Overcoming the Reality Gap: Imitation and Reinforcement Learning Algorithms for Bipedal Robotic Locomotion Problems

David Reinerio Yanguas Rojas

Universidad Nacional de Colombia
Faculty of Engineering, Mechanical, and Mechatronic Engineering Department
Bogotá, Colombia
2023

Overcoming the Reality Gap: Imitation and Reinforcement Learning Algorithms for Bipedal Robotic Locomotion Problems

David Reinerio Yanguas Rojas

Thesis presented as partial requirement for applying to the title of:
Ph.D. in Mechanical and Mechatronical Engineering

Advisor:
Ph.D. Eduardo Alirio Mojica Nava

Universidad Nacional de Colombia
Faculty of Engineering, Mechanical, and Mechatronic Engineering Department
Bogotá, Colombia
2023

Dedication

This thesis is dedicated to the most incredible person in my life - my beloved mother Ana Lilia Rojas Leal. Without her unwavering love, sacrifices, and tireless efforts in both her work and raising me, I would not have had the extraordinary opportunity to dedicate myself to the pursuit of research.

This thesis stands not only as a testament to my academic accomplishments but as a tribute to your boundless love and selflessness. It is a small token of gratitude compared to the immeasurable debt of gratitude I owe you. You have given me the wings to soar, the courage to pursue my dreams, and the strength to overcome any obstacle.

Acknowledgements

I would like to express my deepest gratitude to Professors Eduardo Mojica-Nava, Alben Cárdenas, and Jonatan Gómez for their invaluable support and guidance throughout the development of this research project. Their expertise, encouragement, and unwavering commitment to academic excellence have been instrumental in shaping the direction and success of my work.

I am immensely grateful to MITACS for their generous support in funding my research internship at the Université du Québec à Trois-Rivières (UQTR). The opportunity provided by MITACS not only enriched my academic experience but also allowed me to collaborate with esteemed researchers and access vital resources, which significantly contributed to the advancement of my research.

Furthermore, I am grateful to the Universidad Nacional de Colombia for their unwavering support throughout my academic journey. I would like to express my sincere appreciation for the opportunities I had as a recipient of the Auxiliary Professor and Assistant Professor scholarships. These scholarships not only provided financial assistance but also allowed me to deepen my knowledge, engage in valuable teaching experiences, and contribute to the academic community.

Finally, I would like to express my heartfelt appreciation to my friends and family for their unwavering support, understanding, and encouragement throughout this journey. Their belief in my abilities and their constant motivation have been a source of inspiration and strength.

To all those mentioned above and to those who have contributed in various ways but may not be named explicitly, I extend my sincere gratitude. Without your support and contributions, this research endeavor would not have been possible.

Resumen

Título en Español : Superando la brecha de la realidad: Algoritmos de aprendizaje por imitación y por refuerzos para problemas de locomoción robótica bípeda.

Esta tesis presenta una estrategia de entrenamiento de robots que utiliza técnicas de aprendizaje artificial para optimizar el rendimiento de los robots en tareas complejas. Motivado por los impresionantes logros recientes en el aprendizaje automático, especialmente en juegos y escenarios virtuales, el proyecto tiene como objetivo explorar el potencial de estas técnicas para mejorar las capacidades de los robots más allá de la programación humana tradicional a pesar de las limitaciones impuestas por la brecha de la realidad.
El caso de estudio seleccionado para esta investigación es la locomoción bípeda, ya que permite dilucidar los principales desafíos y ventajas de utilizar métodos de aprendizaje artificial para el aprendizaje de robots. La tesis identifica cuatro desafíos principales en este contexto: la variabilidad de los resultados obtenidos de los algoritmos de aprendizaje artificial, el alto costo y riesgo asociado con la realización de experimentos en robots reales, la brecha entre la simulación y el comportamiento del mundo real, y la necesidad de adaptar los patrones de movimiento humanos a los sistemas robóticos.
La propuesta consiste en tres módulos principales para abordar estos desafíos: Enfoques de Control No Lineal, Aprendizaje por Imitación y Aprendizaje por Reforzamiento. El módulo de Enfoques de Control No Lineal establece una base al modelar robots y emplear técnicas de control bien establecidas. El módulo de Aprendizaje por Imitación utiliza la imitación para generar políticas iniciales basadas en datos de captura de movimiento de referencia o resultados preliminares de políticas para crear patrones de marcha similares a los humanos y factibles. El módulo de Aprendizaje por Refuerzos complementa el proceso mejorando de manera iterativa las políticas paramétricas, principalmente a través de la simulación pero con el rendimiento en el mundo real como objetivo final.
Esta tesis enfatiza la modularidad del enfoque, permitiendo la implementación de los módulos individuales por separado o su combinación para determinar la estrategia más efectiva para diferentes escenarios de entrenamiento de robots. Al utilizar una combinación de técnicas de control establecidas, aprendizaje por imitación y aprendizaje por refuerzos, la estrategia de entrenamiento propuesta busca desbloquear el potencial para que los robots alcancen un rendimiento optimizado en tareas complejas, contribuyendo al avance de la inteligencia artificial en la robótica no solo en sistemas virtuales sino en sistemas reales.
Palabras clave: Entrenamiento de robots, Locomoción humanoide ,Técnicas de aprendizaje artificial, Aprendizaje por refuerzos, Aprendizaje por imitación, Control no lineal, Brecha entre simulación y realidad.

Abstract

Title in English: Overcoming the Reality Gap: Imitation and Reinforcement Learning Algorithms for Bipedal Robotic Locomotion Problems

The thesis introduces a comprehensive robot training framework that utilizes artificial learning techniques to optimize robot performance in complex tasks. Motivated by recent impressive achievements in machine learning, particularly in games and virtual scenarios, the project aims to explore the potential of these techniques for improving robot capabilities beyond traditional human programming.

The case study selected for this investigation is bipedal locomotion, as it allows for elucidating key challenges and advantages of using artificial learning methods for robot learning. The thesis identifies four primary challenges in this context: the variability of results obtained from artificial learning algorithms, the high cost and risk associated with conducting experiments on real robots, the reality gap between simulation and real-world behavior, and the need to adapt human motion patterns to robotic systems.

The proposed approach consists of three main modules to address these challenges: Non-linear Control Approaches, Imitation Learning, and Reinforcement Learning. The Non-linear Control module establishes a foundation by modeling robots and employing well-established control techniques. The Imitation Learning module utilizes imitation to generate initial policies based on reference motion capture data or preliminary policy results to create feasible human-like gait patterns. The Reinforcement Learning module complements the process by iteratively improving parametric policies, primarily through simulation but ultimately with real-world performance as the ultimate goal.

The thesis emphasizes the modularity of the approach, allowing for the implementation of individual modules separately or their combination to determine the most effective strategy for different robot training scenarios. By employing a combination of established control techniques, imitation learning, and reinforcement learning, the framework seeks to unlock the potential for robots to achieve optimized performances in complex tasks, contributing to the advancement of artificial intelligence in robotics.

Keywords: Robot training, Bipedal locomotion, Humanoid locomotion, Artificial learning techniques, Reinforcement learning, Imitation learning, Non-linear control, Reality gap, Sim to real

Contents

Figures List

Tables List

1. Introduction

1.1. Motivation

In recent years machine learning techniques have gotten awe-inspiring results as the victory of the artificial intelligence AlphaGO over Lee Sedol, the world champion of the board game Go [Google, 2015]; the more recent wins over human professionals in the real-time strategy games Dota 2 and StarCraft in the hands of OpenAI five [OpenAI, 2018] and AlphaStar [Vinyals et al., 2019] respectively; the success in dealing with the problem of protein folding addressed with by Deepmind with Alphafold [Jumper et al., 2021]; and the rise of the large language models as ChatGPT by OpenAI [Thorp, 2023]. These achievements, among many others, awaken the interest in the study of artificial intelligence techniques for the resolution of complex problems and entail the possibility of obtaining super-human performances in many areas.

With this motivation in mind, the possibility of developing machine learning strategies for robots using different approaches to achieve optimized performances that could surpass those obtained by traditional human programming is being considered. In particular, the academic community is studying these artificial learning techniques (especially reinforcement learning techniques) to achieve robust and efficient controllers for different class problems to obtain similar results to those in video game environments. However, this is not an easy task as there are many challenges to consider when trying to implement these techniques in robotic agents in reality. Among these challenges, some of the most important ones that have been identified and on which the academic community is working are the considerable variability of the results after multiple trials of the techniques on the same problem as presented in [Recht, 2018], the difference between the behaviors shown by the robots in simulation compared to the behaviors those that occur in reality (often referred to as *reality gap* or *sim to real*) as presented in [Tan et al., 2018] and the cost (in time and energy) represented by testing directly on the robot of interest compared to the cost required by simulations as mentioned in [Koos et al., 2013]. It should be noted that learning processes for robots do not necessarily have to start from scratch, but it is possible to incorporate preliminary knowledge to perform the training processes more efficiently. For example, reference movement patterns can be used as developed in [Merel et al., 2017], it is possible to imitate animal behaviors as worked in [Ijspeert, 2008], or it is also possible to take as a starting point the parameters corresponding to previously determined sub-optimal behaviors that can be improved as performed in [Rosolia and Borrelli, 2016], among some other approaches.

