU's (inertial measurement units) is a common method to control the posture of the robot, obtaining the joints values and triggering the limbs correctly.

Reinforcement Learning application by trial-and-error to predict and correct the robot’s posture using the current posture, its stability and the relative angular position of each joint.

It is intended that the actions provide the angular positions of the torso to establish the best orientation according to the terrain the robot is located. Simulations will be made using *Gazebo*. CMEMS dynamical model is necessary to apply the posture control.

Given the framework of the theme, the present dissertation is part of the project “ATHENA” (All-Terrain Hexapod for Environment Navigation Adaptability). Its main objective is to develop a smart hexapod robot, capable of moving in any terrain. This theme has the support and partnership of the Automation and Robotics Laboratory (LAR), department of Electronic Engineering of University of Minho.

All studies are directed to predict and control the robot’s posture according to its movement, either being in flat or rough ground. Results will be validated and obtain through experimental tests.

1.1. Objectives

The primary focus of this dissertation is to proficiently develop posture control techniques utilizing Reinforcement Learning to effectively adjust the height of a hexapod robot when navigating inclined surfaces. The robot's central task is to discern the degree of inclination on the terrain and accurately apply the most appropriate height adjustment control parameter for the specific slope it encounters.

To achieve the primary goal of this dissertation, the following tasks were assigned:

- Identify a leg movement algorithm that achieves a harmonious balance between minimizing energy consumption and ensuring the smoothness of limb trajectories;

This task aims to reduce oscillations in the system. A range of algorithms will be introduced and evaluated based on criteria such as Cost of Transport, height variations, as well as roll and pitch dynamics. Furthermore, a diverse set of simulation environments will be crafted to rigorously assess each algorithm across various scenarios, bolstering the reliability of the outcomes.

- Development of a Q-Learning algorithm and simulation scheme;

To ensure smooth simulations, this task was defined. It encompasses the creation of a comprehensive reward algorithm and a versatile code framework, created to facilitate adaptation and future research involving posture adjustment through the application of Reinforcement Learning.

- Q-Learning Simulations;

Simulations will be conducted using the refined reinforcement learning algorithm within an array of distinct simulation environments, each featuring unique slopes. This will shed light on the identification of the optimal height control parameters tailored to each specific incline.

- Determine optimal height control parameters and compare results;

The final task of this dissertation, which meets the overall purpose of the project, is to identify the optimal height control parameters for each specific slope. This will be based on the results of the Q-Learning simulations and their comparison with a simulation with no control parameters applied.

1.2. Dissertation Structure

This dissertation is divided into five chapters, which are organized as explained in the following paragraphs.

The first Chapter corresponds to the introduction, where the topic is introduced along with its importance so that the relevance of the dissertation is perceptible. The objectives, to be addressed in the final conclusions, are also outlined.

The second Chapter presents the literature review, which encompasses a thorough investigation into hexapod robots distinctive characteristics, the intricate control mechanisms governing their movements, and the strategic use of reinforcement learning, particularly concentrating on the Q-Learning algorithm's application in enhancing their adaptability and decision-making capabilities.

The third Chapter describes the virtual hexapod model used in this research, studying the kinematic model responsible for controlling limb motion and guiding the robot movement. A study is made on gait generation mechanisms, including stance and swing phases, and applicable leg path algorithms. It also presents the test environments and simulation results.

In the fourth Chapter, the posture adjustment is investigated, encompassing the role of sensors for data assessment, the mechanisms facilitating posture orientation determination and control parameter manipulation. The simulation framework employed for Q-Learning-based posture adjustment, the construction of reward algorithms within the reinforcement learning context, and the conduction of simulations in diverse slope environments were explored. Finally, a comprehensive comparison and conclusions drawn from the extensive analysis performed.

In the fifth Chapter, and last Chapter, the key conclusions of the dissertation are highlighted in light of the set objectives and testing program results. Perspectives and suggestions for future work are also discussed in order to underline the importance of the outcomes and work done while guiding for potential changes that can be adopted.

2. Literature Review

Robots are tools envisioned to perform tasks done traditionally by humans, whether in industrial surroundings or any other place that would benefit with automation. Autonomous robots offer a solution to access dangerous environments that could be a threat for human life. For detecting and rescue applications, these mobile robots become an optimal addition to maneuver rough terrains.

Legged robots are superior in navigating hazardous environments compared with their wheeled counterparts because they only need discrete footholds for walking and can adapt their gait while walking (Zhao et al., 2018). Due to their intrinsic static stability, which is the ability to maintain the body steady and upright when only reaction forces are applied to the system, insects have been examined as inspiration for robots that can navigate autonomously in complex situations. Not only that, but their design allows the generation of different gait patterns which increases their adaptability to the environment (Coelho et al., 2021).

Hexapod robots have the best overall static balance and gait in comparison to biped and quadruped robots. Additionally, hexapod robots are capable of climbing walls and carrying loads higher that its own weight. Although hexapod robots are superior in some traits they have some drawbacks, such as energy consumption, due to the superior number of motors in each leg. The mechanical complexity is also a bottleneck in the design of hexapods, since it influences each leg's number of Degrees of Freedom(DoF) (Lagaza & Pandey, 2018).

Artificial Intelligence (AI) is considered a recent subject but worthy topic of investigation in the control of mobile robots. Russell and Norvig (2009) describe AI as the art of creating machines that perform functions that require intelligence when performed by people. The concept of AI can be divided according to the learning methodology. Amongst the AI branches, Machine Learning (ML) aims at data learning and making predictions and/or decisions. Within ML, the Reinforcement Learning (RL) approach examines problems that require sequential decision making. Therefore, RL is usually related to optimal control and operations research and management (Y. Li, 2018).

This chapter exposes the theoretical concepts applied in this dissertation. Firstly, hexapod robots and their inherently characteristics and mechanisms. An extensive review on the main hexapod body structures, followed by an evaluation of the common legs and joints arrangements. The main sensors and actuators applied in modern-day hexapod robots, the first being what allows the robot to understand the physical world, as well as its internal state, and the latter being what gives power to make the legs move. Afterwards, a study on the control of hexapod robots, analysing the performance indices that evaluate and compare the performance of hexapod robots, the movement control and posture control that regulate the body's posture and leg movements. The main gaits utilized in recent research are exposed, in addition to the most common control architectures that are employed in robot systems.

Lastly, Reinforcement Learning concepts and applications are studied. An exploration of the general terminology used in RL problems to give a grasp on its core principles. Then, an in-depth study on the main learning approaches and the mathematical framework used to create, train, and evaluate RL agents. Additionally, the examination of the practical application of Q-Learning, which is a widely used RL algorithm, to solidify the understanding of these concepts.

2.1. Hexapod Robots

A hexapod robot comprises a main body with six legs attached. Due to the discrete landing points and the redundant DoF of the legs, the performance of hexapods is considered better than wheeled and tracked robots (Q. Liu & Jing, 2015). The overall capacities of hexapod motivate their study for walking across inclined surfaces, when compared to biped, quadruped and even octopod robots.

This type of legged robots can navigate irregular terrains due to their inherent static stability, which is the capacity for maintaining the body stable and upright when only the reaction forces are applied to the system. The distribution of the limbs around the main body, provides the generation of omnidirectional motion and fault tolerant locomotion (Tedeschi & Carbone, 2014).

2.1.1. Main body

The main body, or torso, of a hexapod robot can be classified between two different types of structures, a rectangular shaped structure or a hexagonal shaped structure. The first one (see Figure 2-1 (a)) is considered to be insect inspired and the six legs are distributed symmetrically in each side. The rectangular design provides a fast locomotion in the longitudinal and lateral directions. However, the hexapod becomes more rigid in terms of maneuverability, and, turning trajectories are more difficult to perform (Zaghloul et al., 2016). On the other side, hexagonal shaped hexapods (see Figure 2-1 (b)) have all six legs distributed symmetrically around the main body. The presented configuration can achieve the same walking speed in any given direction and its more flexible.

Rectangular hexapod robots are better suited than their counterpart in moving along a straight line, however they require a special gait for turning actions that consists of four steps.

Hexagonal hexapod robots have true radial symmetry, so the body has no front or rear and there is no preferential direction for the motion. Tedeschi and Carbone (2014) proved that hexagonal robots have superior stability margin, stride and turning ability compared to rectangular robots.

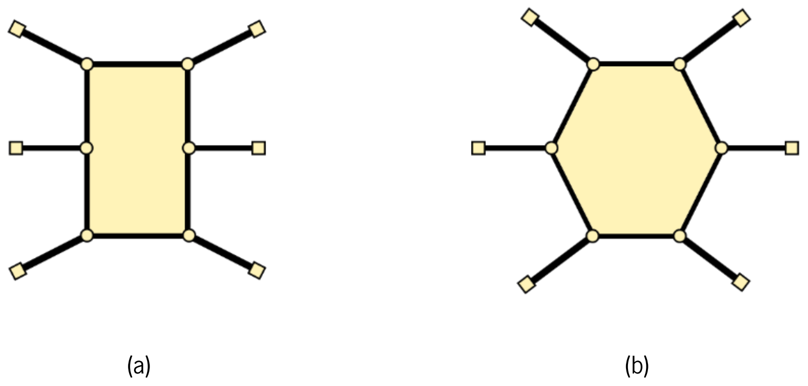


Figure 2-1 Leg Distribution in Hexapod robots: (a) Rectangular hexapods, (b) Hexagonal hexapods.

2.1.2. Robotic Legs

Artificial locomotion systems are mechanical structures with legs, each one comprising several links connected by prismatic or rotational joints. Gao et al., (2022) defined legged robots as multi-input, multi-output, multi-end-effector systems. These systems present advantages over conventional vehicles that use wheels. Nonetheless, legged robots reveal complex kinematic and dynamic settings, which make their analysis and control difficult (Silva & Tenreiro Machado, 2007).

Legged locomotion vehicles present superior mobility in natural terrains, due to the discrete foothold positions. The ability to use discrete footholds can improve energy consumption since they deform the terrain less than wheeled or tracked vehicles. Legged robots can include redundant legs which improve drastically the static balance and locomotion even with damaged legs (Silva & Tenreiro Machado, 2007). The mechanical design and kinematic chain of the limbs' mechanism favours the robot adaptation to the surface irregularities. The mechanism workspace allows an infinite combination of joints' positions, hence the straightforwardness in the adjustment of the feet coordinates. The disadvantages of designing leg mechanisms include their low speed, and complexity of the generation of control algorithms. Despite that, in some situations the energy consumption can be reduced to energetically optimize the limbs trajectories, as aforementioned in the previous subsections.

Figure 2-2 shows two different types of hexapod legs, bio inspired legs and non-zoomorphic legs.

Bio inspired legs configuration is influenced by animal gaits, such as reptiles, mammals or arachnid. Non-zoomorphic legs usually take inspiration in anthropomorphism designs. The main characteristics of each leg type are listed in Table 2-1.

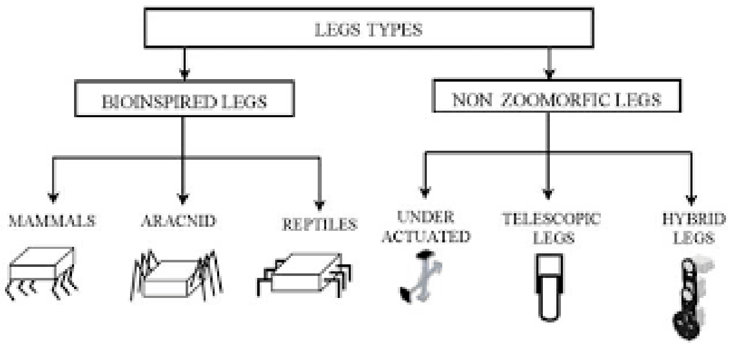


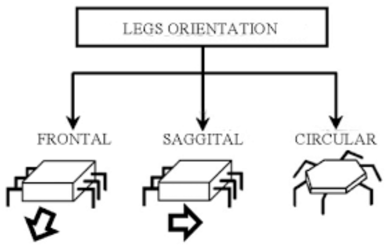
Figure 2-2 Hexapod leg types. (Tedeschi & Carbone, 2014)

Table 2-1 Hexapod leg types and characteristics.

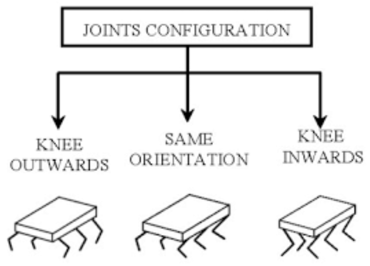
Leg type	**Principal characteristics**	**Reference**
Mammal	Lower power consumption to support the body since it is above the legs.	(Tedeschi & Carbone, 2014)
Aracnid	Legs extremities are situated on both sides, sticking the knees at the top of the body.	(Tedeschi & Carbone, 2014)
Reptil	Legs are placed on both ends and knees to the side of the base.	(Tedeschi & Carbone, 2014)
Under actuated	High compliance legs that only require an actuator at the hip.	(Saranli et al., 2001)
Telescopic leg	Increase in the maximum load capacity of the robot.	(Pfeiffer et al., 1995)
Hybrid leg	Foot is designed as a powered wheel that regulates the velocity and force of contact during the support phases.	(Nava Rodriguez et al., 2010)

The design of robotic legs can be classified in two more categories, legs orientation and joints configuration (see Figure 2-3). A rectangular hexapod robot can have two different configurations (see Figure 2-3 (a)): frontal or sagittal. In the first one, the directions are perpendicular to the advancement of the legs position. In the sagittal the movement is parallel to the robot legs. The circular assembly (Figure 2-3 (a)) as the legs positioned radially to the body, which allows movement in any direction.

Regarding joints configuration there are three possible designs (see Figure 2-3 (b)): knee outwards, knee inwards and knees in the same direction.



(a)



(b)

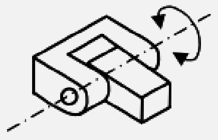
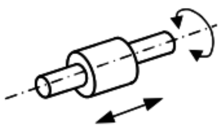
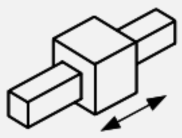
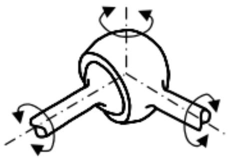
Figure 2-3 Hexapod legs orientation (a) and joints configuration (b). (Adapted from (Tedeschi & Carbone, 2014))

2.1.3. Joints

Joints are an inevitable part of robots since they are used to connect the mechanical parts and allow the relative movement between them. The principal objective of a mechanical joint is to hold parts together and transmit a loading force from a given structural component to an adjoining component (Josephs & Huston, 2018). Different type of movements requires distinct type of joints whether the movement is translational or rotational.

The main joint types are (Bajaj et al., 2015; Flores & J.C, 2005): cylindrical, revolute, prismatic and spherical joints. Table 2-2 explains the four types of joints.

Table 2-2 Types of Joints.

Joints		Description	DoF	Reference
Revolute		Allows a relative rotation movement.	1	(Flores & J.C, 2005)
Cylindrical		Allows one relative translation movement and one relative rotation movement.	2	(Flores & J.C, 2005)
Prismatic		Allows a relative translation movement.	1	(Flores & J.C, 2005)
Spherical		Allows rotation in three theoretic relative rotations.	3	(Flores & J.C, 2005; Talaba, 2012)

2.1.4. Sensors

Robots require sensors to perceive their surroundings in the physical world. Additionally, sensors for estimating the internal state of the robot can be included (Gao et al., 2022). For that, sensors are divided into two categories (He & Gao, 2020), namely proprioceptive and exteroceptive sensors.

Proprioceptive sensors are used to monitor the robot's internal systems and components. Encoders (Electromechanical device which calculates the joint angle and speed via electrical signals), torque sensors and IMU's (Inertial Measurement Units) are examples of this type of sensors.

Exteroceptive sensors measure environmental information using visual, non-visual and contact sensors. These sensors can be separated in two different sub-categories (He & Gao, 2020): for physical and geometrical information. The former group obtains data for the kinematic and dynamic control of the robot using contact sensors and the latter gets information for mapping and localization using visual and non-visual sensors.

Figure 2-4 shows the commonly used sensors in legged robots.

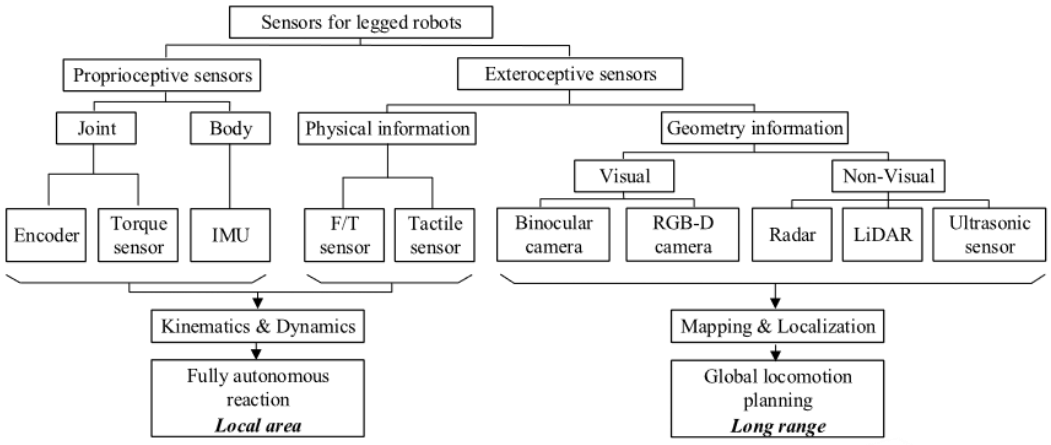


Figure 2-4 Sensors used in legged robots. (He & Gao, 2020).

2.1.5. Actuators

Actuators provide the necessary force and motion required for leg movement. There are several requirements for actuators, such as robustness or compliance, velocity control, low-impedance force control, high-power density (He & Gao, 2020).

The actuators can be divided into three main types (Gao et al., 2022; He & Gao, 2020), namely: electrical, pneumatic and hydraulic.

Electrical actuators convert electrical energy into mechanical motion, Table 2-3 shows their advantages and disadvantages. Pneumatic actuators use energy of compressed air or gas to produce force. These types of actuators are known for their high force output and fast response time as well as having a low cost. Table 2-4 displays their advantages and disadvantages. Hydraulic actuators use a pressurized fluid to generate mechanical motion. They provide extreme high power density and are robust against impulsive loads, usually found in heavy machinery. Table 2-5 represents their advantages and disadvantages.

Table 2-3 Advantages and disadvantages of electrical actuators. (Gao et al., 2022)

Advantages	**Disadvantages**
Precise control.	Limited Power.
Speed control.	High initial cost.
Energy efficient.	Possibility of over-heating.
Low maintenance.	Limited force output.
Easy installation.	Low stiffness.

Table 2-4 Advantages and disadvantages of pneumatic actuators. (Xavier et al., 2022)

Advantages	**Disadvantages**
High force output.	Limited Precision.
Fast response time.	Compliant systems.
Simple control.	Noisy systems.
Low maintenance.	Difficult to control their linear position.
Low cost.	Needs a constant supply of compressed air/gas.

Table 2-5 Advantages and disadvantages of hydraulic actuators. (He & Gao, 2020).

Advantages	**Disadvantages**
High power.	Low compliance.
Precise control.	Leakage possibility.
Robust.	Requires maintenance.
High force density.	Expensive.
High precision.	Viscosity of fluid changes with temperature.

Besides the presented actuators, some unconventional actuators applied to hexapod robots include (Tedeschi & Carbone, 2014):

- Ionic polymer-metal composites: when applied with voltage differences their shape change.
- Polyacrylonitrile: polymer gel that activates with changes in pH.
- Shape Memory Alloys (SMA): exhibit contraction with heat.

2.2. Control in Hexapod Robots

Hexapod robot control is crucial as it affects the robot's capacity to maneuver through challenging settings, carry out specified tasks, and interact with its surroundings. The many legs and multiple DoF on hexapod robots make the task of controlling difficult and complex.

A well-designed control system can make the robot move with agility, navigate challenging terrain, and perform a variety of jobs with great precision and accuracy. Not only that, but the control system is responsible for guaranteeing the security of the robot and its surroundings.

In this chapter, the performance indices will be presented since they are metrics that are used to evaluate the performance of a control system, and can be used to estimate the effectiveness of the control. Afterwards an analysis to the movement control, exploring existing controllers and focusing on the leg path. A study of posture control, focusing on several methods used in correcting the robot posture. Lastly, an analysis of control architecture, reviewing the main methods and studying the most used type of control architecture.

2.2.1. Performance Indices

Performance indices are established parameters to correctly evaluate and compare hexapod robots. Since a more complete analysis requires more than one parameter the following will be presented:

- Duty factor
- Statistic stability criteria
- Cost of transport (CoT)

Duty factor

The duty factor β is the ratio of the time a leg spends on the ground to the cycle time and is defined as (C.-D. Zhang & Song, 1993):

$$\beta = \frac{support\ period}{cycle\ time} \tag{2.1}$$

In a hexapod robot a duty factor β < 0.5 corresponds to a running gait and β > 0.5 is equivalent to a walking gait (He & Gao, 2020).

Statistic stability criteria

Hexapod robots need to stay statically stable at all the times during each gait in order not to fall with three or more legs in contact with the ground. Stability is achieved whenever the projection of the center of mass (CoM) is inside the support polygon that is comprised by the legs. This criterion uses the Zero Moment Point (ZMP) principle, which is defined as the minimum distance between the projection point of the CoM vector on the supporting polygon and each side of the polygon.

Cost of transport

The virtual robot used in this study has an overall of eighteen active joints, which is equivalent to a high energy consumption. Therefore, its motion needs to be energetically optimized to increase the system autonomy. The CoT is used to evaluate the swing trajectory and to identify the energy consumption for each gait cycle, and is expressed as follows,

$$\zeta = \frac{P}{mg\|\upsilon\|} \tag{2.2}$$

where P is the power consumption, which is obtained through the torque and angular velocity of each joint, m is the system mass, g is the gravitational acceleration, and υ is the overall robot velocity (Coelho et al., 2022).

2.2.2. Movement Control

Navigation in complex environments needs movement stability. This task is of utmost importance since with a successful application it may replace humans in explore and rescue missions.

A control system is necessary to trigger the robot's limbs and prevent it from falling during its movement. The system success depends on its ability to autonomously adapt to the terrain typology without damaging itself (Coelho et al., 2021).

Different types of control have been classified according with the methodology used to control the movement of a hexapod robot. Each methodology (traditional controllers, bio-inspired architectures and Reinforcement Learning) (see Figure 2-5) will be described below.

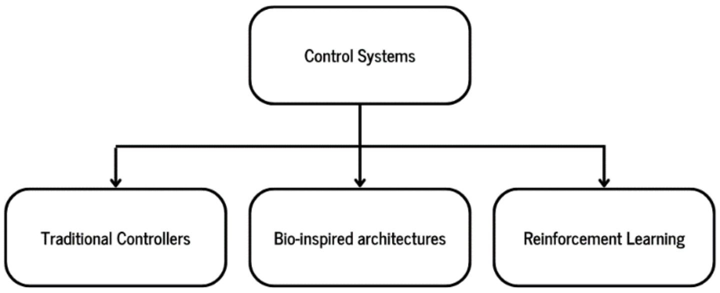


Figure 2-5 Control systems methodologies.

Traditional controllers

Traditional control systems consider the hexapod as a rigid body connected to six robot manipulators and analyze the actuation and control of each leg individually. Furthermore, the locomotion is described through kinematic and dynamic models. Since there are a vast number of adopted methodologies for design of these control architectures, they were grouped in three categories (Coelho et al., 2021).

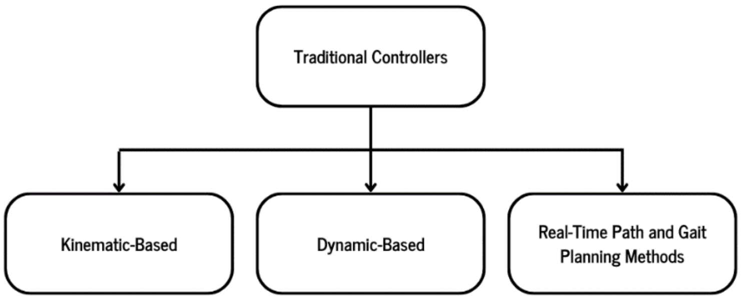


Figure 2-6 Traditional controllers categories.

Kinematic-Based Control

This methodology relies on the calculation of the desired angular position of the joints or the torque of their actuators according to the desired motion of the limbs. The environment influences the stability of the robot, and to generate adaptive locomotion, it is necessary to obtain a perception of the surroundings.

Most methods to generate adaptive gaits consider an evaluation of the contact forces of the feet to detect the roughness of the terrain (Coelho et al., 2021).

Zha et al., (2019) did research about a searching algorithm, which aimed at determining new footholds when the limbs could not detect contact forces during the swing phase.

This research applied a constraint of stability, the Static Stability Margin (SSM) to evaluate the stability of the gait. This method evaluates the distance of the ground projection of the CoM of the robot toward the edges of the Support Polygon (see Figure 2-7). If the CoM is inside the SP, the robot is statically stable. Coelho et al., (2021) states that SSM is used to ensure that the robot is upright during the transition between the phases of the gait. Rojas et al., (2015) states that to solve a problem in an environment with a large number of forbidden locations a different constraint was used. The longitudinal stability margin (LSM) has a different methodology, where the minimal distance is calculated considering the direction of motion (see Figure 2-8).