This project aims to propose a modular training scheme based on artificial learning techniques that allow the integration of several of the approaches of the area to robotic locomotion tasks. Specifically, the analysis of bipedal locomotion tasks is proposed as a case study for the development of the project because in this kind of task, aspects of interest regarding artificial learning applied to complex robotic systems can be elucidated, and there is a lot of reference material both for the comparison with traditional methods such as those presented in [Ding et al., 2019] and also preliminary information sources such as motion capture data as in [Merel et al., 2017].

1.2. Background and Related Work

1.2.1. Reinforcement Learning

The study of artificial learning is a booming field, and one of its most studied lines of work currently corresponds to reinforcement learning. In this line, the aim is for an agent to be able to generate a mapping between the space of its observations and the space of its actions in a given environment in such a way as to maximize the reward obtained from the interaction, as mentioned in [Sutton and Barto, 2008]. This reward is a function that measures the quality of the observed behavior (determined by the programmer) that allows the agent to associate which action-state pairs lead to the best performances in a manner analogous to the behavioral reinforcement applied in animals (good behaviors are rewarded while bad behaviors are penalized). Using these ideas, it is possible to achieve complex behaviors through repeated interaction of agents with their environment. For example, it is possible to stabilize an inverted pendulum using simple reinforcement learning models. Likewise, using more sophisticated tools, such as convolutional neural networks, allows solving more complex problems, such as playing video games from image pixels or solving locomotion problems in simulation environments, as presented in [Duan et al., 2016]. It is worth noting that there is a wide variety of methods to perform reinforcement learning since, as in any optimization problem, intelligent exploration of the search space is required, and this exploration can be performed in many ways. It is also worth noting that although these algorithms were initially proposed for discrete models (modeled as Markov decision processes), their use has been widely extended to continuous problems.

In [Recht, 2018], an overview of reinforcement learning is presented. A parallel is made between the approach proposed by the areas more related to computer science and the approach presented by the areas more pertaining to automatic control, calling for the integration of the research community around these topics to work towards what is termed actionable intelligence. Along the same line of work, in [Mania et al., 2018], it is shown how a reinforcement learning scheme allows for solving complex problems efficiently while presenting some of the significant challenges of using these techniques, such as the variability of the results and the need to evaluate the performance of training algorithms robustly and comparably.

On the other hand, some works seek to change how the reward function is proposed to improve the performance of reinforcement learning algorithms by giving additional bonuses that encourage the exploration of the environment more appropriately according to the problem. For example, in works such as [Burda et al., 2018], the use of curiosity as a bonus factor for the reward function is implemented, achieving diversification of the executed behaviors by looking for states that "surprise" the agent by presenting observations that have not been seen before or that differ with the expectation of the same predictive models. Similarly, in [Lee et al., 2019], the use of entropy factors as a bonus to the reward function is generalized so that as a parameter q (called the entropic index) is adjusted, the algorithm's behavior is tuned to increase exploration or exploitation in the search space. As another alternative, the requirement for an a priori definition of the reward function is criticized in [Singh et al., 2019], and the use of a robot framework is proposed that allows the determination of the reward function from the demonstration of desired behaviors in front of a vision system.

OpenAI and the Gym library

OpenAI is one of the largest companies dedicated to research in artificial intelligence and robotics and has developed several projects in these areas. One of the most notable projects is the development of the Gym library, which contains a wide variety of tools and test problems frequently used in the validation and comparison of artificial learning algorithms as presented in [Duan et al., 2016]. A large part of the community has adopted these problems as a standard for testing the performance of their algorithms. In particular, locomotion problems implemented with the MuJoCo simulator introduced in [Todorov, 2019] have been used in works such as [Lee et al., 2019], [Mania et al., 2018], [Recht, 2018], [Schulman et al., 2015], [Merel et al., 2017], [Lillicrap et al., 2015], [Erez et al., 2015], and [Uchibe, 2018] among many others to validate the effectiveness of their methods.

1.2.2. Learning by Demonstration or Imitation

There are many studies on robot learning, and many tools have been developed for using robots. For example, in [Chernova and Thomaz, 2014], a compendium of the basic concepts of learning by demonstration for robots with human teachers is presented, showing some methods of interaction between "teacher" and "learner", a classification of the types of tasks developable by robots according to the complexity of the tasks (and the complexity of their learning) and methods to refine the learned behaviors are also presented. The usual work cycle required to improve the performance of tasks learned by demonstration using techniques such as batch learning or teacher correction, among others, is also presented.
Along the same line of work, a demonstration training algorithm is presented in
[Chernova and Veloso, 2010]. A single "teacher" presents how to perform a given task to multiple robots that learn from it simultaneously. This process is based on the interaction

between students and teachers in a classroom, whereby once the teacher performs the demonstration, the students try to imitate it. If they require more information, they request additional information from the teacher and iteratively refine their procedures. In this paper, they present and validate a learning scheme under which two humanoid robots learn to perform a ball organization task cooperatively from the teaching of a human teacher.
A recent alternative to learning by demonstration is presented in [Duan et al., 2017], which introduces a learning technique applied to robots and based on the artificial vision that seeks to extend the information learned in the demonstration of a single task (block stacking) to multiple other instances of the same task under different conditions. In this case, some instances of the task performed by a human in a virtual reality environment are taken as a basis. Subsequently, the robot predicts the next movement from the experience obtained for each new task. This way, the task can be completed for multiple initial conditions even if the robot has not explicitly seen it.
As another alternative, there are works such as the one presented in [Koenemann et al., 2014] in which a humanoid robot is sought to be able to mimic human movements obtained from motion capture while preserving the robot's balance. To do so, they take the position of the robot's end-effectors as a reference but prioritize preserving the robot's center of mass within its lift polygon either on both feet or only on one of them. This prioritization is achieved by generating the robot poses based on those obtained by the motion capture system but modifying them so that the result is a statically stable pose at each instant.
Another work to highlight learning by imitation is the one presented in [Merel et al., 2017] in which motion capture data is taken as a reference to imitate to achieve efficient, robust, and more natural gait patterns than those obtained without any reference. In this work, it is mentioned that direct imitation is usually not feasible since the geometric characteristics of the individuals do not necessarily match those of the agent that is imitating it, so the imitation must be done more elaborately. To perform the imitation, they propose the use of generative adversarial neural networks (GANs) to generate the action signals on the agent from the information of the agent's state, which is compared with those provided by the data so that with the passage of iterations, the generative network produces results very close to the data but operating on the agent of interest. Within the same work, they propose modularizing the learned behaviors and their use as tools to develop more complex behaviors, such as climbing stairs or navigating environments with obstacles.

1.2.3. The Reality Gap

Due to the iterative nature of learning methods, it is widespread that they tend to be developed much more frequently in simulated environments than in real environments since simulations can be performed much faster and at a significantly lower cost in most cases. However, simulators have certain differences from reality either due to errors in the characterization of the system, due to unmodeled dynamics, or errors inherent to the model (such

as discretization). Notably, the calculation of contact forces between rigid and deformable bodies is a problem that is not entirely solved and is an important source of errors in the simulation of robotic systems. Because of this, the results obtained by optimization processes in this kind of environment do not necessarily transfer directly to real robots, to the point that results with very high performance in simulation may become completely unfeasible in reality or may require subsequent stages of fine-tuning of the model to be appropriate.

Randomization of the task domain is often presented as an alternative to overcome the reality gap. Notably, the work presented in [Tobin et al., 2017] seeks to enable manipulator robots to move objects from a vision system in a robust manner with multiple distractors from simulation training in which the textures with which different objects in the scene are rendered as well as the position and geometric features of various elements are randomly generated between experiments so that robust policies are learned. Reality is taken as " just another scenario", which was shown not to require additional tuning to function properly as in other approaches.

In [Tan et al., 2018], the use of robust control models in conjunction with the refinement of the modeling to a great detail is presented as an alternative so that the simulator represents in a sufficiently reliable way the real robot and, in this way, the behaviors obtained in the simulation are compatible with the real robot. As a case study, they present the modeling of a quadruped robot intended to learn efficient walking patterns during training, highlighting that the patterns learned only in the basic simulation are not transferred appropriately. To mitigate the differences between reality and the simulator, they model in detail the actuators of the robot and the latency between the control signals on the system. In this way, they achieve efficient behaviors both in reality and in the simulation, closing the gap for this problem.

On the other hand, in [Koos et al., 2013], there is a brief review of what the reality gap is, explaining some of the methods that are commonly used to deal with it (such as introducing perturbations to the simulation or tuning the policy resulting from the simulations to reality a posteriori, Among others) and an approach called the Transferability Approach is proposed. This approach poses a multi-objective optimization problem in which the aim is to maximize the performance function and a transferability function at the same time. This transferability function seeks to capture which behaviors correspond highly between reality and simulation. It is obtained by performing a reduced number of experiments in reality together with the experiments in simulation so that those that show a performance close to the simulation get high values of transferability and vice versa. In this way, the final behaviors obtained at the end of the process are feasible both in simulation and in reality, taking advantage of the versatility of the simulators to perform a large number of tests with few experiments on the real robot.

Similarly, in [Rodriguez et al., 2018], the efficiency of the number of experiments required by other methods is criticized, and the use of Bayesian optimization methods with experiments both in simulators and in real robots for bipedal locomotion problems is proposed as an

alternative. It should be noted that in this case, the learning is not done from zero, but it is based on a predetermined open-loop running pattern which is complemented by a feedback loop that is responsible for strengthening the performance of the system by modifying the action on the different actuators according to the state of the robot.