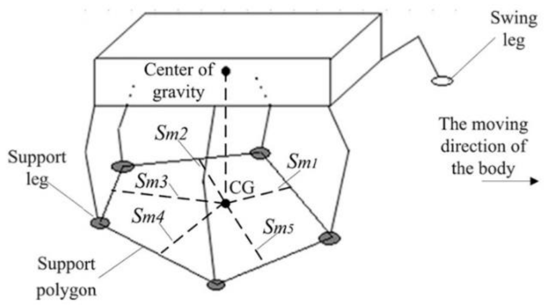


Figure 2-7 SSM of a hexapod. (Zha et al., 2019).

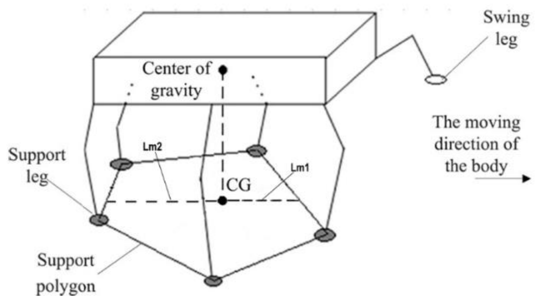


Figure 2-8 LSM of a hexapod. (Adapted from (Zha et al., 2019)).

Dynamic-Based Control

The robot's dynamic formulation enables the investigation of the motion deviation caused by external forces and torques applied to the hexapod in order to find the values of its interaction with the environment (Coelho et al., 2021).

This type of control is explored with different approaches, presented in Table 2-6.

Table 2-6 Dynamic based control research.

Research	Description	Ref
Dynamic gait stability margin (DGSM)	Evaluates the stability of a gait through the generated angular momentum and the edges of the SP.	(Mahapatra et al., 2019)
Spring-loaded inverted pendulum (SLIP)	The system estimated the torque values of the actuators, which ensured the virtual dynamic model of the limbs behaved as linear springs during the stance phase.	(Soyguder & Alli, 2012)
Compliance controller + kalman filter	Adjusts the position of the limbs based on the error between the measured and the expected contact forces. The Kalman filter predicts the ZMP and verifies if it is within the SP.	(Deng et al., 2017)
Impedance controller	Increase stability of the motion and its energy efficiency. The torque of each joint is used to calculate the roughness of the ground.	(Bjelonic et al., 2018)
Euler-Lagrange method	Determine the deviations between the desired and real positions of the joints through the contact forces and the generated torques of the actuators.	(Faigl & Čížek, 2019)

One of the most important applications of these models is the evaluation of the contact forces between the feet and the ground. Not only that, another advantage of using the dynamic model of the robot is the possibility of evaluating the energy consumption of the actuators (Coelho et al., 2021).

Real-Time Path and Gait Planning Methods

In this subsection some of the methods used will be presented, where one of the main criteria is the processing time required to define the best trajectory for the robot to follow in a complex environment.

Some methods discuss the use of computer vision algorithms to calculate safe trajectories. These algorithms were also used to teach the hexapod to adapt its behaviour in unknown environments.

A brief description of each method will be presented in Table 2-7.

Table 2-7 Real-Time path and gait planning methods.

Method	Description	Ref
Rapidly exploring random tree (RRT)	The algorithm was implemented to estimate a safe path in an unknown environment. Not only that but it was also used for path planning in an environment with obstacles.	(M. S. Khan et al., 2015)
Artificial neural network (ANN) + Fuzzy logic	The method aimed at reducing the time a controller needs to plan its gait. Fuzzy Logic determined the correct actuation of the limbs and ANN decided the most adequate locomotion.	(Coelho et al., 2021)
Support vector machine	The robot adjusts the position of its CoM to avoid unsafe ground using the contact forces when walking across brittle surfaces.	(Coelho et al., 2021)

Bio-Inspired Controllers

Bio-inspired control architectures aim at mimicking the biologic process of generation of locomotion. The goal is to provide an optimal adaptation of the behaviour of the hexapod to the environment. Inspired in biological principles from the Central Pattern Generator (CPG) these systems have a high control center, similar to an animal brain (Coelho et al., 2021).

These systems use coupled oscillators to rhythmically activate the swing and the stance phase of the limbs. Since there are different type of oscillators, they are presented in Figure 2-9. The interest of implementing sensory feedback in bio-inspired architectures is to produce adaptive locomotion by modifying the oscillator's parameters and output signal. The output of the coupled oscillators is converted to angular positions of the joints.

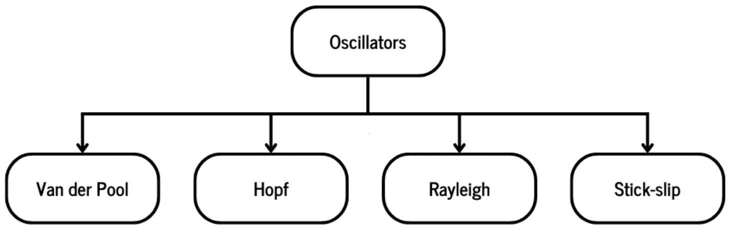


Figure 2-9 Types of oscillators.(Coelho et al., 2021).

Reinforcement Learning

By several trial-and-error encounters between the robot and its surroundings, RL gives a control system the capacity to learn how to operate in order to accomplish the intended goal. This type of application gives the robot a superior level of autonomy without the need to learn previous information about the environment (Coelho et al., 2021).

Since the subject of RL is further clarified in Reinforcement Learning a short explanation of the three main applications (Figure 2-10) of RL in the autonomous learning of a hexapod robot in an environment is provided. Hong et al., (2017) presented a combined Fuzzy Logic with Q-learning to generate real-time control for obstacle avoidance. In this method the algorithm had a fast convergence and learned an optimal strategy, which enabled the avoidance of obstacles.

Liu et al., (2019) used the Monte Carlo method to optimize the Markov decision process and generate a stable gait. With the help of this technique, the issue of determining the likelihood of transition states, a component of the Markov decision process that affects the stability margin position states, is resolved. The robot could successfully adjust its locomotion to different surfaces.

Recent research found two different approaches to the damage recovery problem. Verma et al., (2019) suggested a Neural Network to self-diagnose the damages and the algorithm could find the optimal gait with one or two limbs harmed. Chatzilygeroudis & Mouret, (2017) pointed a method which required a high computational power, employed a reset-free trial-and-error algorithm that stored and mapped all policies to select the one that had the maximum expected reward.

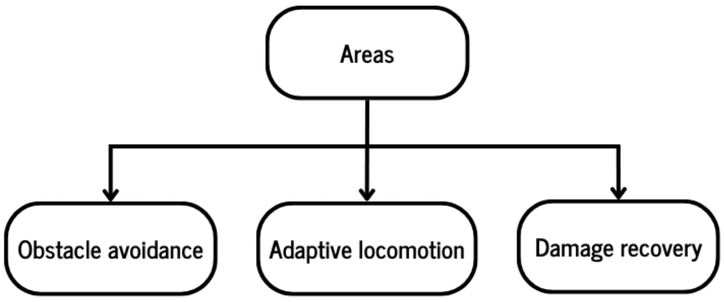


Figure 2-10 Areas of application of RL in autonomous hexapod robot's movement.(Coelho et al., 2021).

2.2.3. Gaits

A gait defines the pattern of leg movements that a robot uses to move and its critical for the locomotion and overall functionality of hexapod robots. It plays a crucial role in determining the efficiency, stability, maneuverability, and adaptability of the robot. There are three main terms that describe gait (Alexander, 1984): duty factor (already introduced in Duty factor), stride and relative phase.

The stride is a complete cycle of leg movements, from the setting down of a foot to next setting down of the same foot. The stride length, λ, is the distance travelled in a stride and the stride frequency, *f*, is the number of strides in unit time. This criterion is mainly used with periodic gaits.

The relative phase criterion is a measure used to characterize the synchronization of leg motions in hexapod robots. It calculates the time difference between two leg movements as a proportion of the entire gait cycle time. Changing the relative phase criteria can optimize the robot's gait for varied jobs and settings, impacting the stability and efficiency of the robot's movement.

Hexapod robots can have two different setups in terms of gaits (Xu & Ding, 2014) (see Figure 2-11): Periodic and aperiodic gait.

Periodic gaits refer to a sequence of leg movements that repeat after a fixed interval of time, creating a periodic pattern. Considered more stable and efficient, making them suitable for applications that require precise control over the robot’s movement. Aperiodic gaits do not repeat in a periodic pattern. They are more complex and provide flexibility and adaptability in unpredictable terrain.

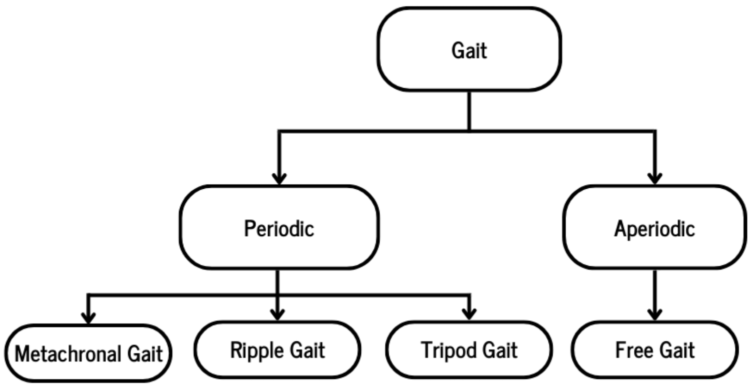


Figure 2-11 Types of hexapod's gaits.

There are four common gaits of hexapods (see Figure 2-11) (Alexander, 1984; Campos et al., 2010): metachronal, ripple, tripod, and free gait.

Since periodic gaits have the characteristic of repetition it is possible to extract their gait diagrams, which represents the sequence of leg movements in a hexapod robot to achieve locomotion.

The metachronal gait (see Figure 2-12 (b)) is adopted by the hexapod in a slow movement. This gait is described as back to front propagating “wave”, moving one leg at a time.

The ripple gait (see Figure 2-12 (c)) is considered a medium speed movement. Limbs move in groups of two in the same phase, for example R1 and L3, L1 and R3, and L2 and R2.

The tripod gait (see Figure 2-12 (d)) has the top speed. Limbs move in groups of three in the same phase (L1, L3 and R2, and R1, R3, and L2). Free gait is a versatile and effective walking gait for hexapod robots, allowing them to move in any direction and adapt to a wide range of environments.
It is usually adopted when the robot needs to change its gait according to real-time changes of the terrain and its state information, generating a sequence of gaits with an irregular order. The order of legs changes in a non-fixed but flexible manner (Ding et al., 2020).

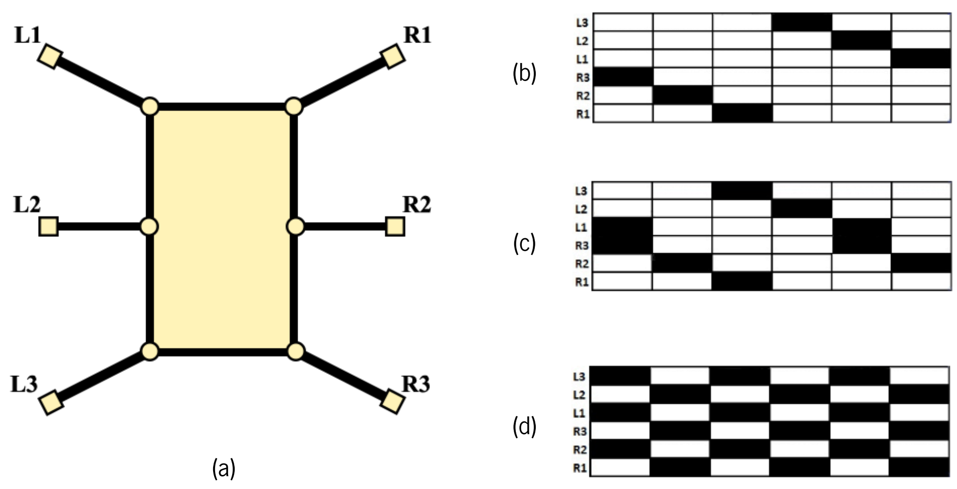


Figure 2-12 (a) Legs distribution; (b) Metachronal gait diagram; (c) Ripple gait diagram; (d) Tripod gait diagram.((b), (c) and (d) adapted from(Campos et al., 2010)).

2.2.4. Posture Control

A hexapod robot posture is related with stability, flexibility and adaptability in a complex environment. Posture control is of utmost importance since the relevance of transposing rough terrain, not only effectively but also quickly, has been growing exponentially (Y. Liu et al., 2020).

Y. Liu et al., (2020) stated that the methods used to control the robot's posture are divided in three categories (Table 2-8): Optimal pose method, suspension control method and the force/position hybrid control method.

Table 2-8 Posture control methods.(Y. Liu et al., 2020).

Method	**Description**
Optimal pose	From the perspective of trunk position control, the robot's posture is controlled to make the hexapod robot have better stability and flexibility. The stability and robustness of the torso are increased.
Suspension control	Controls the posture of the robot from the perspective of force control. Foot force perception information is combined with the control algorithm to supress the influence of external interference on trunk posture.
Force/position hybrid control	From the perspective of force/position hybrid control realizes stable walking by adjusting the position of the CoM.

The three methods have disadvantages, which include: low flexibility, heavy computational burden, and difficulty in establishing the control method.

In regard to the optimal pose, the hexapod robot uses the position control to adjust the posture, that leads to the heavy calculation burden, challenging the real-time performance. Suspension control, with the force control strategy avoids the disadvantages of the latter method, but it has a poor adaptability to complex terrain. This creates the need of a hybrid method of pose control that can avoid complex kinematic solutions and has a good operability (Y. Liu et al., 2020).

2.2.5. Control Architectures

The choice of an appropriate control architecture is a major design issue. A well-planned architecture simplifies robot system management. There are two main concepts: Architectural structure and architectural style. The former refers to how a system is divided into subsystems and how those subsystems interact, and the latter refers to the computational concepts that underlie a given system (Kortenkamp & Simmons, 2008).

There are four main architectural paradigms (see Figure 2-13) and they are shortly presented (Ingrand & Ghallab, 2017; Kortenkamp & Simmons, 2008).

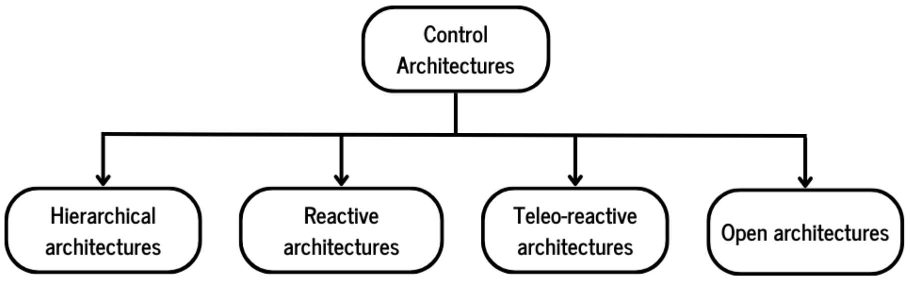


Figure 2-13 Types of control architectures.(Adapted from (Ingrand & Ghallab, 2017)).

The most widely used in robotics is hierarchical architectures, modular components are themselves built on top of other modular components. They organize the software into layers with different temporal requirements. This type of layered decomposition reduces system complexity through abstraction, and they clearly identify and organize the sensory-motor components and the deliberation components.

In reactive architectures plans are generated quickly and relied more directly on sensed information. Composed of input/output automata implementing loops from sensors to effectors. Despite being categorized as a different architecture they can be organized hierarchically.

Teleo-reactive architectures has the same general structure but operate at increasingly lower frequency as they move from the servo level to the reasoning levels.

A paradigm based on planning-acting components distributed at different levels. They offer advantages with respect to the consistency of the system, but have issues scaling up to complex and dynamic environments.

Open architectures are more recent, designed to fulfil the needs of service and personal robots in open environments. They have a distinct characteristic, which is to allow a robot to take data and models over the web.

Since the three-layered robot architectures (see Figure 2-14) are very popular an example will be provided and explained. The lowest layer is behavioural control and is the layer tied most closely to sensors and actuators. The second layer is the executive layer and is responsible for choosing the current behaviours of the robot to achieve a task. The highest layer is the task-planning, responsible for the long-terms goals of the robot. The behaviour layer perceives the environment and carries out the actions of the robot, it is designed to bring speed and reactivity to robot control. The middle layer is the interface between the numerical behavioural control and the symbolic planning layer. It is responsible for translating high-level plans into low-level behaviours. Lastly, the planning layer looks towards the future, it is responsible for determining the long-range activities of the robot based on high-level goals.

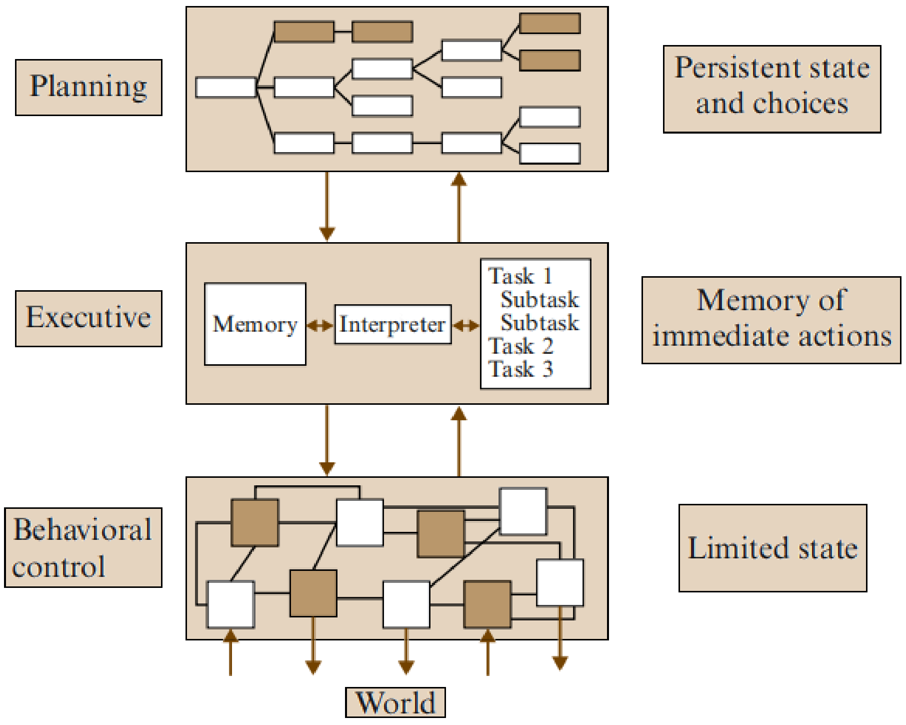


Figure 2-14 Typical tree-layered architecture.(Adapted from (Kortenkamp & Simmons, 2008)).

The architectural style defines how the architecture components communicate with each other. Every component needs to exchange data and send commands and for this there are two basic approaches:

- Client-server: A communication protocol, where the components talk directly with other components. A component can call functions and procedures of another component.
- Publish-subscribe: A communication protocol, where a component publishes data, and any other component can subscribe to that data. In a typical architecture, most components both publish information and subscribe information published by other components.

2.3. Reinforcement Learning

Artificial Intelligence (AI) entry into the industry caused a revolution. Facing an exponential evolution, with great advances in computer hardware and capability, decreased costs in data storage this technology remarks itself superior to the others.

The principal objectives of Artificial Intelligence can be found in Figure 2-15.

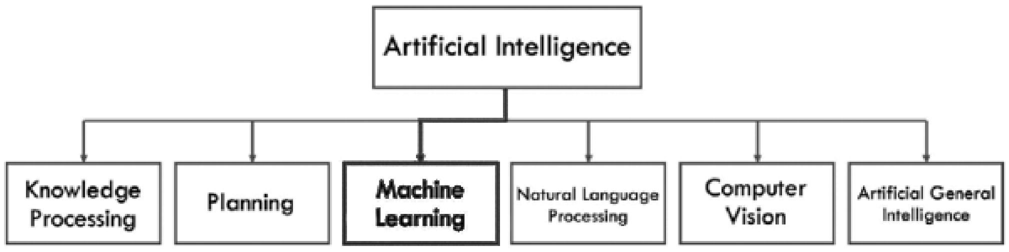


Figure 2-15 Artificial Intelligence objectives.(Adapted from (Nian et al., 2020)).

Machine Learning is a field of Artificial Intelligence, and it can be described as the subject of science that studies and develops algorithms and statistical models to give machines the ability to learn tasks without being programmed for them (Nian et al., 2020).

This topic can be divided in three categories: Supervised Learning (SP), Unsupervised Learning (UL) and Reinforcement Learning (RL) (Figure 2-16). The latter is the main focus of this work.

A Supervised Learning algorithm attempts to generalize across training examples and uses this knowledge to predict labels for unseen data. SP is not capable of outperforming the subject matter expert since the algorithm only mimics the labelling behaviour of the expert. Unsupervised Learning is used for identifying hidden structures within unlabeled data sets, grouping them in categories. The main goals of UL are dimensional reduction, feature extraction and clustering (Nian et al., 2020).

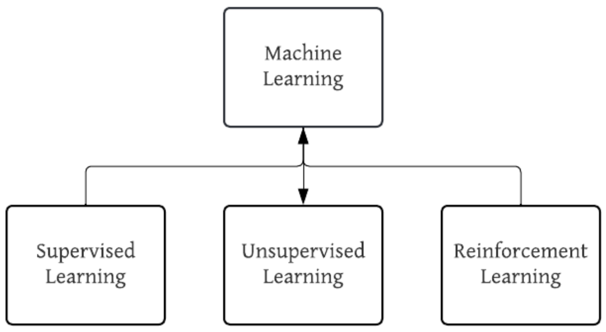


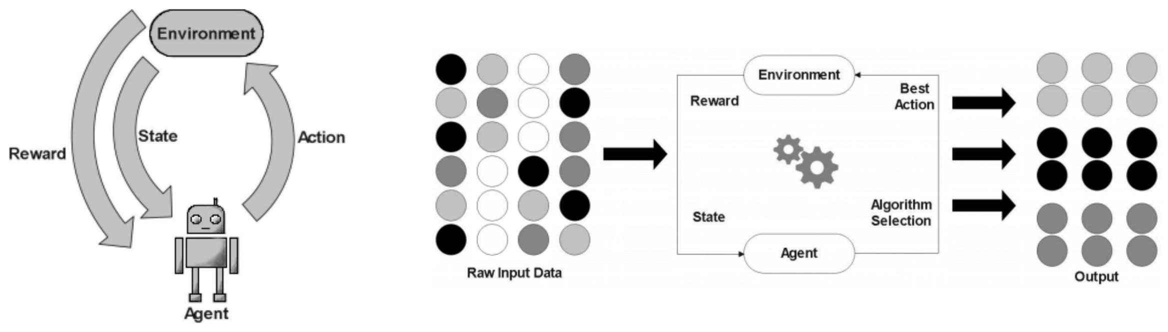
Figure 2-16 Categories of Machine Learning.

Reinforcement Learning (Figure 2-17) presents a new methodology that attempts to give the machines the ability to find solutions that the previously mentioned could not find.

Through an optimal mapping of situations to actions through a trial-and-error search using a system of delayed rewards (also called delayed feedback). These two features distinguish RL from all the other algorithms (Nian et al., 2020).

RL presents algorithms with an agent that learns and interacts with a state space to reach a goal. The agent must be able to sense the state of its environment and perform actions that affect the state. Sutton & Barto, (2018) define that any method capable of solving problems that has an agent capable of sensation, action and goal should be considered a RL method.

These methods have a similar learning to humans and animals and the main part of RL algorithms are based in biological systems of learning.



(a) (b)

Figure 2-17 (a) Basic RL structure; (b) Basic RL architecture.(Adapted from (Dridi, 2021)).

2.3.1. Terminology

Although Reinforcement Learning has many different algorithms with different methodologies of study and application, they still have common terms that should be presented. Table 2-9 show the common terms used in RL and exhibit a brief description.

Table 2-9 Terms used in a RL Model.

Term	Description
Agent	Performs a specific action in an environment in order to receive a reward. It´s main objective is to maximize the cumulative reward.
Environment	Scenario where the agent has to navigate. Includes all objects, events and rules that govern the agent's behaviour and influence the rewards it receives.
State	Current situation in which the agent is present. Contains all the relevant information that the agent needs to choose an appropriate action.
Action	A move/task performed by the agent in an environment in a given state.
Reward	A scalar value that represents the feedback given to the agent for its actions in a particular state. The goal is to influence the agent to learn to maximize the cumulative reward by taking appropriate actions in different states.
Policy	Strategy chosen by the agent based on the current state in order to decide the next task or action.
Value	Long-term reward, or a reward with a discount. Represents the expected cumulative reward an agent can obtain from a state or state-action pair.
Value function	The total reward and identifies the value of the state.

2.3.2. Learning Approaches

The learning approaches of RL agents or RL algorithms can be categorized into two classes:

- Model-based RL: The agent employs a predictive model to learn about the control policy from the environment through a very small number of interactions, and then the model is applied to the subsequent episodes to obtain the rewards.
- Model-free RL: Without any model, the agent learns about the control policy from the environment through trial and error in order to maximize rewards.

Khan et al in (Khan et al., 2020) state that model-free RL has proved itself as a promising approach in the fields of robotics in comparison to model-based RL.

Figure 2-18 shows RL algorithms classification and present some RL algorithms.

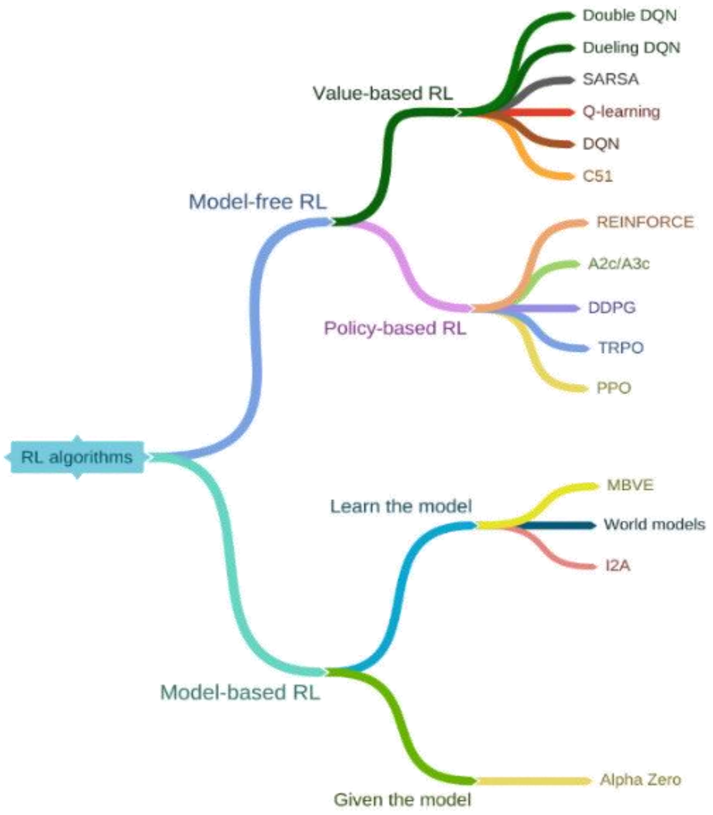


Figure 2-18 Classification of RL algorithms. (Khan et al., 2020)

2.3.3. Markov Decision Process

An important mathematical framework that formalizes the Reinforcement Learning problem as a sequential decision-making process is the Markov Decision Process (MDP). MDP is a discrete-time stochastic control process. This framework is intended to be a simple way of representing essential features of the reinforcement learning problem.

MDP's are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. A MPD manifests a sequence of decision tasks in which, at each time step, the agent observes the current state of the environment and decides upon an action. A transition occurs in the environment, it moves to the next state and the agent receives a reward, either positive or negative. The agent will continue to act in the environment until it reaches a terminal state at a given time step, *T*, or until it completes the goal (Fu et al., 2022).

The reward that the agent gets at each step its affected by a discount factor, γ, and the total reward is defined as the return, *G*.

There are three fundamental classes of methods for solving finite Markov decision problems: Dynamic Programming, Monte Carlo methods and temporal difference learning (Sutton, 2018).

Dynamic Programming

Dynamic programming (DP) is a set of methods that may be used to compute optimal policies given a perfect model of the environment in the form of a Markov decision process. These methods are well developed mathematically but require a complete and accurate model of the environment, which has a great computational expense.

The main idea of DP is the use of value functions to organize and structure the search for good policies.

Monte Carlo Methods

In contrast with dynamic programming, in these methods the agent doesn’t have access to a complete model of the environment and must learn from experience. To produce data identical to data gathered from a real phenomenon that complies with the model's requirements, Monte Carlo techniques are employed.

In order to solve problems statistically when there is no deterministic model, the basic idea behind Monte Carlo methods is to estimate complex or uncertain outcomes by repeatedly creating random samples and averaging their results. By utilizing the law of large numbers, these techniques converge to precise estimations as the quantity of samples or trials grows.

Temporal difference methods

Temporal difference (TD) methods learn a value function that estimates the expected long-term reward for taking a particular action in a state, it learns from experience by iteratively updating value estimates based on the difference between predicted and observed outcomes, which allows the agent to evaluate and improve its policy without waiting for the end of episodes (Fernández-Conde et al., 2022). Dynamic programming and Monte Carlo techniques are combined in TD methods to create a more effective strategy. The main TD methods that exist are:

- Q-learning – Off- policy method used to approximate the optimal action-value function. It iteratively updates *q-values* using the Bellman equation and the maximum expected return over all possible actions.
- SARSA – On-policy method used to learn the action-value function. It updates *q-values* based on the *q-value* of the next state-action pair, considering the action chosen in the next state.
- Expected SARSA – On-policy method that extends SARSA by estimating the expected *q-value* for the next state-action pair, considering the probability distribution of possible actions in the next state.

Considering that this research will evaluate the hexapod actions in different environments by adjusting control parameters that affect how the robot regulates its movement while maneuvering and adapting to changes in the inclination of the terrain, the most suitable method to apply is the Q-Learning.

2.3.4. Q-Learning

Q-Learning assesses the quality of the action taken. A model-free RL that handles problems with stochastic transitions and rewards. A Q-Learning algorithm is a temporal difference method that forecasts a sum dependent on the future values of the signal (Abdulqadir & Abdulazeez, 2021).

Known as off-police RL because it learns the action that is outside the current policy. A Q-table is created to store the state and action values, which is the main reference for the agent to choose the best actions based on their values in the Q-table. Then, the agent interacts with the environment and updates the state-action pairs in the Q-table. The actions can be taken as exploiting or exploring.

The agent exploits when it uses Q-table to view all Q-values and chooses the action based on the maximum Q-value. Random actions performed by the agent are considered exploring. Exploiting depends on maximum future reward, whereas exploring depends on acting randomly (Dridi, 2021).

Q-Learning is defined by,

$$Q^{new}(\mathrm{S}_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \cdot \max_a Q(S_{t+1}, \alpha) - Q(S_t, A_t)\right]. \tag{2.3}$$

where $Q(S_t, A_t)$ is the current Q-table variable value, α is the learning rate $(0 < \alpha \leq 1)$, that affects the degree to which the Q-values are updated during the process, γ is the discount factor $(0 < \gamma \leq 1)$ and $\max_a Q(S_{t+1}, \alpha)$ is the estimate of optimal future value. This algorithm is a variation of the Bellman equation as a simple value iteration update. The Q-values are based on the benefit gained by the selection of activity plus the overall possible reward predicted.

2.4. Summary and Conclusions

In this chapter an extensive analysis was made in order to comprehend the relevance of hexapod robots when compared with other legged robots. A research of the hexapod robots main features, namely the main body structure, legs and joints arrangements and also the most common sets of actuators and sensors, is made to understand the advantages of disadvantages of this type of robots.

Regarding the control of hexapod robots an overview of the performance indices is executed since they are effective at estimating the quality of a control system. Not only that, but the movement control and gaits are evaluated and divided into sections to further understand the use of movement stability in this research. When considering hexapod gaits, the tripod gait is defined as the default walking gait, considering it has the top speed between all periodic gaits and, since it has the highest oscillations, it is the best choice for stability research. In respect to control architectures, the hierarchical architecture is the one applied in this investigation since it is the most researched topic and it has a rather simple system complexity, which aids with comprehension.

Lastly, a thorough study on Reinforcement Learning that presents the common terminology, the learning approaches and learning algorithms. Q-learning, a model-free approach where the agent learns with a control policy in an environment through trial and error, is selected. The use of a Q-table that is composed of actions and states can be adapted to the use of different environments, in this case different worlds with diverse slopes, and a control parameter that has several options. By assimilating the different control parameter values with all the slope environments, it is possible to find a suitable match with the use of Q-Learning.

3. HEXAPOD MODEL AND GAIT ANALYSIS

In this chapter a description of the hexapod robot ATHENA used in this study is presented. A kinematic analysis of the limb is showed for the purpose of understanding the robot's walking. The analysis of the limbs trajectory is also presented in order to establish the best algorithm to use along with the posture control.

3.1. HEXAPOD MODEL DESCRIPTION

In this section, a description of the hexapod robot ATHENA considered in this study is presented. Considering that the physical prototype is not used in the study an exposition is made in order to assist comprehension. The physical body is depicted in Figure 3-1. The robot is composed of 25 rigid bodies, which are interconnected by 24 kinematic joints. The limbs are distributed in two groups of three legs placed symmetrically along the two sides. Each limb includes three revolute joints that are actuated by servo motor with a stall torque of 1.89 N.m. The revolute joints connect the torso with the coxa, femur and tibia segments of each leg, named respectively Thorax-Coxa(TC), Coxa-Trochaterofemur(CTr), and Femur-Tibia(FTi). The hexapod feet are connected to the tibia segments by passive prismatic joints, that absorb the impact with compression springs.

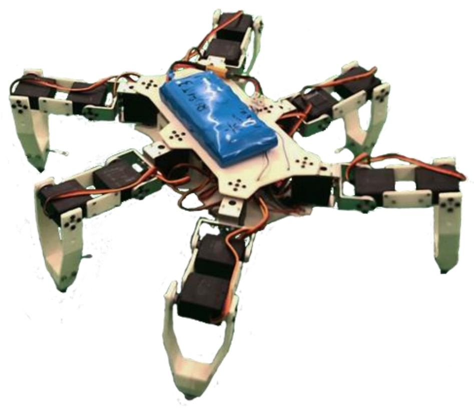


Figure 3-1 Physical model of the ATHENA robot.

Since this research is based on computation simulations, a simplified model of the robot was used in the *Gazebo* software. The system geometry has a minimal influence on the results, as long as the inertial properties are equal, the hexapod modelled in *Gazebo* was composed using simple geometries.

The torso and the coxa, femur and tibia segments were converted into prisms and the feet were modelled as spheres (Figure 3-2) which helps the computational simulations in terms of collision with the ground since there is only one contact point in each point. In order to replicate the physical model joints, all the revolute joints were set to a maximum torque of 1.89 N.m. Also, the connection between the tibia and the foot is assumed fixed.

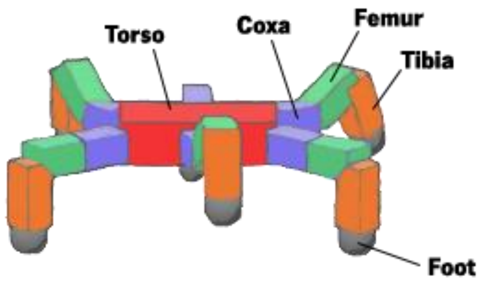


Figure 3-2 Computational multibody model of ATHENA robot.

3.2. Kinematic Model

The hexapod's kinematic model is important for the control of the limbs motion. This is used to control the robot's movement, which requires the determination of the relationship between the angular positions of the joints and the coordinates of the robot's feet. Using the computational model of the ATHENA robot the torso's center of mass is defined as the reference point for all calculations. In addition, for simplification, each separate leg is considered an isolated system, also the tibia and the foot are combined into a single body. The kinematic model of a leg is depicted in Figure 3-3. The angles θ_2, θ_3 and θ_4 are the relative angles of the TC, CTr, and FTi joints, respectively. The motion range of the TC joint is set to $\theta_2 \in [-0.612, 0.612]$rad, and the angular position of the remaining joints is constrained to $\theta_3, \theta_4 \in [-1.484, 1.484]$rad.

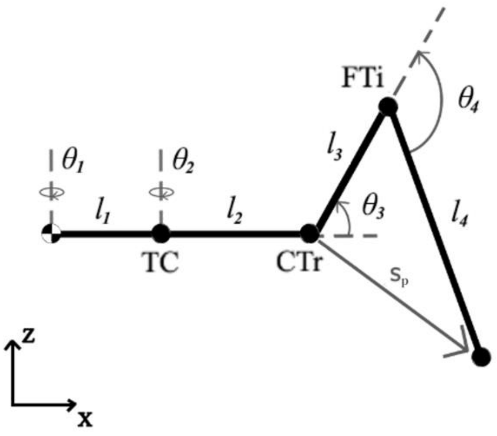


Figure 3-3 Leg kinematic model.

The locomotion control is based in two different estimations from the kinematic model. Firstly, get the current feet coordinates based on the joints' angular positions to check if the desired motion is inside the system's workspace. Secondly, in order to calculate the joints' configuration, it is used the trajectory applied to the feet. The first estimation is based on the forward kinematics configurations and for that the Denavit-Hartenberg convention is applied to obtain the relative position of the foot w.r.t the robot's center of mass, for simplification it will be referenced as *O*. The transformation matrix between two consecutive references is expressed as,

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where i corresponds to the reference number, a_i and d_i denote the translation along the x and z axes, and α_i θ_i correspond to the rotation along the same axes. Using the Denavit-Hartenberg convention and the parameters presented in Table 3-1, it is possible to obtain the relative transformation between the reference *O* and the hexapod feet. This transformation can be determined as,

$$\mathbf{T}_4^0 = \mathbf{T}_1^0\mathbf{T}_2^1\mathbf{T}_3^2\mathbf{T}_4^3 \tag{3.2}$$

which can be reformulated as,

$$\mathbf{T}_4^0 = \begin{bmatrix} \cos(\theta_1+\theta_2)\cos(\theta_3+\theta_4) & -\cos(\theta_1+\theta_2)\sin(\theta_3+\theta_4) & \sin(\theta_1+\theta_2) & x_p^O \\ \sin(\theta_1+\theta_2)\cos(\theta_3+\theta_4) & -\sin(\theta_1+\theta_2)\sin(\theta_3+\theta_4) & \cos(\theta_1+\theta_2) & y_p^O \\ \sin(\theta_3+\theta_4) & \cos(\theta_3+\theta_4) & 0 & z_p^O \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where the vector $\boldsymbol{p}_i^O = \left\{x_p^O,\ y_p^O,\ z_p^O\right\}, i \in [1,6]$ contains the feet' relative coordinates, which are expressed as,

$$\boldsymbol{p}_i^O = \begin{bmatrix} \cos(\theta_1+\theta_2)(l_4\cos(\theta_3+\theta_4)+l_3\cos\theta_3+l_2)+l_1\cos\theta_1 \\ \sin(\theta_1+\theta_2)(l_4\sin(\theta_3+\theta_4)+l_3\sin\theta_3+l_2)+l_1\sin\theta_1 \\ l_4\sin(\theta_3+\theta_4)\ +l_3\sin\theta_3 \end{bmatrix} \tag{3.4}$$

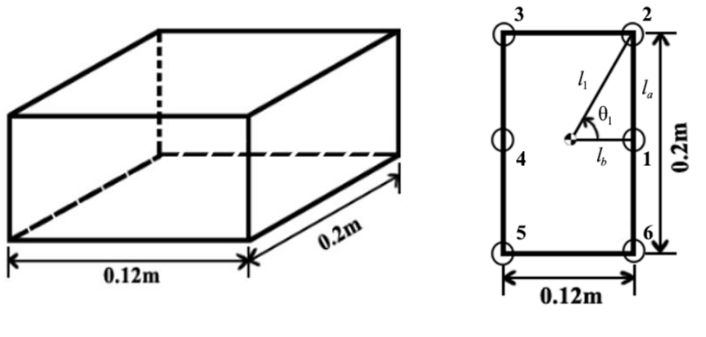
Table 3-1 Denavit-Hartenberg parameters for all relative transformations between *O* and the hexapod foot.

Transformation	α_i**[rad]**	a_i**[mm]**	θ_i**[rad]**	d_i**[mm]**
$\{0\} \rightarrow \{1\}$	0	l_1	θ_1	0
$\{1\} \rightarrow \{2\}$	$\frac{\pi}{2}$	l_2	θ_2	0
$\{2\} \rightarrow \{3\}$	0	l_3	θ_3	0
$\{3\} \rightarrow \{4\}$	0	l_4	θ_4	0

The second estimation uses inverse kinematics calculations to control the joints positions. Using the feet's position to obtain the relative angular positions of the joints. Starting with θ_1, which is constant and depends on the leg position as,

$$\theta_1 = \begin{cases} 0, i = 1 \\ \arctan\left(\frac{l_a}{l_b}\right), \forall i \in \{2,3,5,6\} \\ \pi, i = 4 \end{cases} \tag{3.5}$$

Figure 3-4 shows how the θ_1 value is obtained.



(a) (b)

Figure 3-4 Hexapod Design: (a) portrays the torso measurements, (b) portrays the θ_1 value in regards of the leg.

To calculate θ_2 a reference Q is placed on the TC joint. So, the angular position of this joint is defined as,

$$\theta_2 = \arctan\left(\frac{x_p^Q}{y_p^Q}\right) \tag{3.6}$$

in which $\boldsymbol{p}_i^Q = \left\{x_p^Q, y_p^Q, z_p^Q\right\}$ denotes the relative position of $\boldsymbol{p}_i$ w.r.t the reference Q, and is expressed as,

$$\boldsymbol{p}_i^Q = (\mathbf{T}_1^0)^{-1}\boldsymbol{p}_i^O \tag{3.7}$$

where $\mathbf{T}_1^0$ is the transformation matrix between the torso reference O and Q. By applying forward kinematics with the TC joint coordinates and θ_1 the CTr joint coordinates, reference U, are calculated,

$$U_i^Q = (\mathbf{T}_1^0)^{-1}\mathbf{T}_2^0 \tag{3.8}$$

Considering the vector $\mathbf{s_p}$, which is the size of the vector between the reference U and the foot, the value of the CTr and the FTi joints are as follows,

$$\theta_3 = \arccos\left(\frac{-l_4^2 + l_3^2 + \|s_p\|^2}{2l_3\|s_p\|}\right) - \arcsin\left(\frac{z^U - z^p}{\|s_p\|}\right) \tag{3.9}$$

$$\theta_4 = \pi + \arccos\left(\frac{-\|s_p\|^2 + l_4^2 + l_3^2}{2l_3 l_4}\right) \tag{3.10}$$

3.3. Gait Generation

The study of gait generation aims at analyzing the influence of the limbs' trajectory in the robot locomotion. Although the feet are not in contact with the ground during most of the swing phase, their velocity and acceleration influence the torso posture and the overall consumption rate. Thus, it is necessary to compare different algorithms to choose the one that has a lower impact in the variation of its posture. In order to reduce the influence of deviations in the system stability the same hexapod gait will be applied. The motion applied in ATHENA is the tripod pattern. The legs are divided in two locomotion groups, alternating between swing and stance phase. Since there are two different phases, each is going to have a distinct trajectory. In the stance phase the same trajectory will be applied in each case as the aim is for the limbs to support the robots' weight by ensuring the contact with the ground. For the swing phase, this research considers five different functions – a triangular function, a parabola, a cubic and a fourth-degree Bezier curves and a cubic Spline Curve.

For a more complete study all different algorithms were applied in four different environments – a normal plane world, a normal plane world with steps, a ramp with 10-degree slope and a terrain composed by uneven blocks, that will be referred as a testbed.

3.3.1. Stance Phase

The stance phase main requirements are to support the torso weight and ensure the feet contact with the ground. The trajectory of the leg during the stance phase is as it follows,

$$f_0(t) = \begin{Bmatrix} x(t) \\ y(t) \\ z(t) \end{Bmatrix} = \begin{Bmatrix} x_{p,0} \\ y_{p,0} + \frac{S}{T}t \\ z_{p,0} \end{Bmatrix} \tag{3.11}$$

where S is the robot step length, T is the time of a step cycle and and $\boldsymbol{p}_{i,0} = \{x_{p,0},\ y_{p,0},\ z_{p,0}\}^{\mathrm{T}}$ is the foot initial position.

3.3.2. Swing Phase

The swing phase analysis compares five different functions, the first one combines two linear trajectories, one for the foot ascending motion and another for the descending motion.

For the ascending motion the trajectory is expressed as,

$$f_1(t) = \begin{Bmatrix} x(t) \\ y(t) \\ z(t) \end{Bmatrix} = \begin{Bmatrix} x_{p,0} \\ y_{p,0} + \frac{S}{T}t \\ 2\frac{h - z_{p,0}}{T}t + z_{p,0} \end{Bmatrix} \text{for } t \le \frac{T}{2} \tag{3.12}$$

where h is the trajectory maximum height. As for the limb descending motion, it is defined as,

$$f_1(t) = \begin{Bmatrix} x(t) \\ y(t) \\ z(t) \end{Bmatrix} = \begin{Bmatrix} x_{p,0} \\ y_{p,0} + \frac{S}{T}t \\ -2\frac{h - z_{p,0}}{T}t + z_{p,0} + 2h \end{Bmatrix} \text{for } t > \frac{T}{2} \tag{3.13}$$

The parabola is a second degree polynomial function, which is described as,

$$f_2(t) = at^2 + bt + c \tag{3.14}$$

where a, b and c are constants which must be obtained through three control points $\mathbf{c_i} \in [0, 2]$. Assuming that $\mathbf{c_0} = \mathbf{p}_{i,0}$, the remaining control points are defined as,

$$\mathbf{c_1} = \left\{0, \frac{S}{2}, h\right\}^{\mathrm{T}} + \mathbf{c_0} \tag{3.15}$$

$$\mathbf{c_2} = \{0, S, h\}^{\mathrm{T}} + \mathbf{c_0} \tag{3.16}$$

since this motion occurs in the y and z axes, the values of a, b and c need to be estimated for cases, using the following expressions,

$$\{a_y, b_y, c_y\}^{\mathrm{T}} = \boldsymbol{M}^{-1}\{y_{c,0}, y_{c,1}, y_{c,2}\}^{\mathrm{T}} \tag{3.17}$$

$$\{a_z, b_z, c_z\}^{\mathrm{T}} = \boldsymbol{M}^{-1}\{z_{c,0}, z_{c,1}, z_{c,2}\}^{\mathrm{T}} \tag{3.18}$$

where $\boldsymbol{M}$ is defined as,

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0.25 & 0.25 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.19}$$

with the constants of each plane, it is possible to describe the limbs' motion, the result expression is the following,

$$f_2(t) = \begin{Bmatrix} x(t) \\ y(t) \\ z(t) \end{Bmatrix} = \begin{Bmatrix} x_{p,0} \\ a_y t^2 + b_y t + c_y \\ a_z t^2 + b_z t + c_z \end{Bmatrix} \tag{3.20}$$

For the Bezier trajectories, the generic form of a Bezier curve can be expressed as,

$$f_j(t) = \sum_{i=0}^{n}(1-t)^{n-i}t^i\mathbf{c}_\mathrm{i},\ j \in \{3,4\} \tag{3.21}$$

where n is the curve degree and t is the time interval for the trajectory. The main difference between the cubic and the fourth-degree curves is the number of control points. Assuming $\mathbf{c_0} = \mathbf{p}_{\mathrm{i},0}$, for the first case, the values of $\mathbf{c}_\mathrm{i}, i \in [1,3]$ are as follows,

$$\mathbf{c_1} = \left\{0, \frac{S}{4}, \frac{h}{3}\right\}^\mathrm{T} + \mathbf{c_0} \tag{3.22}$$

$$\mathbf{c_2} = \left\{0, 3\frac{S}{4}, \frac{h}{3}\right\}^\mathrm{T} + \mathbf{c_0} \tag{3.23}$$

$$\mathbf{c_3} = \{0, S, 0\}^\mathrm{T} + \mathbf{c_0} \tag{3.24}$$

As for the fourth-degree Bezier curve, the values of $\boldsymbol{c}_i, i \in [1,4]$ are the following,

$$\mathbf{c_1} = \left\{0, \frac{S}{4}, \frac{h}{3}\right\}^\mathrm{T} + \mathbf{c_0} \tag{3.25}$$

$$\mathbf{c_2} = \left\{0, \frac{S}{2}, h\right\}^\mathrm{T} + \mathbf{c_0} \tag{3.26}$$

$$\mathbf{c_3} = \left\{0, , 3\frac{S}{4}, \frac{h}{3}\right\}^\mathrm{T} + \mathbf{c_0} \tag{3.27}$$

$$\mathbf{c_4} = \{0, S, 0\}^\mathrm{T} + \mathbf{c_0} \tag{3.28}$$

For the cubic spline its necessary five control points $\mathbf{c_i} \in [0,4]$. Assuming that $\mathbf{c_0} = \mathbf{p}_{\mathrm{i},0}$, the remaining control points are expressed as,

$$\mathbf{c_1} = \left\{0, \frac{S}{4}, \frac{h}{1.35}\right\}^\mathrm{T} + \mathbf{c_0} \tag{3.29}$$

$$\mathbf{c_2} = \left\{0, \frac{S}{2}, h\right\}^T + \mathbf{c_0} \tag{3.30}$$

$$\mathbf{c_3} = \left\{0, 3\frac{S}{4}, \frac{h}{1.35}\right\}^T + \mathbf{c_0} \tag{3.31}$$

$$\mathbf{c_4} = \{0, S, 0\}^T + \mathbf{c_0} \tag{3.32}$$

in which $\mathbf{c}_{\boldsymbol{i},y} \in [0,4]$ and $\mathbf{c}_{\boldsymbol{i},z} \in [0,4]$ and using the *CubicSpline* function in Python, the cubic polynomial for each interval $[\mathbf{c}_{\boldsymbol{i},y,}\ \mathbf{c}_{\mathbf{i+1},\mathrm{y},}]$ can be found, which is defined as,

$$S_i(x) = a_i + b_i(x - \mathbf{c}_{\boldsymbol{i},y}) + c_i(x - \mathbf{c}_{\boldsymbol{i},y})^2 + d_i(x - \mathbf{c}_{\boldsymbol{i},y})^3 \tag{3.33}$$

where a_i, b_i, c_i, d_i are coefficients computed by the *CubicSpline* function. f_5 is defined as the group of $S_i(x)$ equations, and with that the limbs movement can be expressed as,

$$f_6(t) = \begin{Bmatrix} x(t) \\ y(t) \\ z(t) \end{Bmatrix} = \begin{Bmatrix} x_{p,0} \\ y_{p,0} + \frac{S}{T}t \\ f_5\left(y_{p,0} + \frac{S}{T}t\right) \end{Bmatrix} \tag{3.34}$$

3.3.3. Environments

A complete study requires that all algorithms are tested in different environments to know which algorithm is the most appropriate to apply in the posture adjustment. In order to have diversity in the terrains, four different setups where created, which are shown in Figure 3-5. The first terrain (Figure 3-5 (a)) works as a baseline for the values obtained in the latter environments. A slope environment (Figure 3-5 (c)) and two environments with obstacles (Figure 3-5 (b) and (d)), simple and complex, were used in order to give the robot different situations that can be found in the world.