1.3. Problem Identification

When thinking about implementing robots to perform complex tasks, it is desired that their programming is as simple as possible for the person teaching the task. Also, it is sought that the robots are efficient in developing it in terms of speed of execution and associated energy consumption. However, traditionally the programming of robots for the execution of tasks has been a complex process that requires specific knowledge and skills for its realization since the programming and kinematic and kinetic analysis of a robot are not trivial tasks.
Artificial learning techniques (such as imitation learning or reinforcement learning) have been proposed to allow robots to execute tasks properly without explicit programming as an alternative to achieve this ideal of simple and practical programming. This alternative presents many opportunities for the development of the area since virtual agents trained using artificial learning algorithms have demonstrated remarkable performances, such as the victory of virtual entities in board games and video games by artificial intelligence against human professionals. However, there are also many challenges to consider to realize this application. Some of the most relevant are:

- The high variability shown by the results obtained by artificial learning methods, not only for robotics tasks but for many other classes of problems as developed in [Recht, 2018]. In that work, it is presented that the high variability leads to the results obtained by these methods being very susceptible to change from one run of the algorithm to another to the point where different results are reported by varying the seeds of the random number generators employed as well as by varying the corresponding hyperparameters.

- The high cost of time and resources involved in performing many experiments on robots which are necessary due to the nature of these algorithms as mentioned in [Rodriguez et al., 2018]. Conducting experiments on the actual robots involves time, energy, wear and tear on the robot, and some risk of damage from falls or unsafe behaviors. In addition, one also has that contrary to virtual environments, resetting the experiments is usually done manually and requires time.

- The difference between the behaviors observed in real robots and those observed in simulation, i.e., the reality gap. This gap hinders the direct application of these techniques in purely simulation environments since behaviors that present a very high

performance are not necessarily transferable to real robots either because they are exploiting some phenomenon that only occurs in the simulator or because the model is not sufficiently detailed to correspond to reality as presented in [Koos et al., 2013].

- The fact that human motion patterns are not necessarily compatible with robots as presented in [Koenemann et al., 2014] or even more, the fact that even when compatibility is given (intrinsically or by making appropriate adjustments), these patterns are not necessarily the most suitable to perform the task with optimized performance since when seeking to teach a new task to a robot, the person in charge of the demonstration will not necessarily be an expert in the execution of this with what the results will not necessarily be of high quality.

Thus, the problem to be addressed is the development of a robot training framework based on artificial intelligence that allows robots to perform complex tasks in an optimized way in terms of execution speed and energy consumption associated with them, solving each of the above challenges. This framework is intended to apply to different tasks that can be modeled under a reinforcement learning scheme, i.e., an agent that interacts with its environment and obtains certain rewards (or punishments) from such interaction. In contrast, a person can demonstrate the task for its execution, even if it is not optimal.

1.3.1. Case Study - Bipedal Locomotion

It is important to clarify that countless tasks fall under this categorization (tasks that can be approached with artificial intelligence techniques). However, this thesis will focus only on bipedal locomotion tasks as a case study. This case is chosen as it allows us to properly illustrate the advantages mentioned earlier and the challenges that emerge from using these methods for robot learning.

- There is wide variability in the results demonstrated by different artificial learning algorithms as presented in [Mania et al., 2018] on humanoid locomotion tasks in the MuJoCo Locomotion benchmark presented in [Duan et al., 2016]. This variability is evidenced by the wide variety of different gait patterns that can emerge from random conditions, among which there are patterns similar to those employed in reality by people. Still, also strange behaviors appear, such as lateral or asymmetric gaits where one leg moves much more than the other.

- The cost of performing the experiments directly on the robots of interest is evident as the locomotion tasks involve wear and tear on the robot parts, generally require manual reconfiguration of the environment and have some risk of damage due to falls.

- Simulation models for walking robots are usually not close enough to reality. Even with quadruped models, it is impossible to directly migrate motion patterns learned in

simulation to reality as presented in [Tan et al., 2018]. Therefore, biped models that are even more delicate to work with (because they usually lack static stability during some intervals within their gait) are even more exposed to this phenomenon.

- Although humanoid bipedal robots exist, in general, the specific morphology, the number of degrees of freedom, and weight distribution of these robots are not necessarily identical to that of humans, so the patterns presented by humans will generally not be directly transferable patterns for robots and even when they are, they will be subject to improvement.

Additionally, bipedal robotic locomotion is a problem for which a large amount of information is available for development and comparison since it is a problem widely studied both by classical methods (predetermined closed solutions) as presented in [Ma et al., 2018] or in [Kobayashi et al., 2018] and by learning-based methods as in [Merel et al., 2017]. Similarly, it is worth noting that there are motion capture databases for various human movements (such as walking, jogging, and running, among many others), such as those found in [Carnegie Mellon University Graphics Lab, 2004]. Additionally, generating new data with relative ease is possible, which is considered sufficient availability of material for studying and developing new techniques.

1.3.2. Formal Problem Definition

The problem is presented as a reinforcement learning problem from the control perspective following the notation and ideas presented in [Recht, 2018]. We have a time-invariant dynamical system of the form:

$$x_{t+1} = f(x_t, u_t, e_t)$$

where f is a function that maps from the state x_t, the input u_t and the current random perturbations e_t of the system to the next state of the same x_{t+1} at time t. We also consider a function $R_t(x_t, u_t)$, which, for each time instant, assigns a reward to the pairs of (x_t, u_t), which we seek to maximize. Thus, the problem in general terms is stated as follows:

$$\text{maximize } \mathbb{E}_{e_t}\left[\sum_{t=0}^{N} R(x_t, u_t)\right]$$

,

$$\text{subject a } x_{t+1} = f(x_t, u_t, e_t)$$

,

$$(\text{given } x_0),$$

.

We seek to maximize the cumulative reward function obtained by the dynamic system over time by manipulating the control signal u_t. To determine u_t, we define a function called

"policy" $\pi_t(x_t)$, which maps the information of the system state (and possibly the pairs $(x_0, u_0, ..., x_t, u_t)$ visited so far) to the available set of actions.

For the project, the dynamic system of interest will correspond to the biped robot under study, in which case f will correspond to the equations governing the robot's motion (which must be approximated in the modeling process), x_t to the robot state information (inferred from its sensors), u_t to the control signal imposed on the robot (whether voltages, torques or set point signals are applied) and e_t will be the external disturbances affecting the robot as well as the effect of dynamics not considered in the modeling.

The reward function to be considered must be determined during the project's development. However, a priori, it is known that it should be increasing concerning the speed of movement of the robot in the task v_t and should be decreasing concerning the energy consumption c_t (or a function that is proportional to it) so that the optimization objectives are balanced,

$$R(x_t, u_t) \propto v_t$$

,

$$R(x_t, u_t) \propto -c_t$$

.

Finally, it is emphasized that the policies π_t can take many different forms and that their determination is the key to resolving the problem. However, the a priori search space is too large to be studied in general terms since it is possible to use many types of structures for the function. One can use anything from relatively simple structures, such as linear mappings of the form $u_t = \theta x_t$ as in [Mania et al., 2018], to more complex structures, such as deep neural networks as in [Nguyen and La, 2019]. Thus, to reduce this search space, the policies are defined as functions with a predetermined form given with θ parameters, which end up being the variables with which the problem is solved. So, the optimization problem results in the following:

$$\text{maximize}_{\ \theta} \left[\mathbb{E}_{e_t} \left[\sum_{t=0}^{N} R(x_t, u_t) \right] \right]$$

$$\text{subject to } x_{t+1} = f(x_t, u_t, e_t)$$

$$\text{subject to } u_t = \pi(\theta, x_t)$$

$$(\text{given } x_0),$$

Thus, the development of the problem will consist of determining the policy's predetermined structure and optimizing its performance based on its parameters.

1.3.3. Research Question

How can robotic systems be trained to perform bipedal locomotion tasks so that the obtained performances are optimized in terms of energy consumption and execution speed, despite the limitations imposed by the reality gap?

1.3.4. Complementary Questions

- How can information from human motion patterns be efficiently transferred to bipedal robots?
- How do differences in morphology and initial pose between a robot and a person influence the execution of a locomotion task?
- How can energy efficiency be improved in the execution of a task learned by demonstration?
- How can experiments conducted in simulation and with real robots be integrated to achieve functional and efficient results?

1.4. Objectives

1.4.1. General Objective

To design a robot training strategy based on reinforcement learning that, through experiments on both real robots and simulation, allows for the execution of bipedal robotic locomotion tasks in an optimized manner in terms of energy consumption and execution speed and whose results are feasible for implementation in real robots despite the reality gap.

1.4.2. Specific Objectives

- Develop an imitation learning algorithm that allows robots to perform bipedal locomotion tasks in a manner similar to human demonstrations and is stable not only in simulation but also in real robots.
- Design a reinforcement learning algorithm (based on the quality of the obtained walking patterns) that employs experiments in simulation and with real robots, focusing on bipedal robotic locomotion, to optimize the performance of parametric policies regarding energy consumption and movement speed.