Furthermore, the selection of these diverse environments aligns with the need for adaptability in robotic systems. Robots are often required to navigate and perform tasks in dynamic and unpredictable settings. Therefore, exposing the hexapod robot to scenarios like the slope and obstacle environments is not only a mean to test algorithm performance but also a reflection of the real-world challenges robots may encounter. The study's findings will not only inform about posture adjustment but also contribute to the broader field of robotics by enhancing the understanding of how algorithms can adapt to different terrains and obstacles, ultimately improving the robot's versatility and utility in practical applications.

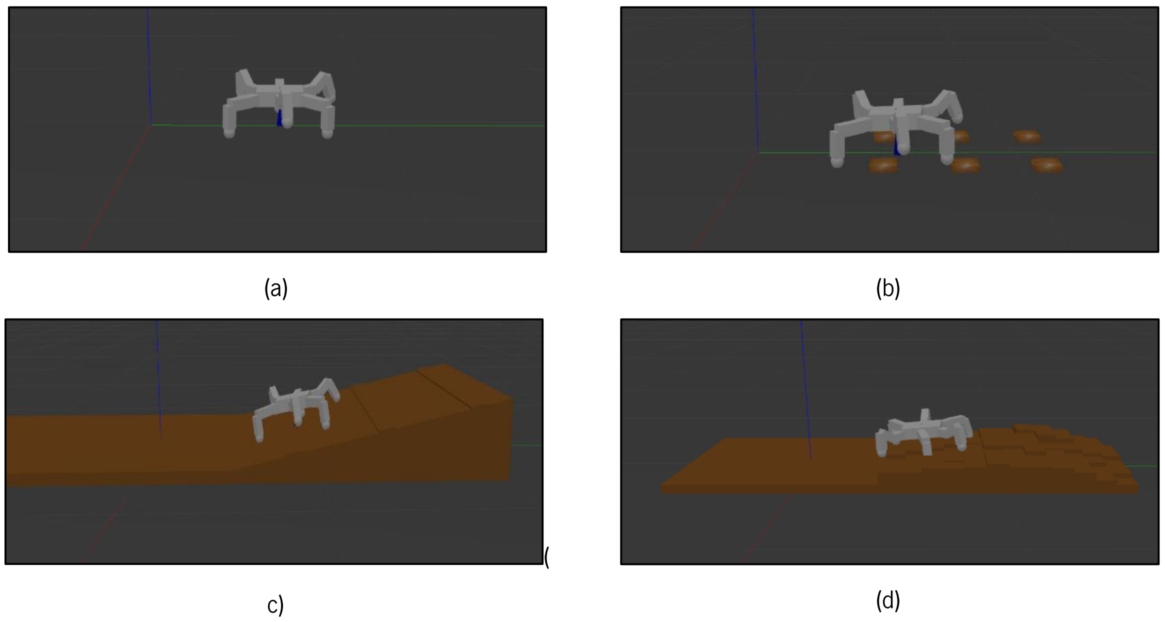


Figure 3-5 Environments tested in the computational simulations: (a) normal world, (b) normal world with steps, (c) ramp with 10-degree slope, (d) testbed.

3.4. SIMULATIONS

As presented in Swing Phase, five different gait algorithms were tested in all environments. Since energy efficiency is a topic of high importance in hexapod robots mobility the energy consumption was compared to find which algorithm had a higher reduction in terms of energy values. For that reason, the CoT was studied and compared throughout all environments. In each simulation, the robot took 20 steps using the tripod gait, at a velocity of 0.045 m/s. Each step was divided into 20 timesteps, during which the power consumption of all joints was extracted by multiplying the torque and the angular velocity. The CoT of each step was calculated by summing the power consumption values of all joints across all timesteps (see Performance Indices for the CoT equation). The presented results in Table 3-2 represent the average and standard deviation of three simulations for each gait algorithm and environment.

In an overall performance assessment across various environments, the 3rd and 4th degree Bézier curves consistently yield superior CoT results. The elevated CoT values observed in the triangular function are attributed to its less smooth trajectory compared to other algorithms. The other algorithms aim for smoother, curve-like paths, while the triangular function's trajectory lacks smoothness, which over-constraints the servomotors, resulting in trajectory errors and significant oscillations. In the normal world with steps and the testbed, there was a substantial increase in both mean and standard deviation values, with standard deviation occasionally surpassing the mean. This increase was linked to instances where the robot's standing legs would slip or fall due to obstacles or sudden balance loss, leading to an overexertion of the servomotors as the robot struggled to regain its balance. Given the rather favourable outcomes of the parabolic and 3rd-degree Spline algorithms, they remain viable options for gait generation.

Table 3-2 CoT Results for the simulations in all environments.

Algorithm / Environment	Triangular	Parabolic	3rd-bézier	4th-bézier	3rd-spline
Normal world	105.38 ± 58.79	3.12 ± 1.01	3.14 ± 1.02	2.60 ± 0.66	5.78 ± 2.76
Normal world with steps	104.73 ± 54.63	12.80 ± 16.47	13.45 ± 17.88	9.80 ± 8.45	17.84 ± 19.25
10° slope	79.99 ± 46.16	8.50 ± 4.27	7.28 ± 2.96	7.25 ± 2.44	10.23 ± 6.27
Testbed	116.29 ± 163.21	20.99 ± 33.23	16.82 ± 14.95	24.44 ± 25.28	21.28 ± 26.32

For further clarification the height variations, as well as the roll and pitch variations will be compared. This data is measured with an IMU applied to the virtual model of the robot.

To better classify the height variation, there will be two parameters: Height variance and distance to 0.10 m line. Figure 3-6 reveals that the triangular algorithm exhibits significant height variations, which also impact roll and pitch values. Consequently, it is determined that this algorithm is unsuitable for deployment on the hexapod robot. Among all other algorithms, the height variations are generally low. Notably, the 3rd-degree Bézier curve consistently maintains its height values above the 0.10 m threshold, reducing the risk of the robot's torso hitting the environment..

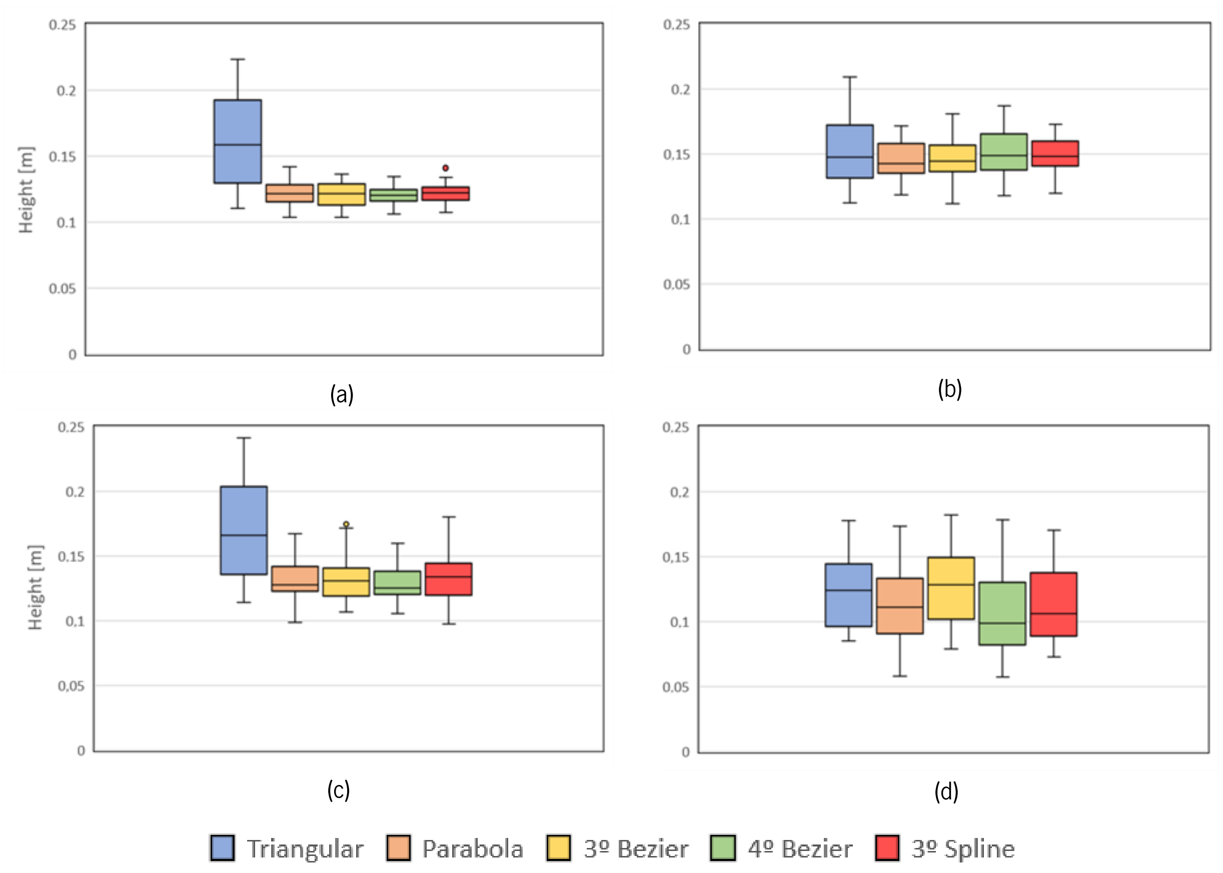


Figure 3-6 Box plots of height variation: (a) normal world, (b) normal world with steps, (c) 10° Slope, (d) testbed.

In terms of roll and pitch, greater stability is correlated with lower acceleration and velocity values, the velocity values are indicated in Figure 3-7 and Figure 3-8. Particularly, the parabola function consistently exhibits higher velocity value variations, leading to its exclusion as a suitable algorithm for application on the robot. In the context of this research, which emphasizes adaptation on inclined surfaces, the 3rd degree Spline is likewise unsuitable due to inferior performance compared to the 3rd and 4th degree Bézier curves (see Figure 3-7 (c) and Figure 3-8 (c)). Considering the better height values of the 3rd-degree Bézier curve, the similarity in roll and pitch values between both Bézier curves, and the increased computational cost associated with the 4th-degree Bézier curve due to the additional trajectory-defining point required, the chosen algorithm is the 3rd-degree Bézier curve.

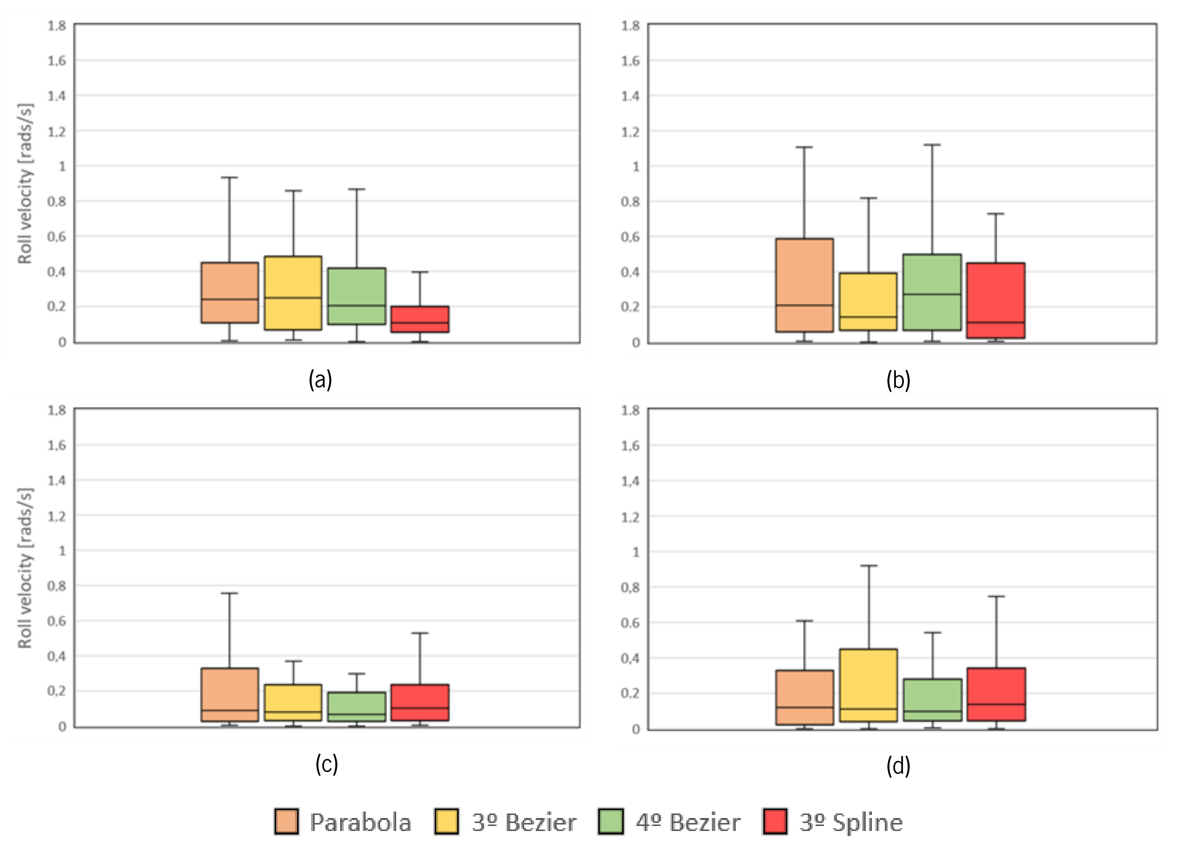


Figure 3-7 Box plot of roll velocity variation: (a) normal world, (b) normal world with steps, (c) 10° Slope, (d) testbed.

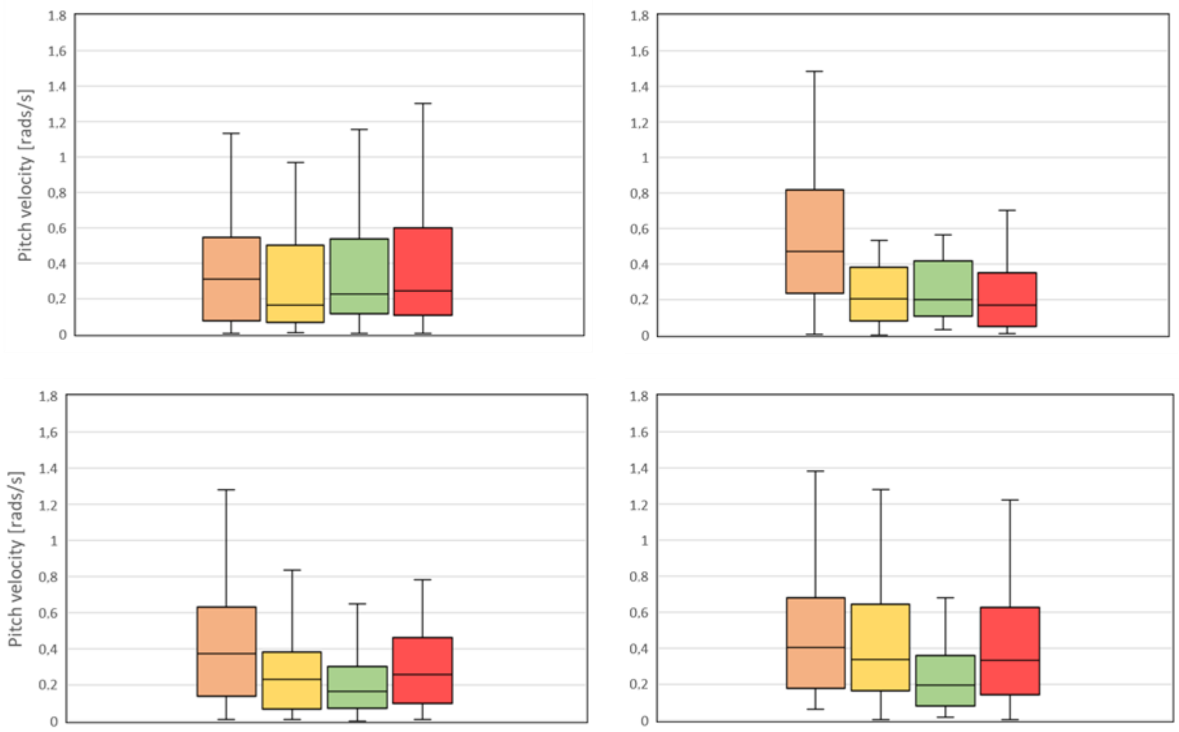


Figure 3-8 Box plots of pitch velocity variation: (a) normal world, (b) normal world with steps, (c) 10° Slope, (d) testbed.

3.5. Summary and conclusions

This chapter has provided a comprehensive exploration of the hexapod model ATHENA, focusing on its description, kinematic model, gait generation, and simulations. The description of the physical hexapod robot ATHENA, while not directly employed in the study, was included to facilitate a clear understanding of the system. Subsequently, the kinematic model was introduced as a critical element for controlling limb motion and, by extension, the robot's overall movement. This model established the crucial link between joint angular positions and the feet's coordinates, with the torso's center of mass serving as a reference point for all calculations. These foundational equations of motion and kinematic principles were identified as essential for posture control development.

The study then delved into gait generation, with a specific focus on how limb trajectories impact robot locomotion. The investigation highlighted the intricate relationship between foot velocity, acceleration, torso posture, and energy consumption during locomotion. To provide a comprehensive analysis, various algorithms were applied in four diverse environments: a normal flat surface, a flat surface with steps, a 10-degree sloped ramp, and a challenging terrain composed of uneven blocks referred to as the testbed. The results of this extensive examination revealed that the 3rd and 4th-degree Bézier curves consistently outperformed other algorithms in terms of CoT. The selection of the 3rd-degree Bézier curve as the preferred algorithm was driven by its superior height values, comparable roll and pitch velocity to the 4th-degree Bézier curve, and the associated computational efficiency advantages.

In conclusion, this chapter has laid the groundwork for understanding the ATHENA hexapod model, its kinematic principles, gait generation strategies, and the subsequent simulations. The insights gained from this chapter are instrumental in informing the subsequent chapters of this study, which will focus on posture control and adjustment.

4. Posture Adjustment

For a successful locomotion in rough terrain the main focus should be for the robot to have a correct posture adjustment. Timely corrections of the posture depending on the information of the robot's feet is a necessity to prevent crashes or inabilities to move forward. A correct assessment of the external information allied with a precise control are the key factors for an effective walking state.

In this chapter it is presented the sensors used to aid with the posture analysis, the posture control parameters which affect the height of the body, as well the roll and pitch inclination of the torso. An extensive study of the application of Reinforcement Learning, specifically Q-Learning, as a tool to find the best control parameters for each given inclined terrain. Also, the development of the code scheme that connects the *Gazebo* environment, the posture control and the Q-Learning algorithm is exposed.

4.1. Sensors

In order to aid with the assessment of the data obtained by the robot walking it was necessary to implement three types of sensors. Firstly, an infrared sensor was installed at the bottom of the torso to measure its distance to the ground, this was used to find a reference height while the robot walks in a plain ground and use it to adapt its walking while going through inclined terrain. At the feet, force sensors were fixed in order to aid in two tasks: verify contact with the ground and measure the reactions force from the ground to the feet so that it was possible to determine the effort applied to the joints of each leg, since a higher reaction force in the foot implies a stronger input from the servomotors at the joints in order to maintain its posture. Lastly, an IMU was installed at the torso to measure the robot's velocity, position and orientation with the intention to calculate the CoT. This IMU outputs the orientation in quaternions, the angular velocity and the linear acceleration in the three axes (X, Y and Z).

Although there is not a visible representation of the IMU sensor the other two sensors were represented in *Gazebo* (see Figure 4-1), and for the purpose of understanding the infrared sensor and the feet sensors a simulation in a normal plane was executed. In Figure 4-2 it is possible to ascertain the height displacement and the reaction forces applied to the foot 2 of the robot during a standard simulation in a normal plane environment.

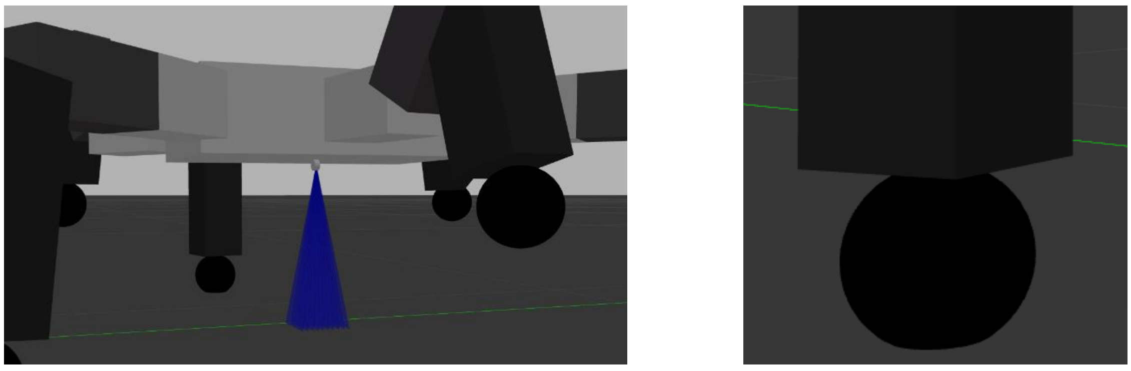


Figure 4-1 Sensors applied in the *Gazebo* environment: (a) infrared sensor, (b) force sensor.

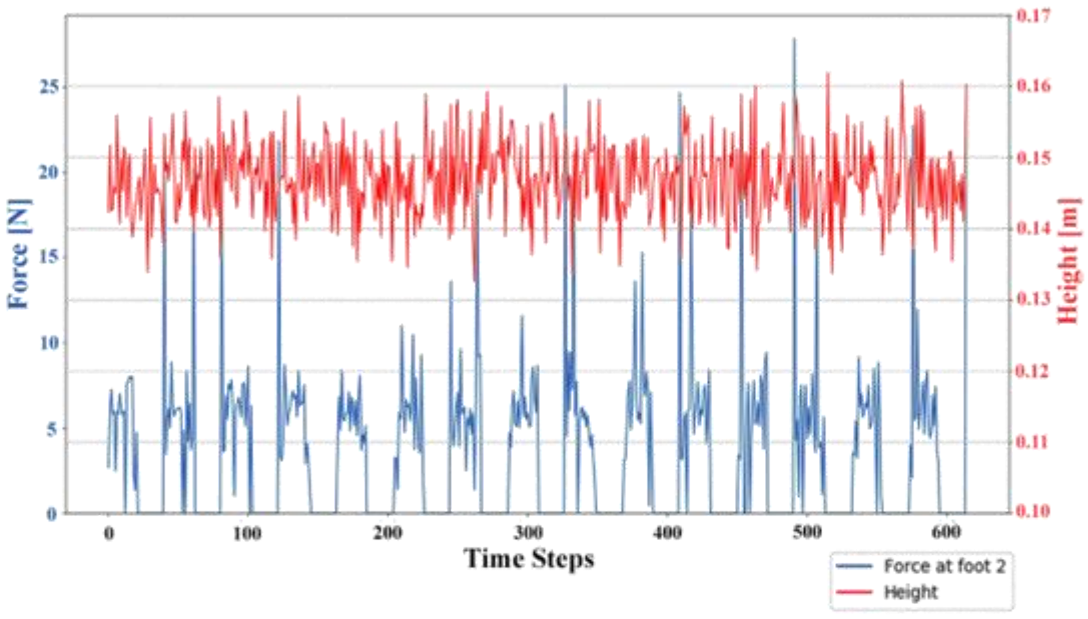


Figure 4-2 Values of height and forces applied at a foot of the robot in a normal plane.

4.2. Posture Assessment

4.2.1. Orientation Approximation

The change of terrain complexity increases the displacement of the torso's position, which limits the hexapod's ability to climb inclined surfaces. The posture control parameters aim at adjusting the hexapod's height and orientation by estimating the ground inclination.

As said in Sensors the hexapod's height displacement is measured by an infrared sensor placed on the torso to measure the distance to the ground and its value is compared to reference height of the robot while walking in a normal plane.

For the orientation, it is necessary to estimate the terrain perception by the relative position of the feet w.r.t the torso (see Figure 4-3). The relative coordinates of the feet in contact with the ground are evaluated by the control. With the relative coordinates of the supporting feet, the terrain can be approximated to a characteristic plane expressed as,

$$a_p \boldsymbol{x} + b_p \boldsymbol{y} + c_p \boldsymbol{z} + d_p = 0 \tag{4.1}$$

in which a_p, b_p, c_p and d_p are constants, and $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ represent the feet coordinates vectors. With the value of a_p, b_p, c_p and d_p the normal vector of the plane is obtained.

Each gait cycle has two sets of moving legs, each set of legs consists of the front and back leg of one side and the middle leg of the opposite side, so it's important to distinguish how the posture corrections will apply depending on each set. To simplify calculations, two sets of supporting legs were assigned to each set of moving legs, a variable, *t*, was created and for the first half of the cycle its value is zero and in the second half its value changes to one.

Although each foot has force sensors and it's possible to obtain the contact forces that prove that the leg is in contact with ground, for the case of climbing a ramp it was considered that the during the transposition of the ramp each set of legs is working in synchrony with the full gait cycle and the set represents the limbs that should be in contact with the ground. Considering the torso's local reference, and a two-dimensional representation of the normal vector, the angular displacement of the roll and pitch angles is determined as,

$$\delta_\alpha = arccos\left(\frac{g'^T n_\zeta}{\|g'\|\|n_\zeta\|}\right), g' = \{x_g, z_g\}^T \wedge n_\zeta = \{0,1\}^T \tag{4.2}$$

$$\delta_\beta = arccos\left(\frac{g'^T n_\zeta}{\|g'\|\|n_\zeta\|}\right), g' = \{y_g, z_g\}^T \tag{4.3}$$

where δ_α represents the roll displacement, n_ζ is a unitary vector, g' is a two dimensional representation of the normal vector and δ_β is the pitch displacement. The normal vector is a three-dimensional value and creating these two-dimensional representations allows to obtain the angular displacements in each plane, thus finding the pitch and roll angles.

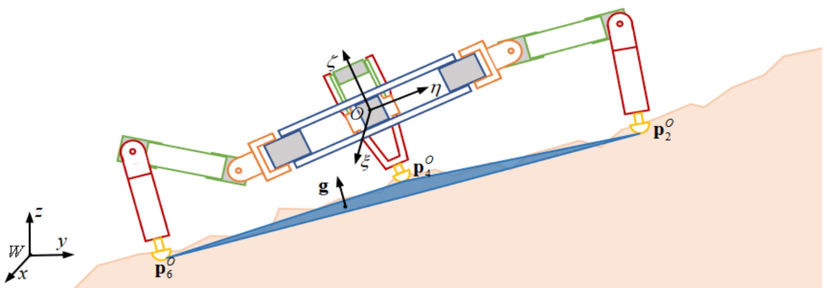


Figure 4-3 Schematic representation of the plane formed by the supporting limbs.

4.2.2. Control Parameters

After obtaining the values of the posture displacement, in terms of height and orientation the posture correction can be performed. The introduction of three control parameters: k_h for the height displacement, k_α for the roll displacement and k_β for the pitch displacement; provide the means to ensure a smooth gait phase transition. These parameters range from 0 to 1, 0 denotes that the control system does not influence the torso's height and orientation and 1 denotes that there is full implementation of the control system.

The height displacement is caused by the limbs' trajectory in the stance phase, so in order to successfully adjust its height, the control parameters are applied in the stance gait equations, provided in Stance Phase. Not only that, but since the swing phase it is influenced by the torso's height, changes were made at the final control point of the cubic Bézier trajectory used to calculate the swing movement. That final control point is defined as,

$$\boldsymbol{p}_{i,3}^{O} = \{0, S\,, k_h\delta_z\,\}^T + \boldsymbol{p}_{i,0}^{O},\ k_h \in [0,1] \tag{4.4}$$

where k_h is the height control parameter, δ_z is the height difference measured by the infrared sensor, being the difference between the height measured w.r.t the ground and the reference height, which is the torso's height while in standing posture, and $\boldsymbol{p}_{i,0}^{O}$ is the initial control point used to calculate the cubic Bézier trajectory w.r.t the reference O.

Regarding the stance phase corrections, the three control parameters are used to adjust and correct the displacement in height and orientation. The feet's final position in the stance phase is expressed as,

$$\boldsymbol{p}_{i,f}^{O} = \boldsymbol{t_r} + \boldsymbol{p}_{i,0}^{O} \tag{4.5}$$

in which $\boldsymbol{t_r}$ denotes the transformation vector, that is expressed by,

$$\boldsymbol{t_r} = R_x^{(4,4)}(k_\alpha\delta_\alpha)R_y^{(4,4)}(k_\beta\delta_\beta)T^{(4,4)}(k_h\delta_z), k_\alpha \in [0,1]\,, k_\beta \in [0,1] \tag{4.6}$$

where R_x and R_y are the rotation matrixes along the x and y axes respectively, k_α and k_β are the roll and pitch control parameters, and T is the transformation matrix, the matrixes can be defined as,

$$R_x^{(4,4)} = \begin{bmatrix} R_x^{(3,3)} & 0 \\ 0 & 1 \end{bmatrix} \tag{4.7}$$

$$R_y^{(4,4)} = \begin{bmatrix} R_y^{(3,3)} & 0 \\ 0 & 1 \end{bmatrix} \tag{4.8}$$

$$T^{(4,4)} = \begin{bmatrix} I^{(3,3)} & j \\ 0 & 1 \end{bmatrix}, j = \{0, -S, k_h\delta_z\}^T \tag{4.9}$$

4.3. Simulation Scheme

For single simulations to obtain reference values in any environment the procedure only involved initiating the *gd.launch* file, which started the *Gazebo* environment, the *simulation.py* that coordinated how long and how the simulation would execute and the *ros_client.py* that connected *Gazebo* and the *.py* file. After concluding all the files close and the data is stored in individual files. However, to ensure that during the *RL* phase there would not be any errors and that the data was stored successfully there was the need to develop a script in order to safeguard a correct execution of all the parts of the code.

In this section a representation of the simulation scheme is made, and a thoughtful analysis is executed in order to ascertain its functioning. The principal focus are the files that were utilized or changed in contemplation of a reliable and functional program. For that, five files will be exposed (see Figure 4-4): *main.sh*, *Q_Learning.py*, *iteration.sh*, *gd.launch* and *simulation.py*. To be noted that all .*sh* files are considered *bash* files that use *C* language, the .*launch* is a XML file and .*py* files use python language.

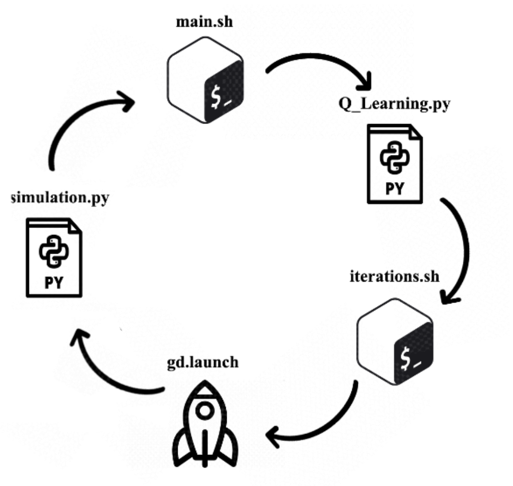


Figure 4-4 Developed simulation scheme.

4.3.1. Simulation Start

The *main.sh* file is executed in the terminal window and is executed to start running the simulations. Table 4-1 describes its functioning.

Table 4-1 Simulation start code depiction.

```
Algorithm 1: Starting the simulations
input: $i$ : iteration number, $iterations$ : total number of iterations
output: $i$ : iteration number
check for iteration number in iteration.txt
if $i \in$ iteration.txt
    $c\_i = i$
else
    $c\_i = 1$
end if
$iterations = 1000$
save iterations in total_ite.txt
for $c\_i \leftarrow 1, \ldots, iterations$ do
    initiate Q_learning.py file
    return $c\_i$ in iteration.txt
end for
if $c\_i = iterations$
    save $i = 1$ in iteration.txt
end if
```

In this code, the iteration number is extracted from a file in which the current iteration is saved, this is made in order to resume the code in case of a malfunction or there was not enough time to finish the full number of iterations. Other values were saved in additional files to give the possibility to resume the simulations without the need of starting over.

4.3.2. Q-Learning Application

This file reveals significant importance, since allows the algorithm to learn which value should be set in the control parameters. In this study, the focus was to determine the best control parameter for height adjustment. Since Q-Learning is a model-free type of RL and is a trial-and-error method some values need to be provided for its correct execution. Such values are the learning rate, discount factor, explore rate and the reward function. The correct use of these values aids to find the best results. Table 4-3 depicts its code.

The application of Q-Learning is divided into two steps. The first step contains the evaluation of the current iteration and extracting the data to update the Q-table. The second step, in which a *class* (named qLearning) was created to encapsulate the functioning of the Q-Learning algorithm and the update of the data for the following iterations. Figure 4-5 shows how the *class* qLearning is built. This *class* is comprised of 6 functions to support the inner workings of the Q-Learning algorithm.

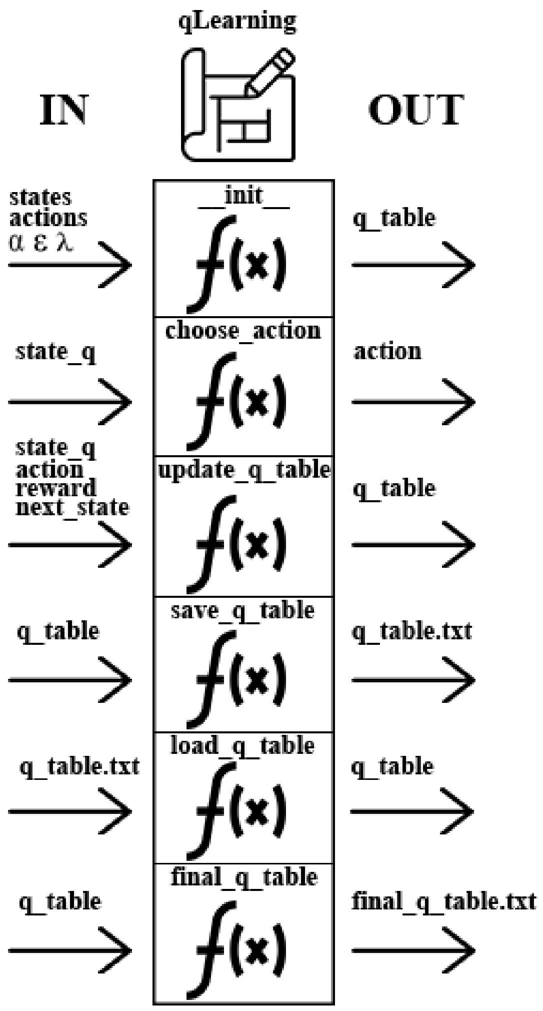


Figure 4-5 qLearning configuration structure.

The *__init__* function is automatically initiated when the *class* is called in code and creates the Q-table, The Q-table (see Table 4-2) in this study is designed to include 6 distinct actions that are parametrized by the k_h values, as well as 5 unique states that correspond to the different environmental configurations.

Table 4-2 Q-table developed in this research.

		States (Ramp inclination)				
		3 °	**6 °**	**9 °**	**12 °**	**15 °**
Actions (k_h values)	**0**	0	0	0	0	0
	0.2	0	0	0	0	0
	0.4	0	0	0	0	0
	0.6	0	0	0	0	0
	0.8	0	0	0	0	0
	1	0	0	0	0	0

The *choose_action* function uses the explore rate value to decide whether to explore or exploit the action to take. A random number is generated between 0 and 1 and in case that value is below the explore rate the algorithm will choose a random action to take, in this research the action is a random k_h parameter value to apply in the simulation. If the value is superior to that of the explore rate the action that is chosen is the one with the highest q-value. Regarding the states, its value depends on the variable *state_q* which will correspond to the coordinate in the vector *states* that include all five environments. For the first iteration the chosen action is $k_h = 0$, because all actions have a q-value of 0, regarding any environment.

With the *update_q_table* function the Q-table is dynamically updated during the reinforcement learning process. The update is performed using the values of *state_q*, *action*, being the action taken in the current iteration, *reward*, indicating the immediate reward received for that action, and *next_state*. Specifically, the Q-table is updated by modyfing the entry corresponding to the combination of *action* and *state_q* used in the current iteration.

Finally, the functions *save_q_table*, *load_q_table*, *final_q_table*, are used in order to pass information between iterations, the first being used to store the updates after an iteration and the second being utilized to load up the updates in the beginning of future iterations. The latter is applied in the final iteration to save the fully developed Q-table when the learning process is finished.

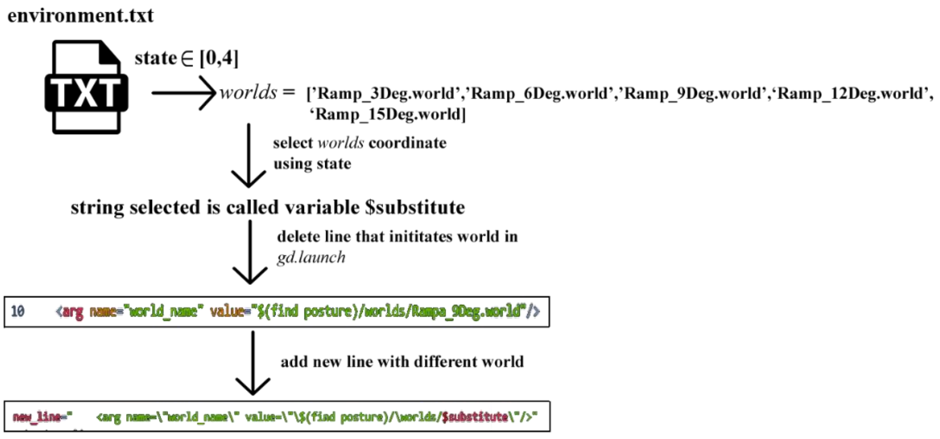
Table 4-3 *Q_learning.py* code scheme.

Algorithm 2: *Q-Learning* application

```
input: i: iteration number, iterations: total number of iterations, $\alpha$: learning rate, $\gamma$: discount factor, $\varepsilon$: explore rate, $k_h$: height control parameter, states: simulation environments, num_states: number of states, num_actions: number of actions
output: $r$: reward value
$k_h$ = { 0, 0.2, 0.4, 0.6, 0.8, 1}
states = {'3Deg', '6Deg', '9Deg', '12Deg', '15Deg'}
num_states = 5
num_actions = 6
check for iteration number in iteration.txt
state = random(1,5)
save state variable in environment.txt
if i = 1
    $k_h = 0$
    save $k_h$ in kh_value.json
    ep = i
    state_q = state – 1
    state_values = 0
    create Q-table
else
    ep = i
    state_q = state – 1
    extract state_values from state_value_update.json
    extract Q-table from q_table.txt
end if
epochs = 1000
for ep ← $1, \ldots,$ epochs do
    action ← apply choose_action function
    save action variable in kh_value.json
    old_state = state_q
    run iterations.sh file
    extract all update values from state_values_update.json
    extract reward from reward_value.json
    $next\_state$ = random(0,4)
    update ←apply update_q_table function
    save ← apply save_q_table function
    save Q-table in q_table.txt
    if ep = epochs
        final ← apply final_q_table function
        get best_action for each state
        save Q-table in final_q_table.txt
    end if
end for
```

4.3.3. Environment Selector

This *bash* file is used to alter the world in the *Gazebo* environment. In order for that change to be possible the *gd.launch* file needs to be manually switched to be able to input a new *Gazebo* world. Since it's not possible to manually modify the file after each iteration a solution was needed. *Iterations.sh* extracts a value from *environment.txt*, that was updated in *Q_learning.py*, that is used to select a coordinate in an array of strings (text data variable). This array contains the name of the files that constitute the worlds created for the simulations. After extracting the string variable from the array, the line in *gd.launch* in charge of launching the *Gazebo* world is deleted and then replaced with the same line but with the new array variable, this makes the *gd_launch* able to launch a different environment, see Figure 4-6 for schematic. Afterwards the *gd.launch* file is initiated.


environment.txt
TXT
state ∈ [0,4]
worlds = ['Ramp_3Deg.world','Ramp_6Deg.world','Ramp_9Deg.world','Ramp_12Deg.world', 'Ramp_15Deg.world]
select worlds coordinate using state
string selected is called variable $substitute
delete line that inititates world in gd.launch
add new line with different world

```
<arg name="world_name" value="$(find posture)/worlds/Rampa_9Deg.world"/>
```

```
new_line="   <arg name=\"world_name\" value=\"\$(find posture)/\worlds/$substitute\"/>"
```

Figure 4-6 *Gazebo* environment switching in *iterations.sh*.

4.3.4. Simulation Launcher

The *gd.launch* file is used for configuring and launching components within a ROS environment, in this case for a *Gazebo* simulation of a hexapod robot. It loads the robot's URDF model, and spawns it in the simulation. While doing that, it launches a node *athena* that runs a Python script, *simulation.py*. It contains several arguments that are used to configure the simulation environment, such as enabling GUI, whether the simulation runs without a graphical user interface, setting simulation time and specifying the URDF model file.

This *.launch* file can be split into five actions: the hexapod model configuration, the simulation environment, where a *.world* file is used to set up *Gazebo*, hexapod model spawn, where the robot's URDF description is generated from the Xacro model file and it spawns the hexapod model in the *Gazebo* simulation, hexapod control, that launches another file, *hexapod_control.launch,* that contains the configuration for controlling the hexapod robot, and the custom node, *athena*, that runs the python script, *simulation.py*, where all the orders and commands for the robot are located, and, not only that, all the data is generated.

4.3.5. Simulation Execution

Although *q_learning.py* considerable importance to this work, the bulk of the code resides in *simulation.py*. Since this file includes an extensive code documentation only the relevant parts will be mentioned. From the calculation of the feet and legs path to the reward system development, without this code arrangement it would not be possible to do the simulations. There are four important functions within this file that work in a cycle to make the robot move (see Figure 4-7). Those are: *step*, *move_joints*, *posture* and the *tripod* function.

In a simulation the robot does 15 steps, and in each step new footpaths are generated for the robot to succeed in walking until the end of simulation. The k_h value is exported from the *kh_value.json* that is created in *q_learning.py* and applied to the leg movements so that the height corrections are made in each new cycle. In the *step* function variables, all the necessary data to calculate the reward at the end of each simulation is saved and exported. The *move_joints* function generates the angle values calculated in *posture* and *tripod* and feed it to the ROS client to make the movement occur in the *Gazebo* environment.

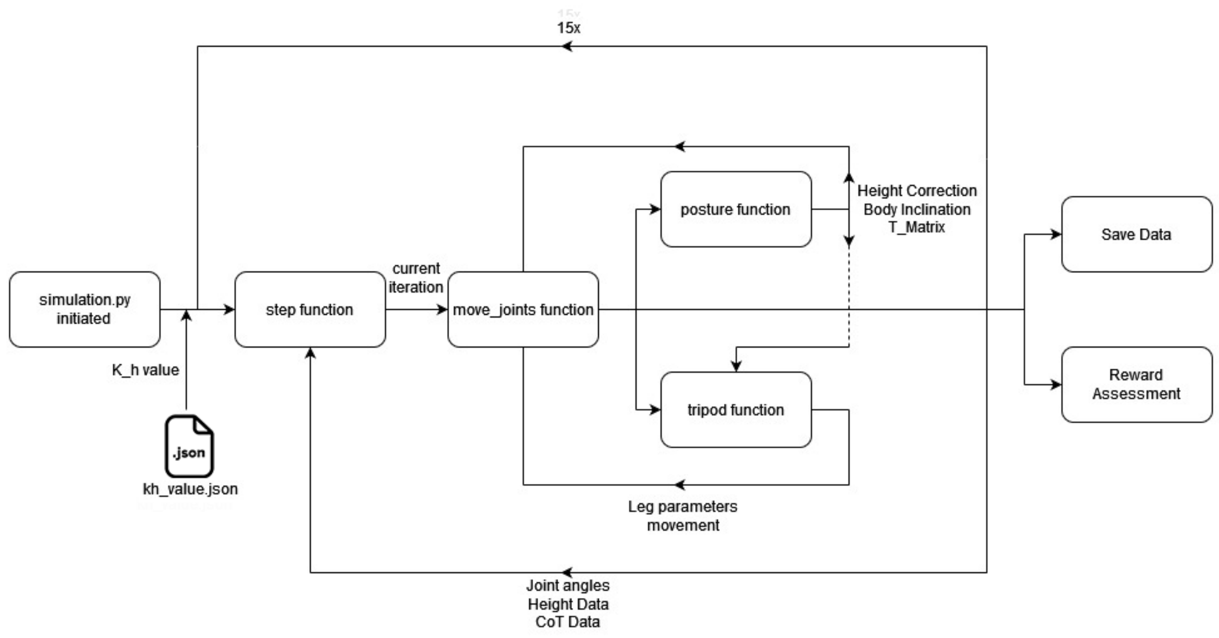


Figure 4-7 simulation.py main functions scheme.

The robot's walking is covered in Kinematic Model and Gait Generation, so a study on starting simulation conditions and ending simulation conditions is executed.

In order to feed the algorithm different conditions to better understand which is the best control parameter to apply, not only it was added different environments (check Environments), but also, two different starting positions were added to simulate a robot in a default standing position and a robot starting with the two front legs already on top of the ramp (Figure 4-8).

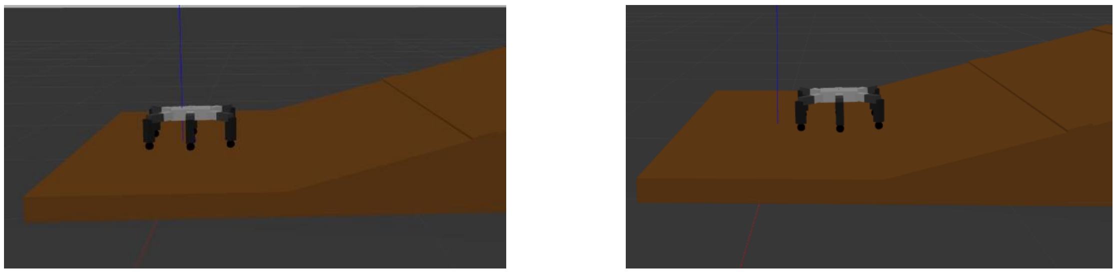


Figure 4-8 Robot's starting position: (a) normal standing position, (b) with two front legs in ramp.

This different starting position was achieved by generating a random number between 0 and 1, and any number that was generated above 0.5 would change the model initial position in the *y* axis, in the simulation world orientation, so that the front legs would be on top of the ramp. The goal was to add an imbalance in the robot's orientation and analyze situations in which the robot was not completely stable in the beginning of the simulation, giving the RL algorithm more inputs to achieve a better all-around understanding of the best control parameter to apply.

Regarding the conditions necessary to end the simulation, there were three possible scenarios: a normal simulation where the robot did all the 15 steps, the robot stops moving, which occurs when there is a disconnection between the ROS client and *Gazebo* and when the robot falls from the ramp.

Although the first scenario was sure to happen in all the simulations, there was a need to reassure that the other two possible scenarios would not crash the system and create a data loss. To ensure a safe shutdown of the program the robot's coordinates in the *Gazebo* world were registered and compared with limits that were set. In the case of the robot's not moving in two straight cycles a shutdown would be issued and for the possibility of the robot falling from the ramp at the end of each cycle the current height would be registered and if between two cycles the height difference had a sudden change the shutdown signal would be issued.

4.4. Reinforcement Learning

4.4.1. Reward Parameters Assessment

In order to ascertain the correct formula to apply the Reinforcement Learning algorithm the robot was tested in all the environments. Contact forces were retrieved during the whole simulations in order to obtain conclusions regarding the influence of the contact with the floor while walking slopes with different inclinations. The robot's feet were split in three different groups (Figure 4-9): the front legs (legs two and three), middle legs (legs one and four) and the back legs (legs five and six). This decision was based on the fact that the hexapod is walking with a tripod gait and by doing that it can be determined how the gait cycle and the ground influence different legs in the same position.

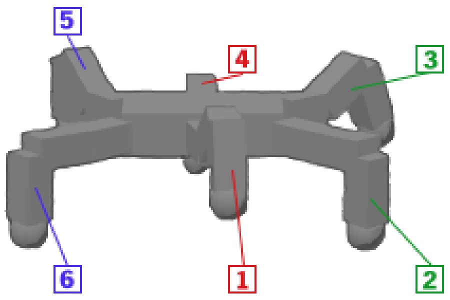
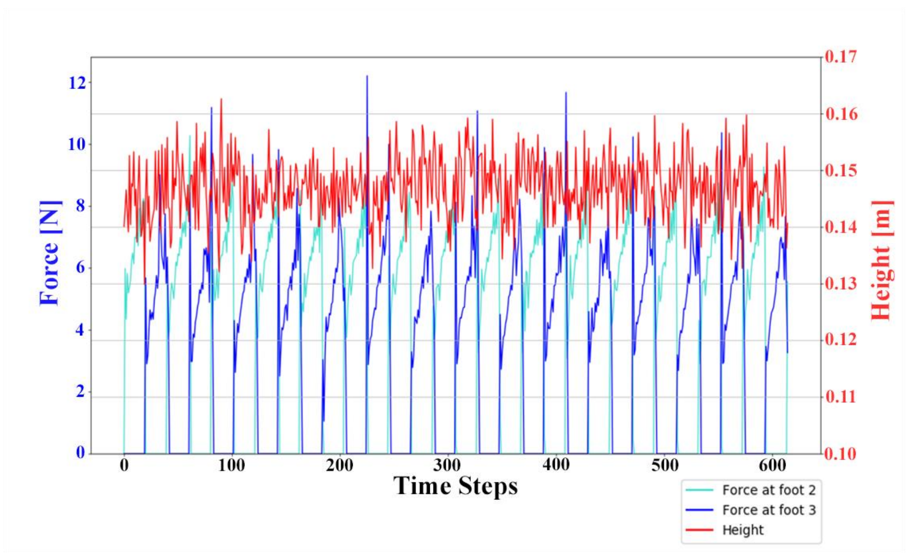


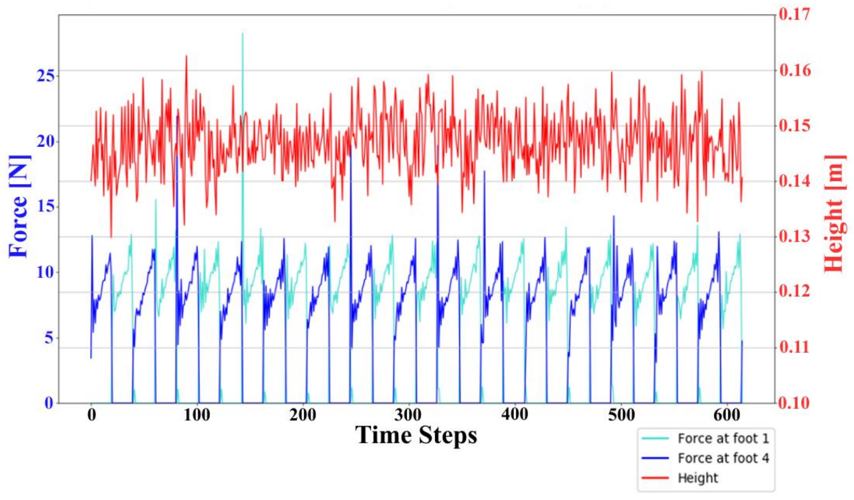
Figure 4-9 Groups of Hexapod robot Legs.