- Determine and implement an efficient mechanism to integrate information from different sources, such as patterns learned by imitation, predetermined models, and/or patterns learned by reinforcement.
- Perform a comparative analysis of the proposed algorithms in both simulation and real bipedal robots, emphasizing the effects of the reality gap and the observed variability in the results.

1.5. Proposed Approach

This work proposes a robot training strategy that allows different robots to learn to walk in optimized ways by implementing artificial intelligence techniques despite the limitations imposed by the reality gap problem. The strategy was developed considering a modular framework considering three main approaches:

- **Non-linear Control Approaches**: Considering that bipedal robots have been able to walk since several years ago, the first approach considered, presented in Chapter 2, entails the modeling of the robots and the control structures employing already well-established techniques in the literature. This first approach allows for determining preliminary workflows and criteria for tackling the whole-body humanoid locomotion problem, which is used as the basis for the following approaches and a comparison reference.
- **Imitation learning**: Taking advantage of the before-mentioned similarity between humans and robots, the second approach considered, presented in Chapter 3, entails the use of imitation learning as an initialization tool for the process aiming to have initial policies that although not optimal for the problem at hand, allow for the determination of feasible human-like gait patterns. We took reference motion capture (MoCap) or preliminary policy results to perform this imitation and adjusted the coefficients or parametric policies to minimize a metric. The metric was a distance function that relates the behavior of the different subjects (whether humans or robots) to a shared space. However, mindlessly minimizing this metric, in general, is not enough since a perfect imitation of another agent does not necessarily guarantee the viability of the policy on the imitator. Therefore, we iteratively adjusted the results to obtain similar and viable results in the target subjects.
- **Reinforcement learning**: As we are interested in obtaining fast and efficient performances, the third approach considered, presented in Chapter 4, entails the use of reinforcement learning as a tool that allows the improvement of different parametric policies, whether randomly initialized or based on other preliminary approaches. To do so, we performed an iterative process where most of the experiments were performed in simulation but always had performance in reality as the ultimate goal.

Once the individual approaches were developed, we obtained several modules that can be implemented stand-alone or in combination with others. Chapter 5 presents some of the different possible combinations of the modules presented aiming to determine the best overall strategy. Finally, Chapter 6 offers the conclusions of the work and some insights for future work.

1.5.1. Comprehensively Tested Approaches

As we introduce various modules, the multitude of potential combinations presents a challenge. Due to practical constraints with our available robots, namely the Darwin Mini (refer to Section 2.3) and MAX-E2 (refer to Section 2.4), we focused our efforts on a subset of feasible combinations during the project.
Our approach included a policy-based method (refer to Section 4.3.2) with a fixed policy. This policy, informed by robotics theory, defines a search space where trajectories align with known problem constraints (refer to Section 4.3.5). Subsequently, we fine-tuned the parameters using three distinct methods:

1. **Manual Tuning:** We initially selected conservative parameters to establish a basic, stable gait as a reference.

2. **Reinforcement Learning Tuning from Aleatory Initial Conditions:** Parameters were optimized from randomly initialized values using a gradient-based method (refer to Section 4.4.2).

3. **Reinforcement Learning Tuning from Imitation Learning Initial Conditions:** Parameters obtained from motion capture data, mapped to the robot space using our proposed Dynamic Retargeting Method (refer to Section 3.5), were further refined using a gradient-based method (refer to Section 4.4.2).

For low-level control in simulations, we employed first-order kinematic control (refer to Section 2.2.1) using hybrid models (refer to Section 2.1) to characterize and estimate the robots' states. In physical experiments, due to hardware limitations, we utilized internal servomotor controllers in position mode to set joint space targets that the robots followed. Additionally, we considered the internal registries of the servomotors and the robots' hybrid models. All implementations on the robots considered transferability limitations and specific strategies (refer to Section 4.5.4).

2. Robots Modeling and Control Models

Humanoid robots have made significant strides in locomotion capabilities over the past few decades from the earliest works on bipedal gait synthesis as [Vukobratovic and Juricic, 1969], there have been several approaches for obtaining stable bipedal robotic locomotion as the use of Hybrid zero dynamics for underactuated robots as [Westervelt et al., 2003], the use of the theory of passive walkers as [Collins et al., 2005], and the use of optimization-based techniques as [Kuindersma et al., 2016]. As we explore the challenges of this type of locomotion, it becomes evident that leveraging the existing modeling and control tools can significantly benefit this thesis's workflow. In this chapter, we delve into the crucial aspects of robot modeling and control, recognizing the value of the previous analytical methods brought to the development of effective locomotion strategies.

While learning models that bypass the need for explicit models or directly map sensor inputs to actuator outputs have gained attention, as in [Lillicrap et al., 2015], it is essential to consider the advantages of employing established models and controllers. Utilizing the knowledge and techniques already developed in the field can enhance our final strategies' sample efficiency and overall performance. The integration of well-established modeling approaches enables us to tap into the wealth of prior knowledge, allowing for a more informed and efficient exploration of bipedal locomotion strategies. For instance, considering formal stability methodologies such as the ZMP point criterion or the Poincaré Return Map criterion, it is possible to have formal guarantees on the robot's stability that are omnipresent in the control community but not so much in artificial learning studies.

An appropriate model serves as the foundation for our research, facilitating meaningful simulations that contribute to bridging the gap between simulation and real-world implementation. Although it is usually unfeasible, a perfect simulator has no reality gap and allows direct transferability. By accurately capturing the kinematics, dynamics, and interactions of humanoid robots, we gain insights into the intricacies of locomotion. Through these simulations, we can narrow the reality gap, refining our strategies and improving the likelihood of successful implementation in the real world.

In this chapter, we embark on a comprehensive exploration of robot modeling and control. We start by discussing the general model of humanoid robots, encompassing state description and sensors, robot actuators, kinematic relationships, kinetic interactions, walking phases, impact models, and stability criteria. Following this, we delve into the various control models employed in humanoid robotics, focusing on low-level control and parametrizing trajectories. Additionally, we provide specific models for two humanoid robots: the Darwin Mini and the

Robotis Engineer Kit (MAX-E2). These models offer detailed insights into their kinematic structure, kinetic parameters, actuation mechanisms, sensing capabilities, and internal PC configurations.
The rest of the Chapter is organized as follows: Section 2.1 presents the different elements of the model considered for the bipedal gait of robots in general, Section 2.2 presents the low-level and trajectory planning strategies for achieving the desired behaviors on the robots, Section 2.3 presents the particularities of the Robot Darwin Mini, and Section 2.4 presents the particularities of the Robot MAX-E2.

2.1. General Model

Following the ideas presented in [Chevallereau et al., 2014], the bipedal robots are modeled as multi-domain hybrid systems of the form:

$$\Sigma = (\Gamma, \mathcal{X}, \mathcal{U}, \mathcal{S}, \Delta, \mathcal{FG}), \tag{2-1}$$

where Γ corresponds to the graph that relates the different phases of the gait, $\mathcal{X}$ corresponds to the state variables of the system (as the joint angles and velocities), $\mathcal{U}$ corresponds to the input vector of the system, $\mathcal{S}$ corresponds to the switching surface that maps the impact conditions, Δ corresponds to the reset map that determines the instantaneous change in the state variables after the impacts and $\mathcal{FG}$ corresponds to the flow equations that determine the evolution of the state variables when there are no impacts. Considering a gait with N_p phases, the resulting model corresponds to

$$\Sigma = \begin{cases} \mathcal{X} = \{\mathcal{X}_i\}_{i=1}^{N_p} : \mathcal{X}_i \subset \mathbb{R}^{n_i} \\ \mathcal{U} = \{\mathcal{U}_i\}_{i=1}^{N_p} : \mathcal{U}_i \subset \mathbb{R}^{m_i} \\ \mathcal{FG} = \{(f_i, g_i)\}_{i=1}^{N_p} : \dot{x}_i = f_i(x_i) + g_i(x_i)u_i \\ \mathcal{S} = \{\mathcal{S}_i^{i+1}\}_{i=1}^{N_p} : \mathcal{S}_i^{i+1} = \{x_i \in \mathcal{X}_\rangle | H_i^{i+1}(x_i) = 0, \dot{H}_i^{i+1}(x_i) < 0\} \\ \Delta = \{\Delta_i^{i+1}\}_{i=1}^{N_p} : x_{i+1}^+ = \Delta_i^{i+1}(x_i^-) \end{cases} \tag{2-2}$$

where $\mathcal{X}_i$ corresponds to the state variables that describe the system in the i-th phase, n_i corresponds to the number of degrees of freedom of the system in the i-th phase, $\mathcal{U}_i$ corresponds to the control variables of the system in the i-th phase, m_i corresponds to the number of actuated variables in the i-th phase, f_i and g_i correspond to the dynamic equations of the system at the i-th phase, $\mathcal{S}_i^{i+1}$ corresponds to the switching surface that takes the system from the i-th phase to the next, H_i^{i+1} is the guard function that parameterizes the switching surface, and Δ_i^{i+1} is the reset maps that sets the state of the system form x_i^- (right before the impact) to x_{i+1}^+ (right after the impact). Note that the phases are considered cyclical so that $N_p + 1 \to 1$.

It is worth mentioning that the system's state must be inferred from the robots' sensors in real environments since this information is not intrinsically available as in the simulation environments.
To characterize each of the components of this model, it is required to consider the components of the system and their dynamics considering:

- The sensors and actuators models.
- The kinematic relationships.
- The kinetic interactions.
- The walking phases and the impact models.