By extracting contact forces applied to each set of legs, it can be estimated how each slope influences each separate set of legs and then determine if the weight distribution and the Cost of Transport is an important subject to have in mind while developing the Reinforcement Learning algorithm in regard to the posture adjustment.

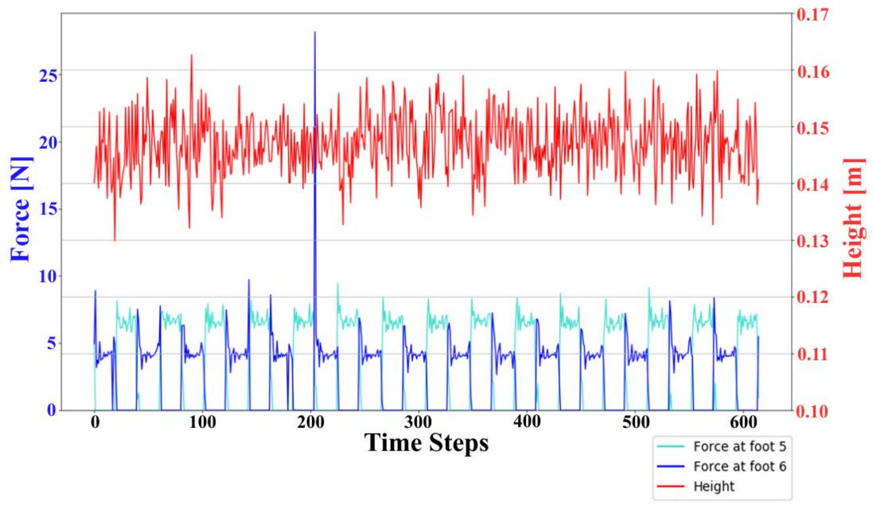
The reference simulation is the robot walking in a regular plain terrain. The front legs feet set had a median contact force of 6.55 N and 5.72 N, the first value being associated with the leg two and the latter to the leg three. Middle legs feet had a median contact force of 9.45 N and 9.08 N, these values being from leg one and leg four respectively. The back legs feet had a median contact force of 6.38 N and 4.78 N, for the leg five and the leg six correspondingly. Since the middle legs are closer to the center of mass of the robot it is logical that these will have a higher load applied during locomotion and the simulation results solidify this theory. Figure 4-10 shows the load on each foot of the robot during the locomotion in the regular terrain, as well as showing the height of the torso.



(a)



(b)



(c)

Figure 4-10 Forces applied to the feet in a regular environment: (a) front legs, (b) middle legs, (c) back legs.

Gathered the results from the reference simulation the robot is tested in five different slopes: 3°Slope, 6°Slope, 9°Slope, 12°Slope and 15°Slope. In Appendix A it is possible to consult the forces applied to each robot foot during locomotion in each slope.

There was a slight increase in the forces applied to the robot feet, mainly the middle and back feet, which is logical since climbing a terrain will require an increase in the forces applied from to robot to the ground in order to keep climbing. For the back legs, a trend of increasing mean contact forces associated with rising slope inclination was observed. However, the minimal increase in the forces applied to the feet deemed these values irrelevant to affect the reward function. Since, these values are directly proportional to the CoT there is no need to include it in the reward function. Consequently, the formulation of the reward function will focus solely on the robot's height during locomotion and its body inclination while ascending slopes.

Table 4-4 Mean contact forces in each leg, in all environments.

		Mean contact forces [N]					
		Front legs		**Middle legs**		**Back legs**	
		Leg two	Leg three	Leg one	Leg four	Leg five	Leg six
Slopes	**3 ° slope**	6.36	5.64	9.50	8.90	6.94	4.88
	6 ° slope	7.55	5.92	9.90	9.64	8.77	8.21
	9 ° slope	7.00	5.16	9.71	9.59	7.60	6.41
	12 ° slope	5.63	4.49	9.80	8.95	8.02	6.70
	15 ° slope	6.09	4.37	9.84	9.83	8.30	6.89

4.4.2. Reward algorithm development

The reward algorithm had some changes during this research and in this section. The choice to present this component before showing the simulation results resides in the fact that the reward function has a pivotal role that heavily influences the outcome of the experiments, and it must not be mixed with the learning values. The goal is to provide a comprehensive understanding of the adjustments that were employed to refine the reinforcement learning model.

The previous section concluded that the height and inclination values of the robot are the best to formulate the reward function, so that was the starting point. From the start of the formulation of the reward function there were four major changes to improve its functionality.

The first iteration of the reward algorithm stored at the end of each step the value of the height in a list and a median value of the body inclination was obtained. The reward equation is the following,

$$r_1 = -\sum_{i=0}^{n-1} \left\| H_{[i+1]} - H_{[i]} \right\| - \left(\left\| s - b_{deg} \right\| \times 2 \right) \tag{4.10}$$

$$r_2 = 10, if \ \left\| H_{[0]} - H_{[i]} \right\| \leq 1.5 \tag{4.11}$$

$$r_3 = -20, if \ d = 1 \tag{4.12}$$

$$r = r_1 + r_2 + r_3 \tag{4.13}$$

H being the list with all the height values registered at the end of each cycle, s the real slope inclination, b_{deg} the median value of the robot's body inclination and d being the variable that indicated if the robot had stop moving or fell of the ramp.

While doing this first set of simulations it was concluded that some changes were needed. It was noted that after the height change, from the normal ground to the inclined ground, the robot would have a minimal height variance in any case, since all legs were on the same plane again. That would influence the final results of the k_h parameter because there was unnecessary data being compared, so the height values should only be compared while the robot is adapting to the inclination change (see Figure 4-11). Thus, the reward function changed to,

$$r_1 = -\left(\sum_{i=0}^{n-1} \left\| H^{s}_{[i+1]} - H^{s}_{[i]} \right\| \times 1.5\right) - \left(\left\| s - b_{deg} \right\| \times 2 \right) \tag{4.14}$$

$$r_2 = 10, if \ \left\| H_{[0]} - H_{[i]} \right\| \leq 1.5 \tag{4.15}$$

$$r_3 = 20, if \ d = 1 \tag{4.16}$$

$$r = r_1 + r_2 + r_3 \tag{4.17}$$

H^s being the height values list when the robot is starting to climb the ramp, and to give more importance to these values, it was added a multiplication of 1.5.

In order to make it possible to extract only the values of height when the robot is climbing the ramp in each cycle the y coordinate of the robot was compared with the world coordinates to ensure that only the values within the limits imposed were extracted,

$$\sum_{i=0}^{n} H_{[i]} \in H^s, if \;\; B_y \in [0.35,0.9] \tag{4.18}$$

with B_y as the y world coordinate of the center of the body of the robot.

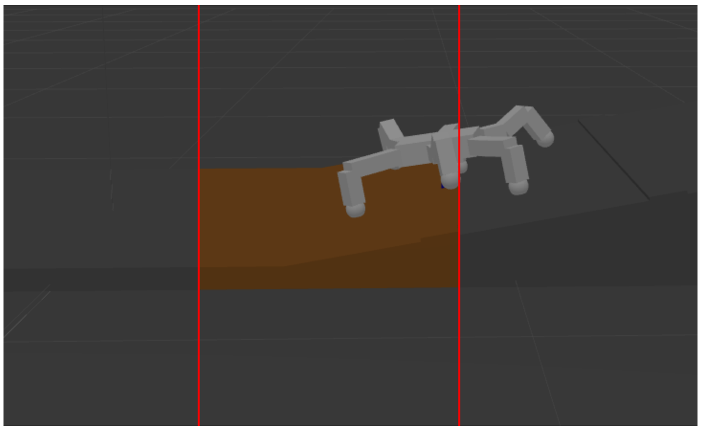


Figure 4-11 Height values extracted in the process of climbing the ramp.

Although these changes improved the results obtained in the simulations, they created a potential problem for the algorithm to distinguish a successful adaptation of the body inclination to each slope in the simulation. Thus, another improvement was made to reward it,

$$r_1 = -(\sum_{i=0}^{n-1} \left\| H^s_{[i+1]} - H^s_{[i]} \right\| \times 1.5) \tag{4.19}$$

$$r_2 = 10, if \;\; \left\| H_{[0]} - H_{[n]} \right\| \leq 1.5 \tag{4.20}$$

$$r_3 = 20, if \; d = 1 \tag{4.21}$$

$$r_4 = \begin{cases} 0, if \; s = 3 \\ 5, if \left\| s - b_{deg} \right\| \leq 1.5 \; \wedge s = 6 \\ 5, if \left\| s - b_{deg} \right\| \leq 2.2 \; \wedge s = 9 \\ 5, if \left\| s - b_{deg} \right\| \leq 2.8 \; \wedge s = 12 \\ 5, if \left\| s - b_{deg} \right\| \leq 3.3 \; \wedge s = 15 \\ -(\left\| s - b_{deg} \right\| \times 2), otherwise \end{cases} \tag{4.22}$$

$$r = r_1 + r_2 + r_3 + r_4 \tag{4.23}$$

The last change involved equations 4.20 and 4.22 to improve the chance of the algorithm finding the best results.

$$r_4 = \begin{cases} 0, if\ s = 3 \\ 5, if \left\| s - b_{deg} \right\| \leq 1\ \wedge s = 6 \\ 5, if \left\| s - b_{deg} \right\| \leq 1.5\ \wedge s = 9 \\ 5, if \left\| s - b_{deg} \right\| \leq 2.5\ \wedge s = 12 \\ 5, if \left\| s - b_{deg} \right\| \leq 3\ \wedge s = 15 \\ -\left(\left\| s - b_{deg} \right\| \times 2 \right), otherwise \end{cases} \tag{4.24}$$

$$r_2 = 10, if\ \left\| H_{[0]} - H_{[n]} \right\| \leq 0.6 \tag{4.25}$$

By applying these changes, the number of simulations that can reach these conditions is reduced, and the possibility to find the best control parameter for each slope inclination is further increased.

4.5. Simulations

The objective of the simulations is to find the best height control parameter for any slope. For that goal, five different environments with different slopes were created so the algorithm could train and discover which is the best option. In each simulation the robot starts either before the ramp or with the front legs already on top of it, the simulation runs for 15 steps and then ends and saves the data. For the purpose of the RL approach each set of simulations consisted of 1000 episodes. To conduct the experiments described in this dissertation an Intel Core i7, 9th generation @2.60Ghz and 8.0 GB RAM capacity was used. The operating system is Ubuntu 20.04 LTS, and the command window was used to launch the simulations, that ran on *Gazebo*. Each set of simulations had a real time duration of 17 hours.

The RL approach had three steps, namely, to find the correct parameters for the reward algorithm (see Reward algorithm development), the tuning of reward algorithm parameters and tuning of the hyperparameters. The last step compared results from simulations with the same reward algorithm and different hyperparameters values to find the optimal solution.

In this section, is presented the training procedure, the hyperparameters applied and the results of said application. The reward values, as well as the Q-values are compared throughout the process to understand the changes that led to the final RL algorithm and simulation results. Additionally, height, roll and pitch data were extracted to analyse how the process improved.

4.5.1. Results

In addition to the reward algorithm, the hyperparameters influence how the agent acts during the simulations. The learning rate (α) sets the size of steps taken by the learning algorithm when adjusting its model to minimize errors, which affects the convergence speed, the discount factor (γ) determines the importance of future rewards in the agent's decision making process and the explore rate (ε) defines how often an agent chooses random actions over the best-known actions. Table 4-5 presents the combination of hyperparameters used in each simulation.

Table 4-5 Simulations hyperparameters.

Step	Simulation number	α	γ	ε
1	1	0.7	0.9	0.2
	2	0.7	0.9	0.2
2	3	0.5	0.9	0.1
3	4	0.8	0.7	0.3
	5	0.8	0.95	0.3
	6	0.8	0.99	0.3

The first step included the research for the correct reward algorithm parameters. Nonetheless, the results from these simulations aided in finding which were the most efficient hyperparameters while working with the RL algorithm. Comparing equations 4.10 and 4.14 it is predicted that the first simulation converges above the second simulation since the latter had an increase in the negative reward regarding height changes. Figure 4-12 proves that prediction. Since the first simulation had a reward algorithm that was not objective oriented towards the goal, the second simulation will work as a baseline to the following simulations. All the simulations are compared in terms of the accumulated reward function convergence, as well as the height, roll and pitch velocities values (see Table 4-6). These values are the average extracted from the end of each episode (1000 episodes) that made each simulation. Since all simulations after the second have a reward algorithm that is objective oriented towards the goal, a higher convergence value is associated with better simulation results.

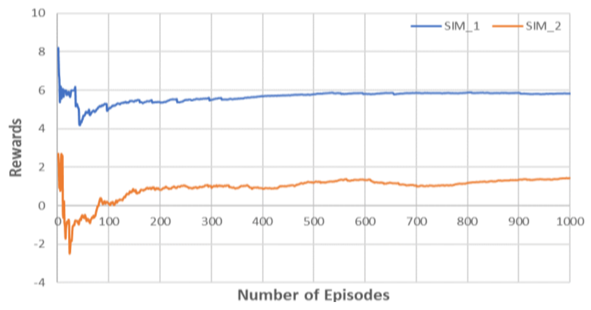


Figure 4-12 SIM_1 and SIM_2 accumulated reward functions.

Table 4-6 Baseline Values for height, roll and pitch.

Parameters	**Values**
Height	0.1450±0.00624 m
Roll velocity	0.3821±0.1258 rad/s
Pitch velocity	0.4063±0.2169 rad/s

In each simulation, at every 100 episodes, it was extracted key values to further understand how the learning evolved. By inspecting Table 4-7, a trend to obtain higher rewards is noted as the episodes increase. When comparing episode 199 and 299, where the environment and control parameter value were the same it is possible to see an increase in the reward and the approximation of the body inclination to the slope inclination. Although some data doesn’t abide by that tendency that is justifiable by the explore input, which allows the agent to seek random combinations in disregard to the exploit input, that always execute the action with higher Q-value. These random combinations can help finding a possible optimal solution that the current agent and reward system had disregarded earlier on the simulations. For the future simulations the key values extracted at each 100 episodes are going to be exposed in Appendix B.

Table 4-7 Key values at each 100 episodes in SIM_2.

Episode	**Reward**	k_h	**Height** [m]	**Roll velocity** [rads/s]	**Pitch velocity** [rads/s]	b_{deg}	**Real slope**
99	–4.692	0.2	0.1398	0.2802	0.2190	4.68°	12°
199	–2.083	0.6	0.1473	0.3619	0.4248	6.03°	12°
299	–0.322	0.6	0.1458	0.4622	0.4078	6.89°	12°
399	+1.306	0.8	0.1525	0.4348	0.3872	4.70°	9°
499	+2.461	0.6	0.1492	0.5395	0.7764	5.29°	9°
599	–1.525	0.4	0.1519	0.3628	0.3145	6.28°	12°
699	–5.516	0.8	0.1449	0.5792	0.5600	7.42°	15°
799	+7.120	0.8	0.1522	0.3313	0.2616	4.6°	6°
899	+8.685	0.4	0.1418	0.3585	0.2734	2.39°	3°
999	+7.213	0.0	0.1378	0.2972	0.2021	4.63°	6°

The final Q-table at the end of the second simulation can be analyzed in Table 4-8. The q-values with a green shade show the optimal control parameter for each slope. In this simulation there is already a bias to choose $k_h > 0$ in most of the environments, except in the 3° Slope, which is understandable since it is an environment similar to a plane terrain. The control parameters that are optimal in this simulation are the following:

- 3°Slope: The best control parameter to apply is $\boldsymbol{k_h} = 0.0$.
- 6°Slope: $\boldsymbol{k_h} = 0.8$.
- 9°Slope: $\boldsymbol{k_h} = 0.6$.
- 12°Slope: $\boldsymbol{k_h} = 0.2$.
- 15°Slope: $\boldsymbol{k_h} = 0.8$.

Table 4-8 Final Q-table of Sim_2.

		States (Ramp inclination)				
		3 °	**6 °**	**9 °**	**12 °**	**15 °**
ACtions ($\boldsymbol{k_h}$ values)	**0**	+39.27	+26.30	+11.46	+4.53	+14.41
	0.2	+27.40	+22.25	+22.60	+23.91	+13.13
	0.4	+28.40	+30.64	+18.52	+18.01	+15.35
	0.6	+25.53	+21.30	+30.53	+17.00	+7.21
	0.8	+29.22	+30.66	+22.12	+9.43	+17.76
	1	+29.52	+19.47	+19.4	−16.76	+8.32

The second step compared the results of the second and third simulation. It was applied the same reward algorithm and different hyperparameters. The third simulation converged earlier and higher which proves that it achieved a better solution compared with the second simulation (see Figure 4-13). The final Q-table generated in the third simulation reached the following conclusions:

- 3°Slope: The best control parameter to apply is $\boldsymbol{k_h} = 0.2$.
- 6°Slope: $\boldsymbol{k_h} = 0.0$.
- 9°Slope: $\boldsymbol{k_h} = 0.4$.
- 12°Slope: $\boldsymbol{k_h} = 0.2$.
- 15°Slope: $\boldsymbol{k_h} = 0.4$.

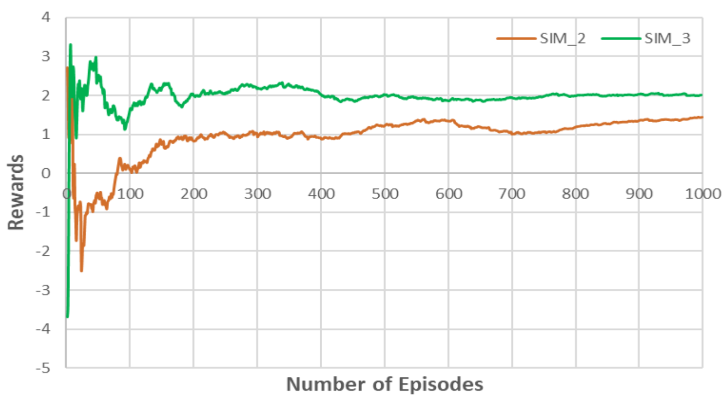


Figure 4-13 SIM_2 and SIM_3 accumulated reward functions.

Table 4-9 Final Q-table of SIM_3.

		States (Ramp inclination)				
		3 °	**6 °**	**9 °**	**12 °**	**15 °**
ACtions (k_h values)	**0**	+29.32	+27.42	+19.69	+8.56	+3.61
	0.2	+36.98	+25.25	+12.63	+16.71	+7.39
	0.4	+27.73.	+26.84	+32.21	−2.45	+21.94
	0.6	+20.87	+18.39	+12.26	+14.65	+7.25
	0.8	+27.74	+22.30	+20.55	+6.88	+8.81
	1	+19.87	+7.08	+20.77	+12.80	+3.20

The software *Gazebo* couldn't assure the same specific height value in the standing posture of the hexapod in each simulation, so it was deemed necessary to compare the standard deviation of the height variations. To compare the height values between the simulations it was applied the coefficient of variation. The second simulation had a coefficient of variation of 4.31% and the third simulation had 4.06%, concerning the height variations. Regarding roll and pitch average values there was a decrease in the third simulation of 3.06% and 15.32%, respectively. Roll and pitch standard deviation values reported a decrease of 3,07% and 18,80%, correspondingly. Table 4-10 shows the SIM_3 values.

Table 4-10 SIM_3 Values for height, roll and pitch.

Parameters	**Values**
Height	0.1437±0.00584 m
Roll velocity	0.3708±0.1221 rad/s
Pitch velocity	0.3523±0.1826 rad/s

The transition to the third step compares the third simulation with the remaining simulations. The latter simulations apply the last iteration of the reward algorithm (check Reward algorithm development), so in terms of goal objective they are more accurate than the reward algorithm in the third simulation. These latter simulations have the same learning rate and explore rate, but the discount factor has different values in order to find the one that has a bigger reduction of the noise, which is random variations that affect the total reward signal that can influence the agent learning. Standard RL simulations have a discount factor≥0.9, the fourth simulation had a discount factor of 0.7 which proved to be an error since it had a worse performance than the third simulation even while having a more accurate reward algorithm, which suggests that the accumulated reward function convergence is not solely dependent on the accuracy of the reward algorithm, so the data generated from this simulation can be disregarded. In Figure 4-14 it is possible to conclude that the fifth and sixth simulations achieved higher convergence values than the third simulation, indicating greater reliability in their final outcomes. Also, a closer analysis of their q-values (see Table 4-11 and Table 4-12) revealed that the discount factor significantly influences the evolution of the q-values, with a higher discount factor emphasizing future rewards, which in turn affects the growth of the q-values.

Overwriting the results from the third simulation with the fifth simulation results, the optimal control parameters for the height for each slope are:

- 3°Slope: The best control parameter to apply is $\boldsymbol{k_h} = 0.4$.
- 6°Slope: $\boldsymbol{k_h} = 0.6$.
- 9°Slope: $\boldsymbol{k_h} = 0.8$.
- 12°Slope: $\boldsymbol{k_h} = 0.8$.
- 15°Slope: $\boldsymbol{k_h} = 0.8$.

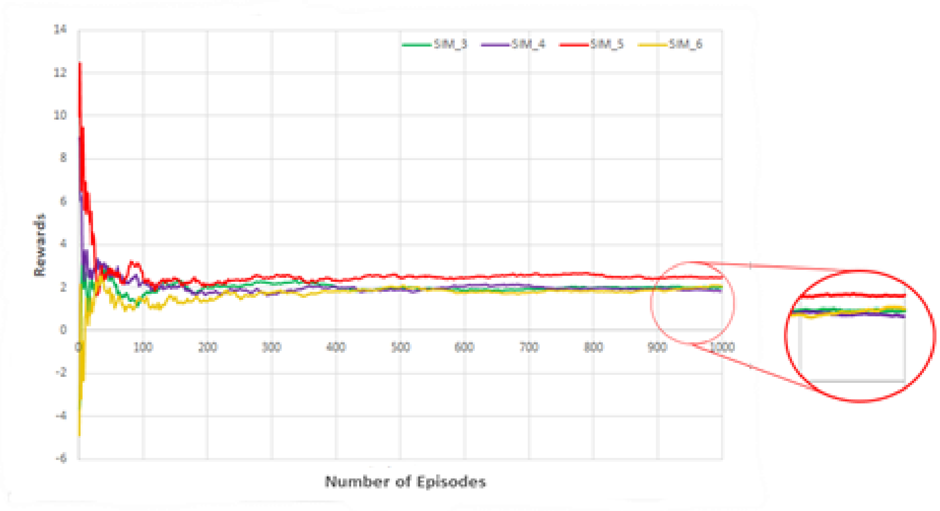


Figure 4-14 SIM_3, SIM_4, SIM_5 and SIM_6 accumulated reward functions.

Table 4-11 Final Q-table of SIM_5.

		States (Ramp inclination)				
		3 °	**6 °**	**9 °**	**12 °**	**15 °**
ACtions (k_h values)	**0**	+116.94	+110.42	+91.42	+82.24	+89.94
	0.2	+106.63	+112.91	+92.84	+92.50	+94.23
	0.4	+118.34.	+115.70	+97.04	+93.58	+93.51
	0.6	+102.91	+121.14	+111.21	+95.20	+92.42
	0.8	+110.53.	+111.65	+111.72	+106.66	+98.45
	1	+102.57.	+92.71	+100.19	+73.84	+92.36

Table 4-12 Final Q-table of SIM_6.

		States (Ramp inclination)				
		3 °	**6 °**	**9 °**	**12 °**	**15 °**
ACtions (k_h values)	**0**	+275.68	+264.86	+260.96	+257.62	+246.31
	0.2	+287.03	+272.33	+274.33	+270.17	+215.66
	0.4	+252.14.	+294.52	+286.93	+264.64	+287.21
	0.6	+301.85	+253.58	+288.32	+268.65	+269.91
	0.8	+201.39.	+219.51	+257.72	+286.89	+241.05
	1	+274.27.	+175.54	+279.87	+272.66	+260.55

Given the superior results observed in the third simulation compared to the baseline values, the performance of the third simulation with that of the fifth and sixth simulations were compared(see Table 4-13). Regarding the height standard deviation values, the coefficient of variation is 4.21% and 4.73%, for the fifth and sixth simulation, respectively, which compared to the third simulation has a slight increase. Concerning the roll average and standard deviation values for the fifth simulation there was a decrease of −1.48% and an increase of 3.63%, correspondingly. For the pitch average and standard deviation, it was registered an increase of 11.68% and 18.55%, individually. As for the sixth simulation, for the roll average and standard deviation values there was an increase of 1.98% and 5.50%, respectively. In the case of the pitch average and standard deviation there was also an increase of 18.28% and 28.22%.