2.1.1. State Description and Sensors

The state of a bipedal robot $\mathcal{X}$ encapsulates different variables related to the robot's pose and its links. It may also contain information related to the forces that act on it or the internal components of the robot as the battery or the actuators. It is worth mentioning that the pose and velocities must be expressed related to a convenient reference frame. Depending on the gait type considered, the state of the bipedal robot can be inferred using only information related to their joint angles as in the case of non-slipping walking gaits where at least one foot is in contact with the floor in each instant, or it may be required to consider additional information as in the case of jogging gaits where the contact with the floor is intermittent.
To determine the state of the bipedal robotic system, it is required to have specific sensors on it. These sensors are all the devices that provide information about the state of the robot and its environment, measuring directly or indirectly certain interest variables. These sensors can be classified into two categories as presented in [Siegwart et al., 2011]. The proprioceptive sensors provide information about the inner state of the robot (as the encoders of each joint or the inertial sensors), and the exteroceptive sensors provide information about the environment and the interactions of the robot with it (as gyroscopes, GPS sensors, or cameras).
Sensor fusion combines data from multiple sensors to obtain a more accurate and complete estimate of a system's state. A single sensor cannot obtain the required accuracy and reliability in many real-world applications. For example, a mobile robot's state estimation in robotics could be based on multiple sensors, such as an inertial measurement unit (IMU) for measuring acceleration and angular rates, encoders for measuring wheel velocities, and cameras for measuring visual features.
Kalman Filters are a widely used method for sensor fusion. They are recursive algorithms that estimate a system's state given uncertain measurements over time. The Kalman Filter works by combining predictions of the system's state based on a mathematical model with measurements of the system's state from multiple sensors. These filters assume that a linear

dynamic system can model the system's state and that the measurements are subject to Gaussian noise. The filter uses a probabilistic model to estimate the system's state and the uncertainty associated with that estimate and computes a weighted average of the predicted state and the measurements, where the relative uncertainties of the predictions and the measurements determine the weights. Although Kalman Filters are initially defined for linear systems, they can be extended to handle nonlinear systems using an extended Kalman Filter or a particle filter.

2.1.2. Robot Actuators

The control variables of the robot $\mathcal{U}$ can be considered from different perspectives. It is possible to consider that the input signals of the system are the set points for the robot's joint angles, the torques applied on each joint, or the voltage applied to the actuator. From a kinetic perspective, it is clear that the physical variables that act on the robot are the torques or the forces of the actuators on each joint. However, the actuators usually have their dynamics, and those forces or torques are not necessarily controlled directly. For instance, the actuators employed on bipedal robots are usually electrically driven servomotors. These servomotors usually have an internal control loop that takes as input the desired position of the joint and employs a simple PID controller to determine the required voltage (and consequently current) delivered to the motor to reach the target position.

2.1.3. Kinematic Relationships

Every robot can be modeled as a kinematic chain (or kinematic tree) where the position and rotation of each interest frame f on the robot can be described as a 4×4 uniform transformation matrix that depends on the articular values q from a base frame to the interest one:

$$T_0^f(q) = \begin{bmatrix} R_{3\times3}(q) & P_{3\times1}(q) \\ 0_{1\times3} & 1 \end{bmatrix}, \tag{2-3}$$

where $T_0^f(q)$ is the transformation matrix that relates the frame f relative to the frame 0, $R(q)$ is the 3×3 rotation matrix, and $P(q)$ is the 3×1 position vector of the frames of interest.
These matrices can be obtained by multiplying the corresponding one to each subsequent frame concerning the previous one up to the root:

$$T_0^f(q) = \prod_{i=1}^{f} T_{i-1}^i. \tag{2-4}$$

Note that even if the rotation matrix has nine elements, several restrictions allow it to be represented in 3 or 4 parameters (as Euler angles or quaternions).

Differential Kinematics

To obtain the linear and angular velocities of the interest, it is possible to forward propagate the velocities of each link from the root frame to any interest one summing those speeds in the appropriate reference frame:

$$\omega_0^i(q,\dot{q}) = \omega_0^{i-1} + \omega_{i-1}^i = \omega_0^{i-1} + R_i\dot{q}_i\hat{z}_i, \tag{2-5}$$

$$v_0^i(q,\dot{q}) = v_0^{i-1} + v_{i-1}^i = v_0^{i-1} + \omega_0^i \times r_{i-1}^i, \tag{2-6}$$

where ω_0^i is the angular velocity of the frame i relative to the origin (the world frame), ω_{i-1}^i is the angular velocity of the frame i relative to the previous frame $i-1$, R_i is the rotation matrix corresponding to the frame i, $\dot{q}_i$ is the angular velocity of the joint i and $\hat{z}_i$ is the direction of the axis where the joint i acts, v_0^i is the linear velocity of the frame i relative to the origin, v_{i-1}^i is the linear velocity of the frame i relative to the previous frame $i-1$, and r_{i-1}^i is the position of the frame i relative to the previous $i-1$.
Alternatively, it is possible to use the Jacobian matrix J_x of the interest variables x (also called task space) concerning q:

$$\dot{x} = \frac{d}{dt}x = \nabla_q x\frac{dq}{dt} = J_x(q)\dot{q}. \tag{2-7}$$

Depending on the task, the interest variables may change. In the bipedal gait scenario, tracking the position and rotation of the balancing foot, the position of the mass center, and the rotation of the torso is often helpful. Note that the Jacobian matrix has a row for each interest variable in x and a column for each joint value in q. Therefore, in general, the matrix is not square but rectangular, and to determine the required $\dot{q}$ for a given $\dot{x}$, it is required to use a pseudo inversion method.
Let $\dot{x_d}$ be a vector of m desired interest variables, $\dot{q_d}$ a vector of n actuated joint values, and $J_x^+(q)$ be the pseudo-inverse of the Jacobian matrix so:

$$\dot{q_d} = J_x^+(q)\dot{x_d}. \tag{2-8}$$

There are many different pseudo-inversion methods. Between them, we highlight the following:

- The Moore-Penrose Pseudo inverse $J_x^+(q) = (J_x^T(q)J_x(q))^{-1}J_x^T(q)$ guarantees the minimization of the inversion error but is very sensitive to numerical problems in the proximity of singularities.
- The damped Moore-Penrose pseudo inverse $J_x^+(q) = (J_x^T(q)J_x(q)+\lambda I)^{-1}J_x^T(q)$ employs the damping factor λ to minimize the norm of the solution as well as the inversion error

with better robustness than the non-damped case (larger λ implies smaller $\dot{q}_d$ at the cost of more significant inversion errors). Note that if $\lambda = 0$, the damped version is equivalent to the original.

- The Extended Jacobian methods $J_x^+ = \begin{bmatrix} J_x(q) \\ \eta(q) \end{bmatrix}^{-1}$ add extra rows to the Jacobian matrix until it is square and, therefore, directly invertible. The added rows must lie in the null space of the original Jacobian (the directions into the articular space that, when moved into them, do not affect the output in the task space). They can be used to fulfill additional tasks, such as reducing the movement from a base pose. This method has been extensively studied [Tevatia and Schaal, 2000]. Still, it can be very demanding computationally since the calculus of the null space of a large Jacobian must be made numerically at each step for matrices larger than 2×2.

Second Order Differential Kinematics

The linear and angular accelerations of the interest frames may be required in some applications. This can be evaluated forward propagating the accelerations of each link concerning the preceding in a similar way as performed with the velocities:

$$\alpha_0^i = \alpha_0^{i-1} + \alpha_{i-1}^i = \alpha_0^{i-1} + R_i \ddot{q}_i \hat{z}_i + \omega_0^i \times R_i \dot{q}_i \hat{z}_i, \tag{2-9}$$

$$a_0^i(q, \dot{q}) = a_0^{i-1} + a_{i-1}^i = a_0^{i-1} + \alpha_0^i \times r_{i-1}^i + \omega_0^i \times v_{i-1}^i, \tag{2-10}$$

where α_0^i is the angular acceleration of the frame i relative to the origin (the world frame), α_{i-1}^i is the angular acceleration of the frame i relative to the previous frame $i-1$, a_0^i is the linear acceleration of the frame i relative to the origin, and a_{i-1}^i is the linear acceleration of the frame i relative to the previous frame $i-1$.
Alternatively, it is also possible to obtain those accelerations, considering the time derivative of the Jacobian matrix of the transformation. However, there is no standard form for the Hessian $H(q)$ for multi-variable vector-valued functions. In this work, we will consider the definition as an $m \times n \times n$ tensor:

$$\ddot{x} = \dot{J}(q)\dot{q} + J(q)\ddot{q} = \left(\nabla_q J(q) \frac{dq}{dt} \right) \frac{dq}{dt} + J(q) \frac{d^2 q}{dt^2} = H(q)\dot{q}^2 + J(q)\ddot{q}. \tag{2-11}$$

2.1.4. Kinetic Interactions

Once the full kinematic relationships of the robot are determined, it is necessary to determine the required torques and forces that the servomotors and the reaction forces must exert to validate the limits of the actuators and the hypothesis onto the reaction forces (tangential

force of the ground reaction force within the friction cone and torques within the permissible bounds). This process of obtaining the torques and reaction forces based on the angular position, angular velocity, angular acceleration, and inertial properties of the robot is known as the inverse dynamic problem and is usually formulated as follows:

$$M(q)\ddot{q} + C(q, \dot{q}) = \tau + F_c, \tag{2-12}$$

where $M(q)$ corresponds to the inertia matrix of the robot, $C(q, \dot{q})$ includes the centripetal, Coriolis, and gravitational forces that act on the robot, τ corresponds to the generalized torques input, and F_c corresponds to the contact forces with the ground.
On the other hand, the forward dynamic problem consists of obtaining the angular acceleration based on the angular position, angular velocity, inertial properties, applied torques, and reaction forces and is usually formulated as follows:

$$\ddot{q} = M(q)^{-1}(\tau + F_c - C(q, \dot{q})). \tag{2-13}$$

Note that the inertia matrix is definite positive, so its inverse always exists.
There are several methods to obtain a robot's kinetic relationships; the two most common approaches are the Newton-Euler and Euler-Lagrange methods.