Although, when compared with the third simulation, the key values present an increase in their values, these cannot be used as a fixed measure of the effectiveness of the simulations. Considering that the optimal control parameters discovered in the third simulation were all k_h<0.4 and for the fifth and sixth simulations the optimal control parameters were all k_h>0.4 it is possible to understand why the key values were higher. From the observed behaviour of ATHENA throughout the computational simulations, higher values of k_h can lead to heavier oscillations. This is justified by the over-compensation of the feet trajectory, which increases their velocity and consequently the friction forces during the collision with the ground. The correction of the feet motion requires more energy consumption to a non-adaptive behaviour ($k_h = 0$). Bearing this in mind, the increase of k_h causes posture instabilities, and although it makes it possible for the robot to overcome the higher slope ramps it causes a higher height disruption. This higher control parameter value has an increased influence in correcting the robot's trajectory, because of that an increased effort is put on the servomotors, which by consequence causes more oscillations while walking.

Table 4-13 SIM_5 and SIM_6 values for height, roll and pitch.

Parameters	**Sim_5**	**Sim_6**
Height	0.1447±0.00609 m	0.1448±0.00685 m
Roll velocity	0.3654±0.1267 rad/s	0.3783±0.1292 rad/s
Pitch velocity	0.3989±0.2242 rad/s	0.4311±0.2544 rad/s

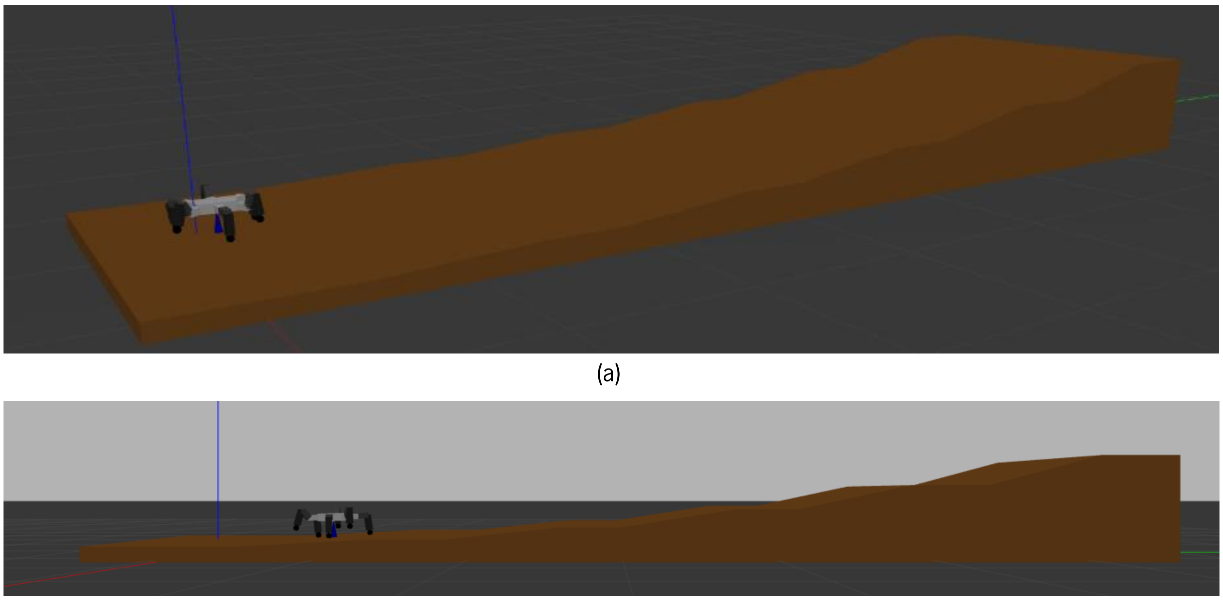
4.5.2. Final Results

To evaluate the research's accomplishments, a concluding simulation is conducted, contrasting a hexapod robot's navigation in a novel, dynamically changing environment (see Figure 4-15) with and without control interference. Building upon insights from the preceding sub-chapter, the robot showcases its autonomous adaptation of optimal control parameters when traversing a varied terrain composed of inclined surfaces with diverse inclinations.

The simulation without the application of height control parameters encountered failure, halting at a 12° slope as the robot's height dipped below the critical 0.10 m threshold (see Figure 4-16), indicating an unsafe operational state close to collision. An examination of the height values evolution within the control-free simulation, reveals that while the robot navigates lower-degree slopes effectively, it struggles to adapt on steeper inclines, leading to a gradual loss of height until an eventual collision with the environment becomes inevitable.

On the contrary, the simulation with applied control parameters adeptly adjusted to varying slopes, achieving successful completion. Despite suffering significant oscillations, even exceeding those observed in the control-free simulation (as evident in Table 4-14), the primary objective of enabling a hexapod robot to navigate inclined surfaces was attained. An examination of the height value progression in the controlled simulation (see Figure 4-16) highlights the correlation between oscillations and alterations in the k_h parameter whenever the terrain slope changes. It is evident that a sudden modification in the control parameter triggers a transitional phase during which the robot recalibrates its gait algorithms to accommodate the newly applied parameter, giving rise to heightened oscillations in the process. Nonetheless, the robot's successful completion of the simulation, proves that despite needing corrections in the process of re-adjusting the control parameter it is still capable of transposing a complex environment with diverse slopes.

Comparing the coefficient of variation for height values, the simulations without control, the simulation with control values in the same step where the simulation without control ended, and throughout the entire controlled simulation yielded percentages of 9.64%, 8.25%, and 9.99%, respectively. In terms of roll, the average and standard deviation increased by 75.75% and 13.38%, correspondingly, when transitioning from the control-free simulation to the controlled one. Similarly, the pitch parameters saw an increase of 111.33% in the average and 107.43% in the standard deviation with control. These substantial increments do not disregard the successful and safe traversal of the environment by the robot with applied control, which is the research's core objective.


(a)

(b)
Figure 4-15 World with all slopes: (a) top view, (b) side view.

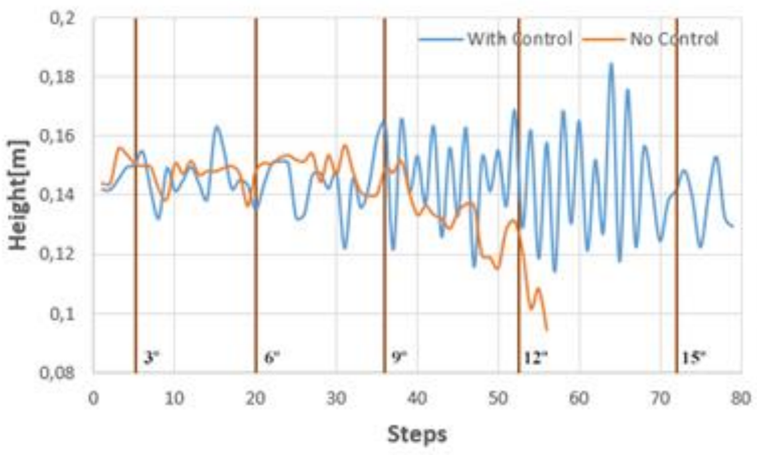


Figure 4-16 Height Variation for the robot with and without control.

Table 4-14 Height, Roll and Pitch values for the final simulation.

Parameters	**No control (56 steps)**	**Control (56 steps)**	**Control (80 steps)**
Height	0.1406±0.0136 m	0.1449±0.0120 m	0.1438±0.0144 m
Roll velocity	0.2081±0.3136 rad/s	0.3735±0.3744 rad/s	0.3657±0.3556 rad/s
Pitch velocity	0.2277±0.2917 rad/s	0.3661±0.4483 rad/s	0.4812±0.6051 rad/s

4.6. Summary and Conclusions

In summary, this chapter comprehensively explored posture adjustment, a fundamental element in enhancing the hexapod robot's adaptability to diverse terrains. Five sub-chapters have been covered, each contributing to an all-around understanding of the posture adjustment process.

Firstly, the integration of three key sensors that significantly enhanced data assessment during the robot's locomotion. An infrared sensor at the torso's base provided reference height information for adapting to inclined terrain, while force sensors at the feet validated ground contact and gauged reaction forces to assess joint effort. An IMU captured velocity, position, and orientation, enabling CoT calculation.

Secondly, correct posture assessment centers on accurate orientation estimation, which is derived from the relative coordinates of the feet. These orientation values enable us to calculate the pitch and roll displacement of the robot's body relative to the ground plane. This displacement data is essential for adapting gait algorithms to account for changes in terrain, ensuring successful navigation in varying terrain conditions.

Thirdly, the creation of a simulation scheme within the context of this research provides a foundation for future enhancements and extensions by establishing an adaptable framework for the Q-Learning approach. This framework enables easy integration of new studies to obtain results involving the control parameters for roll and pitch displacement. Nonetheless, adding more parameters in a Reinforcement Learning problem will demand a higher number of simulations, necessitating substantial computational power for efficient and rapid results. Furthermore, the scheme streamlines the gathering of variables, automatic environmental modifications and data storage.

Fourthly, the critical role of the reward algorithm in the success of an RL algorithm is emphasized. Precise, goal-oriented reward parameters are essential for productive outcomes. Through simulating robot walks on various slopes, the reward algorithm can omit CoT and focus on body inclination and height variations when transitioning between slopes.

Fifthly, the application of Q-Learning to find the optimal height control parameters for each individual slope. In addition to the reward algorithm, fine-tuning hyperparameters is crucial for achieving satisfactory and optimal results. The core objective of the research was successfully achieved: the robot, initially unable to navigate a sloped environment without control assistance, became capable of doing so through the application of height control parameters. However, this caused increased oscillations, attributed to abrupt control parameter changes with varying slope inclinations and the fact that higher k_h values result in posture instability. A smoother transition for k_h shifts can likely resolve the first issue. Addressing the second problem, future research on roll and pitch displacement control parameters can mitigate the amplified oscillations associated with higher k_h values.

In conclusion, the findings and developments presented in this chapter have yielded positive results, signifying a significant step in enhancing the hexapod robot's adaptability to inclined terrains. The comprehensive exploration of posture adjustment, integration of advanced sensors, simulation frameworks, and the optimization of control parameters has provided a solid foundation. This project's success is a testament to its adaptability and readiness for future research. Built to accommodate further additions and increased complexity, it serves as a promising platform for ongoing research and innovation in the field.

5. Final Considerations

The final considerations are included in the parts that follow. The description of the primary conclusions for the current work, as well as the prospects and future work based on what has been done and what can be done, ranging from improvements to new applications.

5.1. Conclusions

Throughout the present dissertation, conclusions were presented whenever it was convenient, however, a general overview of the most significant ones is provided here. Regarding the objectives identified in section 1.2, it is demonstrated throughout this dissertation that all of them were attained.

The main objective of the present dissertation was to correctly adapt a hexapod robot posture while transversing inclined surfaces. It can be considered accomplished as the simulations results are presented and their analysis correspond to a tuning of control parameters that adjust accordingly to the inclination of each simulation. The conclusions drawn from this accomplishment are presented below.

Regarding the gait generation, it was noted that the limbs' trajectory had a notably strong influence on the robot's posture. Although the feet were not in contact with the ground during most of the swing phase, their velocity and acceleration influenced the torso posture and the CoT. As a result, a comprehensive analysis was conducted, encompassing five gait algorithms and four diverse environments. 3rd and 4th-degree Bézier curves consistently outperformed other algorithms in terms of CoT and roll and pitch velocity values. But, ultimately, the selection of the 3rd-degree Bézier curve as the preferred algorithm was driven by its superior height values and the associated computational efficiency advantages.

Concerning posture adjustment, Q-Learning was applied to find the best control parameters for each given inclined terrain. While developing the Q-Learning reward algorithm through simulated robot walking on different slopes, it was found that CoT could be omitted as it was deemed irrelevant. Instead, the algorithm was designed to prioritize body inclination and height variations during slope transitions. Also, a code scheme was developed to connect the posture control with the Q-Learning algorithm, as well as ensuring data storage and error prevention. This code scheme was capable of initiating the Q-Learning algorithm, randomly select a slope environment from an array of five different slope environments and randomly deploy the hexapod robot in two different simulation starting positions, these increased scenarios and starting positions were applied to obtain more trustworthy results. Furthermore, this code scheme was built, not only for this particular scenario of height control in slope environments, but also any type of environment and control parameters. The goal was to make a code scheme easy to adapt and ensure that future research using RL in the hexapod control realm could have a significant head start. As for the control simulations, the core objective of the research was successfully achieved: the robot, initially, unable to navigate a sloped environment without control assistance, became capable of doing so through the application of height control parameters.

In conclusion, this dissertation has successfully achieved the goal of enhancing a robot's adaptability to inclined surfaces through the application of Reinforcement Learning. By equipping the robot with the ability to autonomously select the optimal height control parameter in response to detected inclines, it can safely navigate and adapt its movement for traversing these terrains. Bearing all this in mind, it is possible to conclude that the proposed objectives where attained.

5.2. Perspectives and Future Work

Despite the success in accomplishing the dissertation objectives, there are some improvements that could be made in the project. Even so, upon concluding this work and drawing all the conclusions, there is a rewarding sense that this dissertation has yielded fruitful results.

Regarding the application of the control parameters, along the k_h parameter there are two other important control parameters that were not applied, the roll and pitch control parameters. Time constrictions and lack of computational power imposed a challenge to apply all three control parameters. However, the groundworks of said missing control parameters were applied in the code structure and in future research it is possible, with some minor changes in the code scheme, to study these effectively. It is necessary to mention that the resulting simulations would require a high level of computational power.

Even though the choice of the gait trajectory algorithm is backed by an extensive analysis of five different algorithms, giving relevance to the 3rd and 4th-degree Bézier curves, it is suggested that future research should try to find a correlation between the type of terrain and type of gait trajectory, as it seems that even higher posture stability is obtained with the application of a correct height control parameter and correct gait trajectory algorithm.

Despite the hexapod robot success in adapting to inclined slopes, there were increased oscillations, attributed to abrupt control parameter changes with varying slope inclinations and the fact that higher k_h values result in posture instability. It is suggested that inputting a smooth transition between k_h shifts can reduce the oscillations. Additionally, by incorporating roll and pitch control parameters into posture control, it becomes possible to discover values that enhance stability in complex environments.

REFERENCES

Abdulqadir, H., & Abdulazeez, A. (2021). Reinforcement Learning and Modeling Techniques: A Review. International Journal of Science and Business, 5(3), 174–189.

Alexander, R. McN. (1984). The Gaits of Bipedal and Quadrupedal Animals. The International Journal of Robotics Research, 3(2), 49–59. https://doi.org/10.1177/027836498400300205

Bajaj, N. M., Spiers, A. J., & Dollar, A. M. (2015). State of the art in prosthetic wrists: Commercial and research devices. 2015 IEEE International Conference on Rehabilitation Robotics (ICORR), 331–338. https://doi.org/10.1109/ICORR.2015.7281221

Bal, C. (2021). Neural coupled central pattern generator based smooth gait transition of a biomimetic hexapod robot. Neurocomputing, 420, 210–226. https://doi.org/10.1016/j.neucom.2020.07.114

Bjelonic, M., Kottege, N., & Beckerle, P. (2016). Proprioceptive control of an over-actuated hexapod robot in unstructured terrain. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2042–2049. https://doi.org/10.1109/IROS.2016.7759321

Bjelonic, M., Kottege, N., Homberger, T., Borges, P., Beckerle, P., & Chli, M. (2018). Weaver: Hexapod robot for autonomous navigation on unstructured terrain. Journal of Field Robotics, 35(7), 1063–1079. https://doi.org/10.1002/rob.21795

Campos, R., Matos, V., Oliveira, M., & Santos, C. (2010, September). Gait generation for a simulated hexapod robot : a nonlinear dynamical systems approach. CONTROLO 2010.

Chatzilygeroudis, K., & Mouret, J.-B. (2017). Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics.

Coelho, J., Ribeiro, F., Dias, B., Lopes, G., & Flores, P. (2021). Trends in the Control of Hexapod Robots: A Survey. Robotics, 10(3), 100. https://doi.org/10.3390/robotics10030100

Coelho, J., Silva, D., Marques, F., & Flores, P. (2022, October 19). Real-time optimization of the limbs energetic cost for a hexapod robot according to the terrain topology. 2a Conferência Nacional de Dinâmica de Sistemas Multicorpo.

Deng, H., Xin, G., Zhong, G., & Mistry, M. (2017a). Gait and trajectory rolling planning and control of hexapod robots for disaster rescue applications. Robotics and Autonomous Systems, 95, 13–24. https://doi.org/10.1016/j.robot.2017.05.007

Deng, H., Xin, G., Zhong, G., & Mistry, M. (2017b). Gait and trajectory rolling planning and control of hexapod robots for disaster rescue applications. Robotics and Autonomous Systems, 95, 13–24. https://doi.org/10.1016/j.robot.2017.05.007

Ding, L., Xu, P., Gao, H., Wang, Z., Zhou, R., Gong, Z., & Liu, G. (2020). Fault Tolerant Free Gait and Footstep Planning for Hexapod Robot Based on Monte-Carlo Tree. ArXiv.

Dridi, S. (2021). Reinforcement Learning - A Systematic Literature Review.

Faigl, J., & Čížek, P. (2019). Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only. Robotics and Autonomous Systems, 116, 136–147. https://doi.org/10.1016/j.robot.2019.03.008

Fernández-Conde, J., Cuenca-Jiménez, P., & Cañas, J. M. (2022). Hybrid Training Strategies: Improving Performance of Temporal Difference Learning in Board Games. Applied Sciences, 12(6), 2854. https://doi.org/10.3390/app12062854

Flores, P., & J.C, C. (2005). Introdução ao Estudo de Mecanismos.

Fu, Q., Han, Z., Chen, J., Lu, Y., Wu, H., & Wang, Y. (2022). Applications of reinforcement learning for building energy efficiency control: A review. Journal of Building Engineering, 50, 104165. https://doi.org/10.1016/j.jobe.2022.104165

Gao, Y., Su, B., Jiang, L., & Gao, F. (2022). Multi-legged robots: progress and challenges. National Science Review. https://doi.org/10.1093/nsr/nwac214

He, J., & Gao, F. (2020). Mechanism, Actuation, Perception, and Control of Highly Dynamic Multilegged Robots: A Review. Chinese Journal of Mechanical Engineering, 33(1), 79. https://doi.org/10.1186/s10033-020-00485-9

Hong, J., Tang, K., & Chen, C. (2017). Obstacle avoidance of hexapod robots using fuzzy Q-learning. 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 1–6. https://doi.org/10.1109/SSCI.2017.8280907

Ingrand, F., & Ghallab, M. (2017). Deliberation for autonomous robots: A survey. Artificial Intelligence, 247, 10–44. https://doi.org/10.1016/j.artint.2014.11.003

Josephs, H., & Huston, R. L. (2018). Blake's Design of Mechanical Joints. CRC Press. https://doi.org/10.1201/9781315153827

Khan, M. S., Awan, A. A., Islam, F., Ayaz, Y., & Hasan, O. (2015). Safe-radius based motion planning of hexapod using RRT-connect. 2015 IEEE International Conference on Information and Automation, 415–418. https://doi.org/10.1109/ICInfA.2015.7279323

Khan, Md. A.-M., Khan, M. R. J., Tooshil, A., Sikder, N., Mahmud, M. A. P., Kouzani, A. Z., & Nahid, A.-A. (2020). A Systematic Review on Reinforcement Learning-Based Robotics Within the Last Decade. IEEE Access, 8, 176598–176623. https://doi.org/10.1109/ACCESS.2020.3027152

Kortenkamp, D., & Simmons, R. (2008). Robotic Systems Architectures and Programming. In Springer Handbook of Robotics (pp. 187–206). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-30301-5_9

Lagaza, K., & Pandey, A. (2018). A Literature Review on Motion Planning of Hexapod Machines Using Different Soft Computing Methods. Global Journal of Engineering, Science and Social Science Studies, 3, 1–10.

Li, H., Qi, C., Mao, L., Zhao, Y., Chen, X., & Gao, F. (2021). Staircase-climbing capability-based dimension design of a hexapod robot. Mechanism and Machine Theory, 164, 104400. https://doi.org/10.1016/j.mechmachtheory.2021.104400

Li, Y. (2018). Deep Reinforcement Learning.

Liu, C., Li, Z., Zhang, C., Yan, Y., & Zhang, R. (2019). Gait Planning and Control for a Hexapod Robot on Uneven Terrain Based on Markov Decision Process. 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), 583–586. https://doi.org/10.1109/ICIEA.2019.8834181

Liu, Q., & Jing, T. (2015). A Survey on Hexapod Walking Robot and Gait Planning. Proceedings of the 2015 International Forum on Energy, Environment Science and Materials. https://doi.org/10.2991/ifeesm-15.2015.67

Liu, Y., Wang, C., Zhang, H., & Zhao, J. (2020). Research on the Posture Control Method of Hexapod Robot for Rugged Terrain. Applied Sciences, 10(19), 6725. https://doi.org/10.3390/app10196725

Mahapatra, A., Roy, S. S., & Pratihar, D. K. (2019). Study on feet forces' distributions, energy consumption and dynamic stability measure of hexapod robot during crab walking. Applied Mathematical Modelling, 65, 717–744. https://doi.org/10.1016/j.apm.2018.09.015

Nava Rodri´guez, N. E., Moreno Lorente, L., Carbone, G., & Ceccarelli, M. (2010). A New Design for Cassino Hexapod Robot. ASME 2010 10th Biennial Conference on Engineering Systems Design and Analysis, Volume 3, 557–565. https://doi.org/10.1115/ESDA2010-24020

Nian, R., Liu, J., & Huang, B. (2020). A review On reinforcement learning: Introduction and applications in industrial process control. Computers & Chemical Engineering, 139, 106886. https://doi.org/10.1016/j.compchemeng.2020.106886

Pfeiffer, F., Eltze, J., & Weidemann, H.-J. (1995). Six-legged technical walking considering biological principles. Robotics and Autonomous Systems, 14(2–3), 223–232. https://doi.org/10.1016/0921-8890(94)00031-V

Raibert, M. (1986). Legged Robots that Balance. MIT Press.

Rojas, M., Certad, N., Cappelletto, J., & Grieco, J. C. (2015). Foothold Planning and Gait Generation for a Hexapod Robot Traversing Terrains with Forbidden Zones. 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 49–54. https://doi.org/10.1109/LARS-SBR.2015.70

Saranli, U., Buehler, M., & Koditschek, D. E. (2001). RHex: A Simple and Highly Mobile Hexapod Robot. The International Journal of Robotics Research, 20(7), 616–631. https://doi.org/10.1177/02783640122067570

Silva, M. F., & Tenreiro Machado, J. A. (2007). A Historical Perspective of Legged Robots. Journal of Vibration and Control, 13(9–10), 1447–1486. https://doi.org/10.1177/1077546307078276

Soyguder, S., & Alli, H. (2012). Kinematic and dynamic analysis of a hexapod walking–running–bounding gaits robot and control actions. Computers & Electrical Engineering, 38(2), 444–458. https://doi.org/10.1016/j.compeleceng.2011.10.008

'Sutton, R. S. ' 'Barto, A. G. '. (2018). Reinforcement learning: an introduction (2nd ed.). The MIT Press.

Talaba, D. (2012). The angular capacity of spherical joints used in mechanisms with closed loops and multiple degrees of freedom. Robotics and Computer-Integrated Manufacturing, 28(5), 637–647. https://doi.org/10.1016/j.rcim.2012.03.005

Tedeschi, F., & Carbone, G. (2014). Design Issues for Hexapod Walking Robots. Robotics, 3(2), 181–206. https://doi.org/10.3390/robotics3020181

Verma, S., Nair, H. S., Agarwal, G., Dhar, J., & Shukla, A. (2019). Deep Reinforcement Learning for Single-Shot Diagnosis and Adaptation in Damaged Robots.

Wang, J., Chen, W., Xiao, X., Xu, Y., Li, C., Jia, X., & Meng, M. Q.-H. (2021). A survey of the development of biomimetic intelligence and robotics. Biomimetic Intelligence and Robotics, 1, 100001. https://doi.org/10.1016/j.birob.2021.100001

Xavier, M. S., Tawk, C. D., Zolfagharian, A., Pinskier, J., Howard, D., Young, T., Lai, J., Harrison, S. M., Yong, Y. K., Bodaghi, M., & Fleming, A. J. (2022). Soft Pneumatic Actuators: A Review of Design, Fabrication, Modeling, Sensing, Control and Applications. IEEE Access, 10, 59442–59485. https://doi.org/10.1109/ACCESS.2022.3179589

Xu, K., & Ding, X. (2014). Gait analysis of a radial symmetrical hexapod robot based on parallel mechanisms. Chinese Journal of Mechanical Engineering, 27(5), 867–879. https://doi.org/10.3901/CJME.2014.0619.115

Zaghloul, A., Hussein, W., & Badawy, A. (2016). DYNAMICS AND MEASURING OF FEET FORCE DISTRIBUTIONS OF SIX-LEGGED ROBOT. The International Conference on Applied Mechanics and Mechanical Engineering, 17(17), 1–14. https://doi.org/10.21608/amme.2016.35283

Zha, F., Chen, C., Guo, W., Zheng, P., & Shi, J. (2019). A free gait controller designed for a heavy load hexapod robot. Advances in Mechanical Engineering, 11(3), 168781401983836. https://doi.org/10.1177/1687814019838369

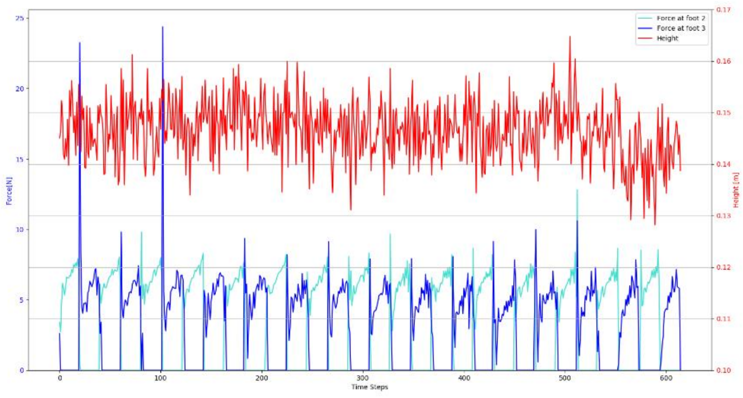
Zhang, C.-D., & Song, S.-M. (1993). A study of the stability of generalized wave gaits. Mathematical Biosciences, 115(1), 1–32. https://doi.org/10.1016/0025-5564(93)90045-C

Zhang, H., Liu, Y., Zhao, J., Chen, J., & Yan, J. (2014). Development of a Bionic Hexapod Robot for Walking on Unstructured Terrain. Journal of Bionic Engineering, 11(2), 176–187. https://doi.org/10.1016/S1672-6529(14)60041-X

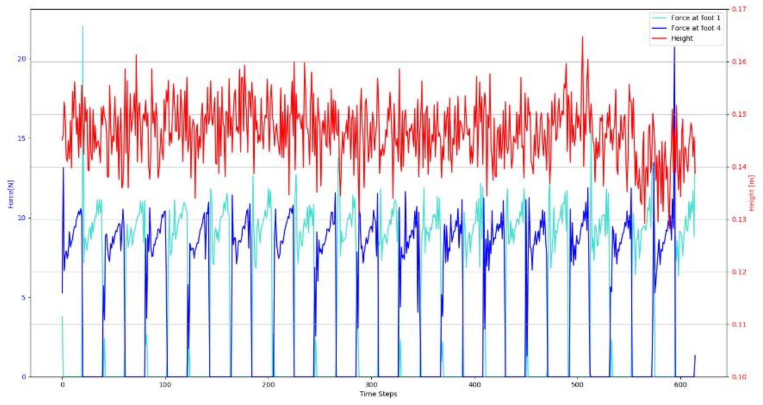
Zhao, Y., Chai, X., Gao, F., & Qi, C. (2018). Obstacle avoidance and motion planning scheme for a hexapod robot Octopus-III. Robotics and Autonomous Systems, 103, 199–212. https://doi.org/10.1016/j.robot.2018.01.007

Appendix

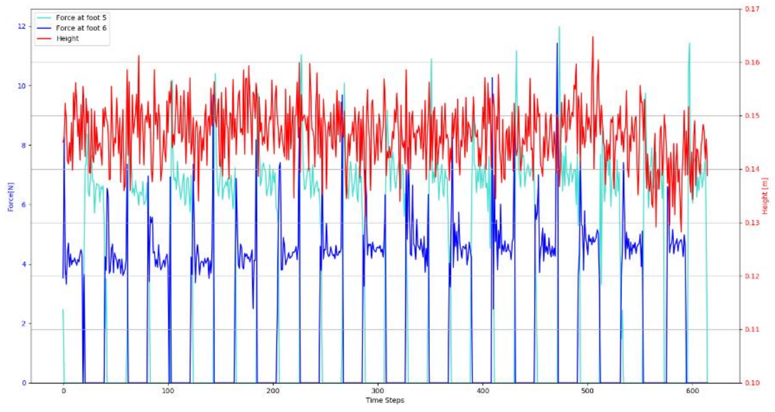
A. Appendix A



(a)

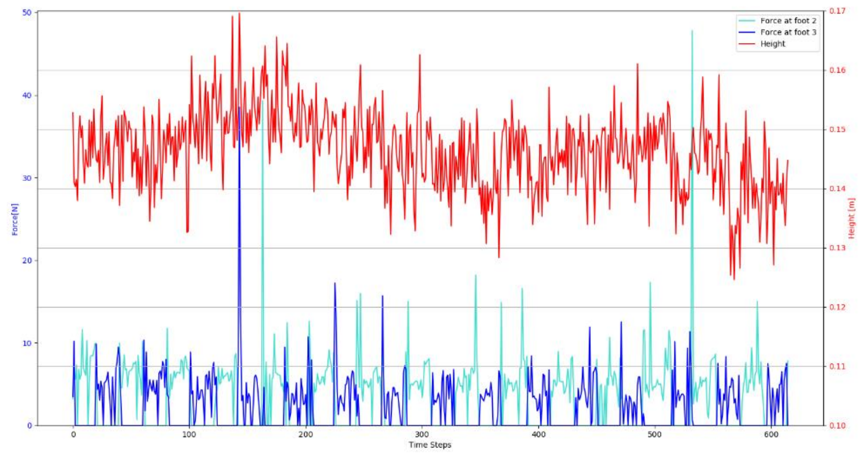


(b)

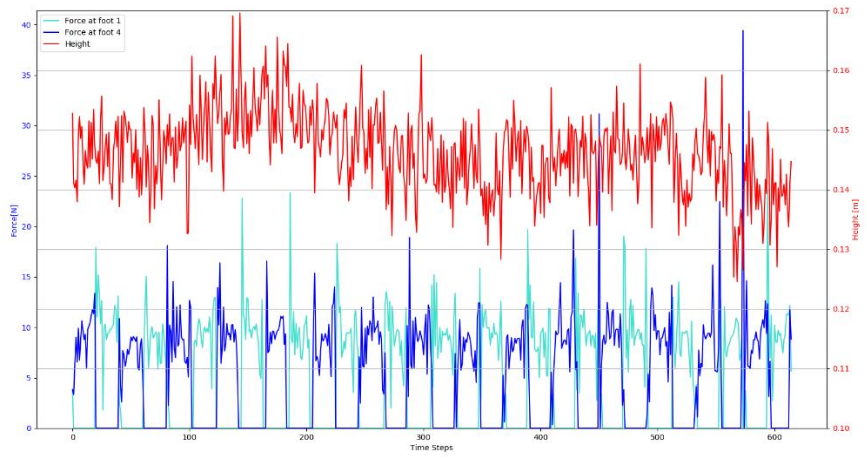


(c)

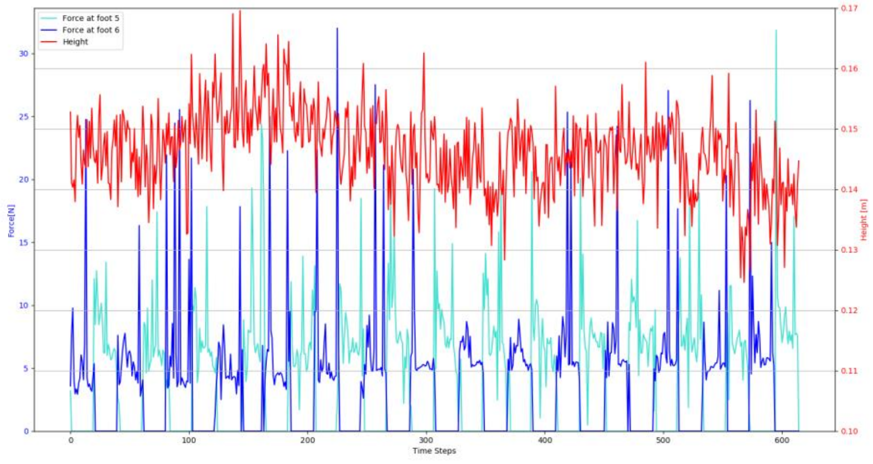
Figure Appendix A-1 Forces applied to the feet in a 3°Slope: (a) front legs, (b) middle legs, (c) back legs.



(a)

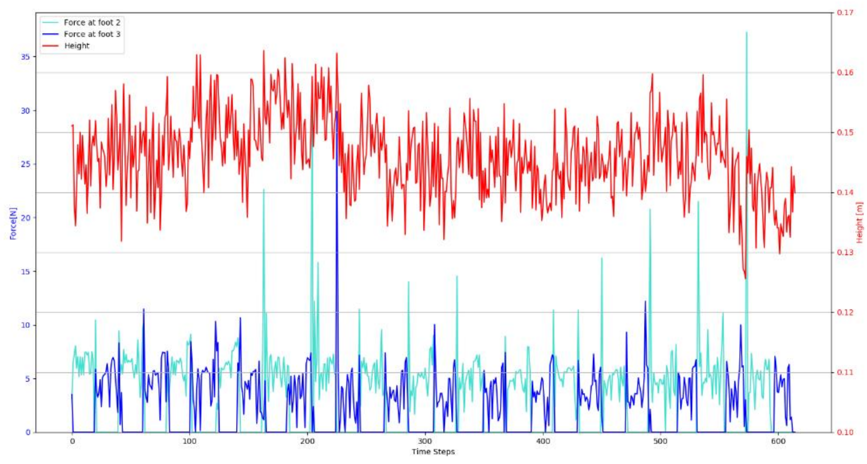


(b)

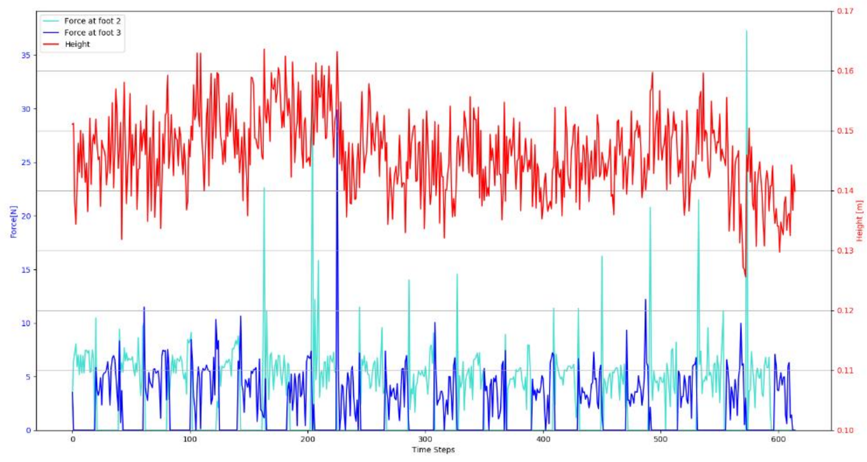


(c)

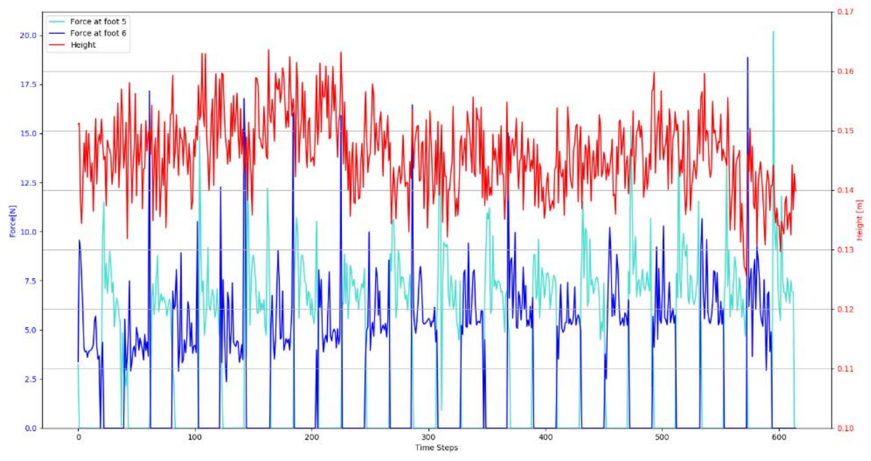
Figure Appendix A-2 Forces applied to the feet in a 6°Slope: (a) front legs, (b) middle legs, (c) back legs.



(a)

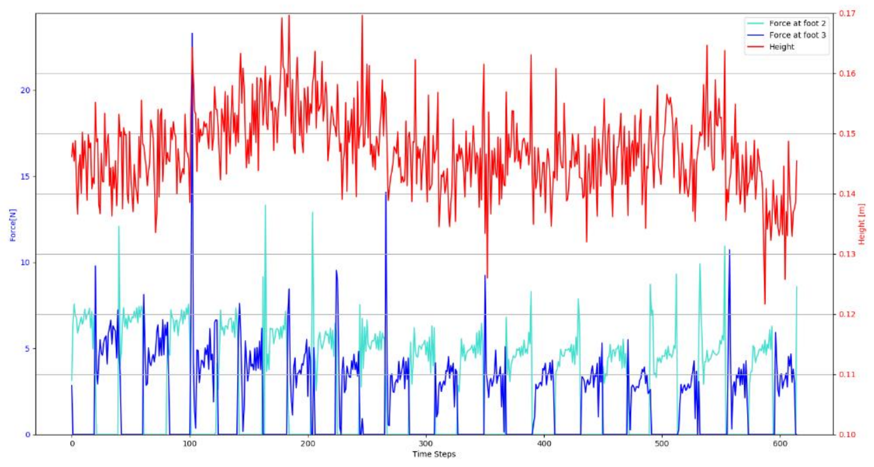


(b)

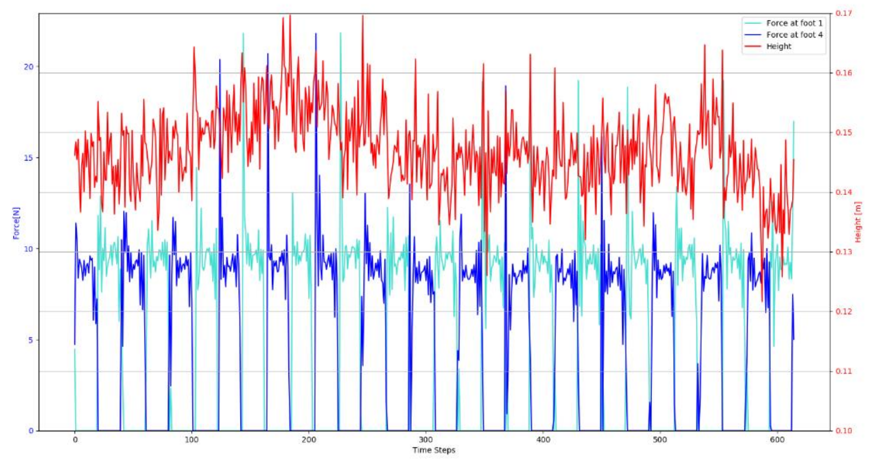


(c)

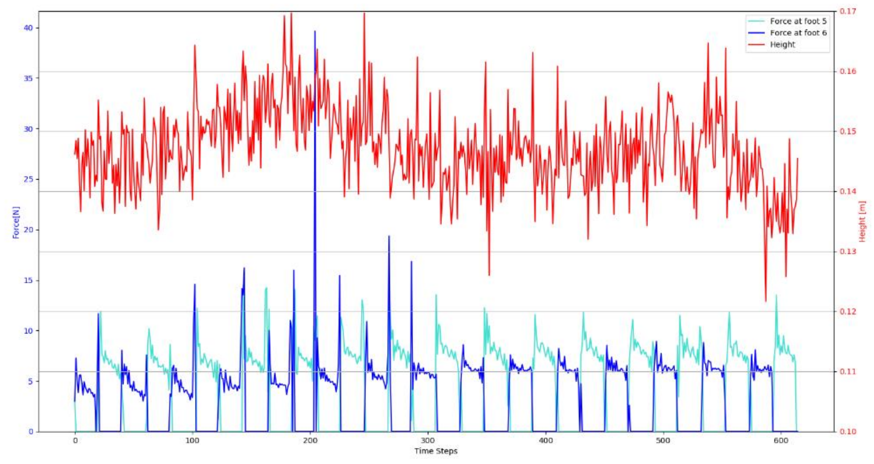
Figure Appendix A-3 Forces applied to the feet in a 9°Slope: (a) front legs, (b) middle legs, (c) back legs.



(a)

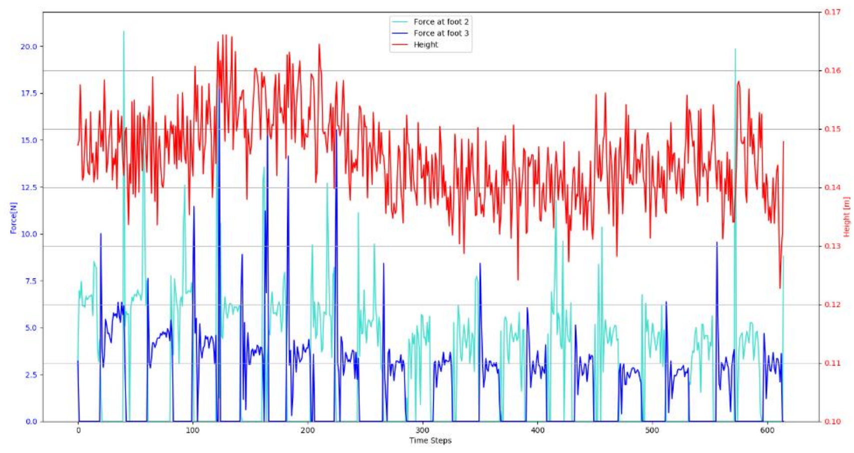


(b)

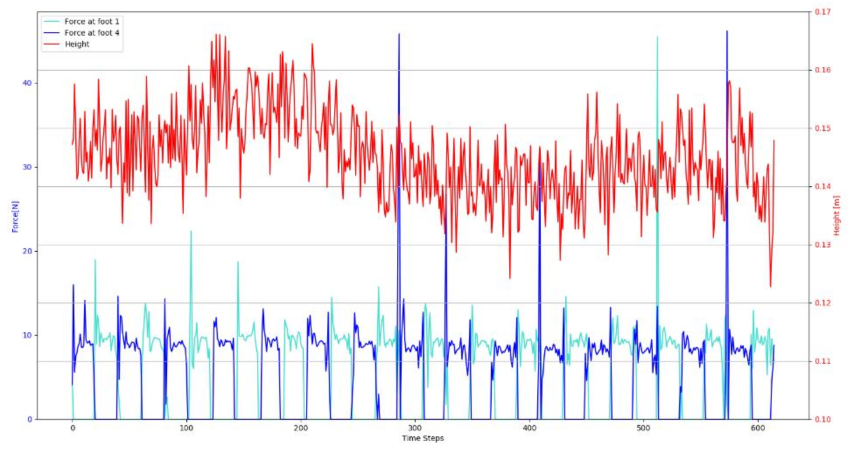


(c)

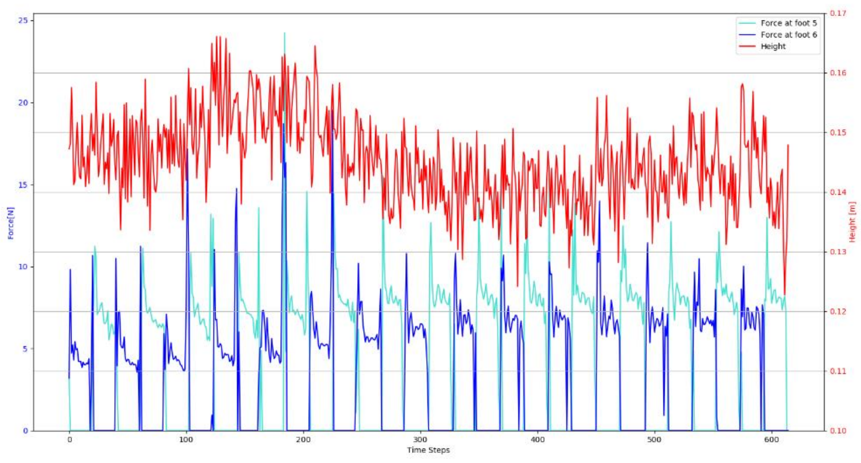
Figure Appendix A-4 Forces applied to the feet in a 12°Slope: (a) front legs, (b) middle legs, (c) back legs.



(a)



(b)



(c)

Figure Appendix A-5 Forces applied to the feet in a 15°Slope: (a) front legs, (b) middle legs, (c) back legs.

B. Appendix B

Table Appendix B-1 Key values at each 100 episodes in SIM_3.

Episode	**Reward**	$\boldsymbol{k_h}$	**Height** [m]	**Roll velocity** [rads/s]	**Pitch velocity** [rads/s]	$\boldsymbol{b_{deg}}$	**Real slope**
99	+7.696	0.0	0.1354	0.5550	0.3140	4.87°	6°
199	+8.435	0.0	0.1357	0.5221	0.3541	5.24°	6°
299	+2.358	0.4	0.1462	0.4500	0.3116	5.26°	9°
399	–1.613	0.2	0.1440	0.2985	0.2107	6.22°	12°
499	+9.732	0.6	0.1464	0.4370	0.3948	2.90°	3°
599	–0.417	0.2	0.1413	0.2396	0.2911	6.82°	12°
699	+1.640	0.0	0.1389	0.4073	0.2902	4.86°	9°
799	–1.244	1.0	0.1456	0.2392	0.3067	3.50°	9°
899	+7.289	0.0	0.1346	0.5148	0.3390	4.67°	6°
999	+2.123	0.4	0.1492	0.2915	0.2033	5.01°	9°

Table Appendix B-2 Key values at each 100 episodes in SIM_4.

Episode	**Reward**	$\boldsymbol{k_h}$	**Height** [m]	**Roll velocity** [rads/s]	**Pitch velocity** [rads/s]	$\boldsymbol{b_{deg}}$	**Real slope**
99	+9.265	0.8	0.1446	0.3119	0.2294	2.67°	3°
199	+1.186	0.6	0.1492	0.3933	0.2793	4.66°	9°
299	+8.535	0.6	0.1492	0.4724	0.2440	2.31°	3°
399	+8.749	0.6	0.1405	0.1597	0.1597	2.40°	3°
499	+9.082	0.6	0.1451	0.0604	0.1838	2.56°	3°
599	+7.358	0.6	0.1525	0.4072	0.4341	4.72°	6°
699	–2.568	0.2	0.1460	0.5412	0.5186	5.75°	12°
799	+9.938	0.0	0.1368	0.5695	0.3339	6.00°	6°
899	+8.614	0.2	0.1494	0.3673	0.3400	2.33°	3°
999	–2.616	0.8	0.1455	0.2891	0.5933	5.77°	12°

Table Appendix B-3 Key values at each 100 episodes in SIM_5.

Episode	**Reward**	k_h	**Height** [m]	**Roll velocity** [rads/s]	**Pitch velocity** [rads/s]	b_{deg}	**Real slope**
99	+2.945	0.0	0.1363	0.6470	0.3576	5.50°	9°
199	+9.966	0.0	0.1475	0.1218	0.1050	2.91°	3°
299	+2.775	0.8	0.1464	0.4152	0.3776	5.42°	9°
399	+9.862	0.8	0.1540	0.0900	0.2531	2.40°	3°
499	–1.559	0.8	0.1531	0.2449	0.8710	6.29°	12°
599	+14.85	0.8	0.1466	0.3730	0.3447	4.89°	6°
699	–1.773	0.2	0.1389	0.3553	0.2410	6.14°	12°
799	–3.335	0.4	0.1365	0.2997	0.2490	5.37°	12°
899	+9.95	0.8	0.1523	0.2847	0.3847	3.06°	3°
999	+14.933	0.6	0.1454	0.2571	0.3063	4.79°	6°

Table Appendix B-4 Key values at each 100 episodes in SIM_6.

Episode	**Reward**	k_h	**Height** [m]	**Roll velocity** [rads/s]	**Pitch velocity** [rads/s]	b_{deg}	**Real slope**
99	+1.257	0.4	0.1552	0.3362	0.3061	4.66°	9°
199	+7.486	0.0	0.1403	0.4147	0.2551	4.78°	6°
299	+9.942	0.6	0.1427	0.3258	0.3169	2.33°	3°
399	+1.760	0.8	0.1386	0.4639	0.3267	4.92°	9°
499	+1.166	0.8	0.1523	0.4165	0.5550	4.62°	9°
599	+3.714	0.2	0.1436	0.5227	0.3098	5.87°	9°
699	–0.354	0.0	0.1304	0.1470	0.2154	6.88°	12°
799	–1.234	0.8	0.1478	0.4370	0.8336	6.48°	12°
899	+2.335	0.8	0.1453	0.2996	0.2133	5.19°	9°
999	–0.996	0.8	0.1449	0.3713	0.7015	6.59°	12°

C. Appendix C

The code developed in all different phases of the dissertation can be seen in: https://github.com/Diogo-Silva99/ATHENA_Posture. The provided link contains three files: "*Athena_Code*," "*gd_test*," and "worlds."

Within "Athena_Code," various files, including the first stage of the client ROS *.py* file, the first stage of ATHENA locomotion based *.py* files, the simulation launcher file that calls all dependencies, a model description in a *.xacro* file, and *.yaml* files containing data on ROS topics, are exposed.

Inside *"gd_test"* several files in the last stage of development, including the client ROS *.py* file, the ATHENA locomotion base *.py* files and the simulation launcher, are revealed.

Finally, in "*worlds"* all simulation environments used throughout this dissertation are presented.