Euler-Lagrange Method

The Euler-Lagrange approach presented in [Mark W. Spong, 2020] considers an energy-based approach that, following the least action principle, states that the change in the system's energy is influenced by the input forces acting on the system. In this approach, the Lagrangian $\mathcal{L}$ of the system relates the kinetic energy of the system $\mathcal{K}$ and the potential energy of the system $\mathcal{U}$

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q) \tag{2-14}$$

and its application results in an inverse kinematic model of the form:

$$\tau = \frac{d}{dt}\frac{\partial}{\partial \dot{q}}\mathcal{L}(q, \dot{q}) - \frac{\partial}{\partial q}\mathcal{L}(q, \dot{q}). \tag{2-15}$$

This approach considers the robot as a whole and tends to be preferred when there are non-rigid elements on the robot and when the robot structure is complex. However, when the robot has several degrees of freedom, its use can become unfeasible as the required derivatives (even with symbolic math toolboxes) may demand too many resources for their calculus.

Newton-Euler Method

The Newton-Euler approach presented in [Mark W. Spong, 2020] considers each link as an individual entity, evaluates its linear and angular momentum

$$p_i = m_i v_0^i, \tag{2-16}$$

$$L_i = I_i \omega_0^i, \tag{2-17}$$

where p_i is the linear momentum of the link i, m_i is the mass of the link i, L_i is the angular momentum of the link i, and I_i is the inertia tensor of the link i.
Based on the second Newton's law:

$$\frac{dp_i}{dt} = m_i a_0^i + I_i \alpha_0^i = \sum_{j \in N_i} f_i^j, \tag{2-18}$$

$$\frac{dL_i}{dt} = I_i \alpha_0^i + \omega_0^i \times I_i \omega_0^i = \sum_{j \in N_i} \tau_i^j, \tag{2-19}$$

where f_i^j is the force that the link j exerts on the link i, where τ_j^i is the torque that the link j exerts on the link i, and N_i is the set of links connected to the link i.
This set of equations allows for determining the dynamic behavior of each link and, by extension, the dynamic equations of the whole robot. To do so, it is often helpful to represent the forces of each element in free-body diagrams as the ones presented in Appendix A. Note that even if each link is considered an individual entity, the reaction forces couple the dynamics of the set, and the velocities, accelerations, forces, and torques must be propagated through the kinematic tree for the calculus.

Recursive Newton-Euler Method

Based on [Siciliano and Khatib, 2016], it is possible to obtain the inverse dynamics of the robot following a recursive process that consists of several steps:

1. Determine the effective forces and torques that act onto each link $\sum_{j \in N_i} f_i^j = m_i a_0^i$ and $\sum_{j \in N_i} \tau_i^j = I_{CM}^i \alpha_0^i + \omega_0^i \times I_{CM}^i \omega_0^i$.

2. From the end of the kinematic chains towards the root, determine the contribution of force and torque applied by the joint for the dynamic balance $f_i = \sum_{j \in N_i} f_i^j - f_i^{i+1}$ and $\tau_i^{i-1} = \sum_{j \in N_i} \tau_i^j - \tau_i^{i+1} - r_i^{CM} \times f_i^{i+1} - r_{i+1}^{CM} \times f_{i+1}$ (each joint provides the wrench for its acceleration and the following link).

3. Project the torque of the joint into the actuator axis to determine its effective contribution (the mechanical structure makes the non-projected part) $\tau_i = \tau_i^{i-1} R_i \hat{z}_i$.

2.1.5. Walking Phases and Impact Models

The switching surfaces S_i^{i+1} and the corresponding reset map Δ_i^{i+1} take the state previous to the impact x_i^- to the state after the impact $x_{i+1}^+ = \Delta_i^{i+1}(x_i^-)$. When the contact with the

floor changes, the walking phase changes accordingly. Different possible transition models depend on the robot model and the collision assumptions. These assumptions correspond to the duration of the double support phase (instantaneous or non-instantaneous) and the type of collision with the floor when stepping (soft or hard collision). Note that having instantaneous double support with soft collisions is impossible.

Instantaneous double support with rigid collision

The gait is decomposed into two phases, a simple support phase where a foot stands rigidly on the floor while the other swings forward, and an instantaneous double support phase where both feet exchange their roles symmetrically. In this case, the collision with the floor and the corresponding reaction forces are considered instantaneous and are reflected as a change in the direction of the movement depending on the landing velocity respecting the momentum conservation laws and the continuity of the position profiles $x_{i+1}^{+} = \Delta_i^{i+1}(x_i^{-})$. This model is frequently used for underactuated walkers as in [Arcos-Legarda et al., 2019] and provides a promising approach for robots with few links or 2D walkers. However, 3D walkers with several links can be very challenging to tackle with this approach as their kinetic model is expensive to calculate. Hence, their extended kinetic model is even more expensive.

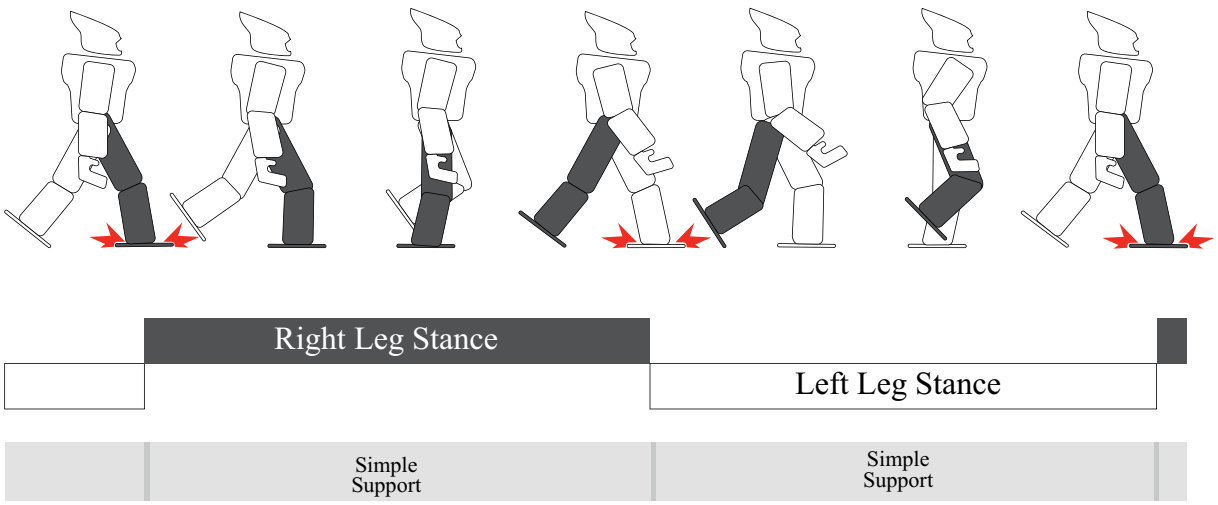


Figure 2-1.: Humanoid gait with instantaneous double support - Most of the time, the robot is supported by a single foot and switches the support instantaneously with rigid collisions

One possible way to evaluate the reset map $x_i^{+} = \Delta_i^{i+1}(x_i^{-})$ presented in [Westervelt et al., 2003] proposes the analysis of the conservation of momentum and the kinematic constraints that the robot must fulfill to perform a successful step. Initially, it is required to extend the kinetic

model adding additional degrees of freedom for the base of the robot (the support foot for our purposes). During this process, we obtain the extended system model:

$$M_e(q_e)\ddot{q_e} + C_e(q_e, \dot{q_e}) = \tau_e, \tag{2-20}$$

where M_e corresponds to the extended inertia matrix of the robot, C_e includes the centripetal, Coriolis, and gravitational forces that act on the robot, and τ_e corresponds to the extended generalized forces input.
With this extended model, it is possible to formulate the momentum equation of the system before and after the collision considering the angular velocities before the impact $\dot{q_e^-}$ and after it $\dot{q_e^+}$. Considering the moment conservation, the change in this momentum depends on the impact force F_{impact} that acts onto the balancing foot so that:

$$M_e(q_e^+)\dot{q_e^+} - M_e(q_e^-)\dot{q_e^-} = F_{impact}. \tag{2-21}$$

On the other hand, it is required that the balancing foot does not move after the impact (no sliding and no rebound), which can be expressed as: $v_{SF}^+ = J_{BF}^-(q_e^+)\dot{q_e^+} = 0$ where v_{SF}^+ corresponds to the velocity of the former balancing foot (now supporting) and J_{BF}^- corresponds to the Jacobian corresponding to this velocity before the collision. Note that $q_e^+ = q_e^-$ as the position is always continuous.
With these two equations, it is possible to formulate the system of equations:
$$\begin{bmatrix} M_e(q^-) & -J_{BF}^-(q^-)' \\ J_{BF}^-(q^-) & 0_{2\times 2} \end{bmatrix} \begin{bmatrix} \dot{q_e^+} \\ F_{impact} \end{bmatrix} = \begin{bmatrix} D_e(q^-)\,\dot{q_e^-} \\ 0_{2\times 1} \end{bmatrix}.$$
Whose solution allows us to determine the post-impact velocities in the reset map.
Finally, the last step for formulating the reset map implies the role swap between the former balancing foot that becomes supporting and vice versa. For 2D walkers, the process only requires the direct swapping of the corresponding variables, but for 3D walkers, it is necessary to consider mirror symmetry concerning the sagittal plane.

Non-instantaneous double support with rigid collision

Again, the gait is decomposed into two phases. Although, in this case, the double support phase is not considered instantaneous. In this case, the collision with the floor is similar to the previous case (considering the momentum conservation and null velocity of the former balancing foot). Still, the former support foot's velocity must remain zero after the transition. In this case, it is required to consider not only the transition map Δ_i^{i+1} but also the additional constraints over the kinematic model during the double support phase, considering that the model loses multiple degrees of freedom as the position and rotation of both feet are fixed in this phase.

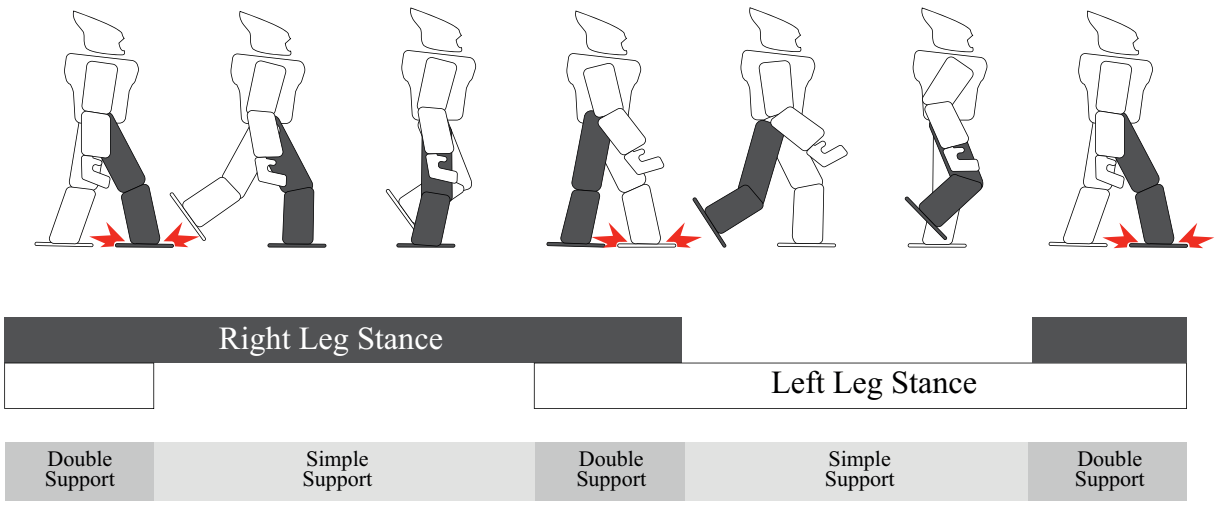


Figure 2-2.: Humanoid gait with non-instantaneous double support - The robot alternates between simple and double support phases with rigid collisions

Considering the closed kinematic chain from the non-instantaneous double support phase is challenging. An alternative is to consider the open chain kinematics considering the position of the former balancing foot as a variable to control fixing its position to maintain its current value. Regarding the kinetics for the closed kinematic chain, in [Siciliano and Khatib, 2016], it is mentioned that it is usual to initially consider the open kinematic chain and then enforce the kinematic constraints. However, these calculi tend to be hard to compute and/or to depend on dynamical parameters that are often not well known, so how to perform them efficiently remains an active open issue for fast robots.

Non-instantaneous double support with soft collision

In rigid collision cases, the reset map Δ_i^{i+1} may be challenging to model and evaluate accurately. As an alternative, in some cases, it is possible to reduce the velocity of the balancing foot so that the effects of the collision get less effect onto the robot. In the limit case, if the velocity of the balancing foot tends to zero at the moment of the contact (a soft collision). Under this supposition $F_{impact} \approx 0$, the robot's momentum is not affected, and the transition map becomes much more straightforward as it is only required to consider the symmetrical remapping of angular positions and velocities. Although this approach usually reduces the overall velocity and efficiency of the gait, it also simplifies the analysis and reduces the impacts the robot must endure during the gait.

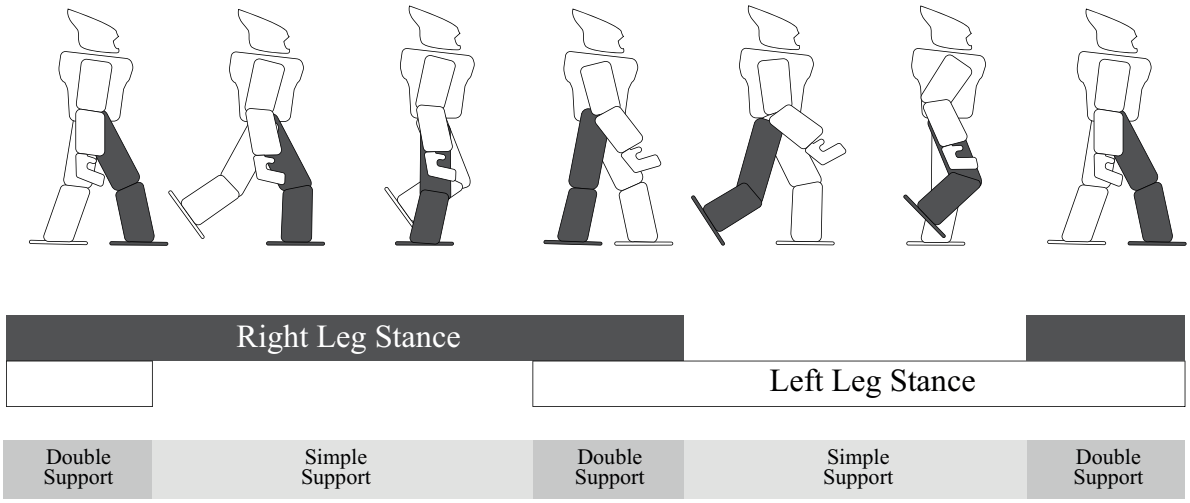


Figure 2-3.: Humanoid gait with non-instantaneous double support - The robot alternates between simple and double support phases with soft collisions

Alternative models

More phases can be considered depending on the foot's structure and the gait's nature. For instance, when we humans walk, the movement can be decomposed into 8 phases [Loudon et al., 2008]:

- Initial Contact
- Loading Response
- Midstance
- Terminal Stance
- Pre swing
- Initial Swing
- Mid Swing
- Late Swing

On the other hand, it is possible to consider models without phases at all. Those models consider the so-called floating base models that do not segment the gait in phases but consider a unique model where the contacts appear as external forces that affect the overall dynamics.

2.1.6. Stability Criteria

Stability is crucial in studying dynamical systems, and bipedal robotic locomotion is no exception. In the context of bipedal robots, stability is defined as the ability of the robot to maintain balance while moving. Several stability criteria are commonly used in bipedal robotic locomotion, including the center of mass projection, the zero moment point (ZMP), and the hybrid zero dynamics - Poincare's return maps.

Center of Mass Projection

The center of mass projection stability criterion is based on the projection of the robot's center of mass onto the ground. If the center of mass is projected within the support polygon, defined by the area where the robot's feet are in contact with the ground, the robot is considered stable.

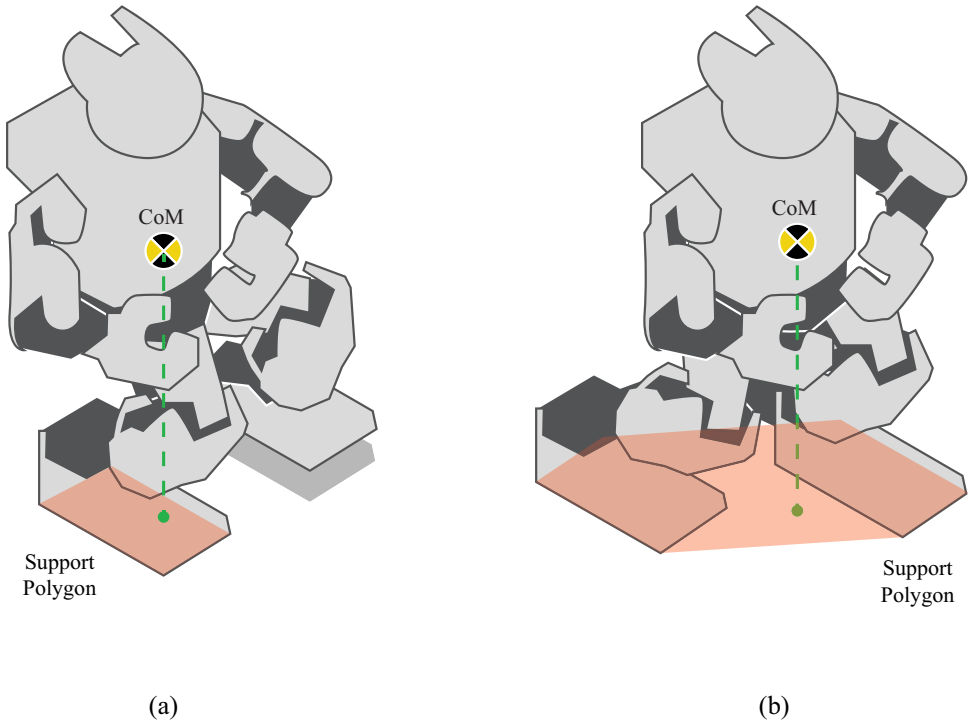


Figure 2-4.: Center of Mass Projection Stability Criterion, as long as the CoM projection lies within the support polygon, the robot is statically stable - (a) In the simple support phase, the Support Polygon corresponds to the area of the sole. (b) In the double support phase, the Support Polygon corresponds to the convex hull of the area of both feet.

This criterion is widely used in bipedal robotic locomotion, as it is relatively simple to

calculate and provides a good measure of the robot's stability. However, this criterion only applies to quasi-static gaits as the accelerations of the robot may cause the robot falls even with the center of mass projection within the support polygon.

Zero Moment Point - ZMP

The ZMP stability criterion is another widely used measure of stability in bipedal robotic locomotion. The ZMP is a point on the ground where the net moment applied to the robot in the axes perpendicular to the ground is zero. It is calculated based on the robot's center of mass, the external forces acting on the robot, and the dynamics of the robot. The robot is considered stable if the ZMP is inside the support polygon. In [Tedrake, 2022], the calculus of the ZMP is reduced to:

$$x_{ZMP} - x_{CoM} = \frac{z_{CoM}\ddot{x}_{CoM} - I\ddot{\theta}}{m\ddot{z}_{CoM} + mg}, \tag{2-22}$$

$$y_{ZMP} - y_{CoM} = \frac{z_{CoM}\ddot{y}_{CoM} - I\ddot{\theta}}{m\ddot{z}_{CoM} + mg}, \tag{2-23}$$

where x_{ZMP} and y_{ZMP} are the X and Y components of the robot's ZMP; x_{CoM}, z_CoM, and y_{CoM} are the X, Y, and Z components of the position of the robot's center of mass; I is the robot's centroidal inertia tensor; $\ddot{theta}$ is the robot's centroidal angular acceleration; m is the mass of the robot; and g is the gravitational acceleration.

Additionally, it is usual that this expression gets further simplified by fixing the height of the center of mass to a constant height h and minimizing the angular acceleration $\ddot{\theta}$. With these considerations, the final expression results in the affine equations

$$x_{ZMP} - x_{CoM} = \frac{h}{mg}\ddot{x}_{CoM}, \tag{2-24}$$

$$y_{ZMP} - y_{CoM} = \frac{h}{mg}\ddot{y}_{CoM}. \tag{2-25}$$

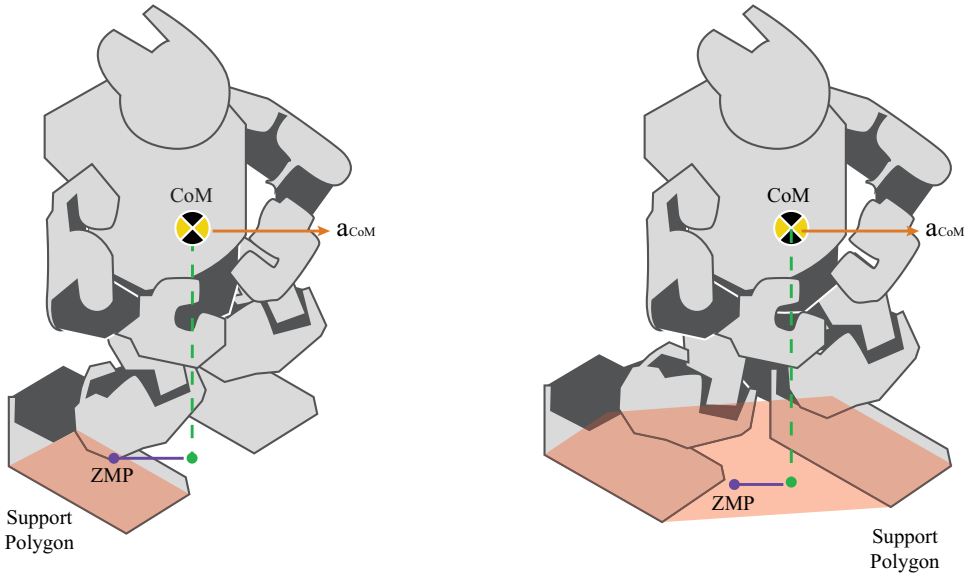


Figure 2-5.: Zero Moment Point Stability Criterion, as long as the CoM projection lies within the support polygon, the robot is dynamically stable (even if the CoM projection criterion is not fulfilled) - (a) In the simple support phase, the Support Polygon corresponds to the area of the sole. (b) In the double support phase, the Support Polygon corresponds to the convex hull of the area of both feet.

This criterion applies to dynamic gaits and is restricted to robots that have feet with non-zero areas. It is worth noting that the ZMP criterion converges to the CoM projection criterion as the acceleration goes to zero.

Poincare's Return Maps

Poincare's return map criteria is a valuable tool for analyzing the stability of limit cycles in dynamic systems. The primary concept involves parametrizing a Poincare section in the system's state space. This Poincare section is arbitrarily chosen considering that it should be crossed by the system's state x during its evolution in time. The system's state at those crosses at the discrete instants k is denoted by P_k. The relation between the points where the crossing occurs, i.e., $P_{k+1} = T(P_k)$, is a discrete Poincare Return Map. If the system is stable, the Poincare Return map will get the points closer at each new cross (whether they have or not a constant period) confirming the stability of the system, or if the points get farther at each new cross, then the system is considered to be unstable.

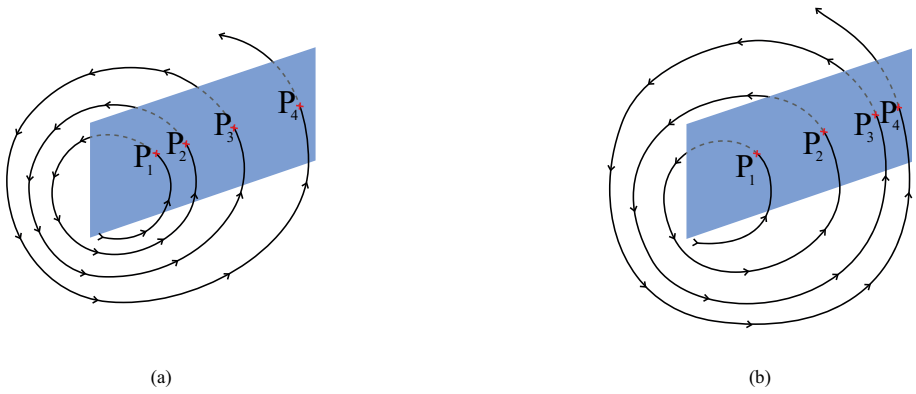


Figure 2-6.: Poincare Return Maps - (a) unstable system: the intersections between the system state in black and the Poincare Section in blue get further and further. (b) stable system: the intersections between the system state in black and the Poincare Section in blue get closer and closer.

In the case of bipedal robots, it is possible to consider any of the switching surfaces of the model as the Poincare section. In particular, it is possible to consider the surface corresponding to the strike of the balancing foot onto the ground so that the Poincare Return Map corresponds to the evolution of the system's state at the end of each step. By analyzing this Poincare Return Map, we can verify the stability of the robot's gait patterns. This analysis can be made considering that the return map is a lineal mapping

$$T(P_k) = MP_k \tag{2-26}$$

where M is a matrix with constant coefficients. The value of M coefficients is estimated through experiments (usually simulations), and if all the eigenvalues of M have a norm ≤ 1, then the system will be considered stable.

2.2. Control Models

Several control methods can be applied to humanoid robots to ensure the tracking of desired trajectories $q \rightarrow q_d$. To this purpose, we must analyze how the low-level control achieves the tracking and how to properly determine soft references at a higher level. The choice of the low-level control model depends on the required level of performance and the available hardware. In this section, we discuss four different approaches: using the internal servomotor controllers (setting q_d references), using first-order kinematic control employing the Jacobian matrix (setting $\dot{q}_d$ references), using second-order kinematic control employing the Hessian

matrix (setting $\ddot{q}_d$ references), and employing torque control through feedback linearization (setting τ references). On the other hand, we considered two main options to determine the soft references to be followed—first, using Bézier Polynomials for parametrizing soft trajectories, and second, ZMP-based trajectories. When used correctly, both of them allow us to determine soft trajectories to be followed.

2.2.1. Low-level control

Internal Servomotor Controllers

One approach to controlling humanoid robots is to rely on the internal servomotor controllers (as the ones presented in sections 2.3.3 and 2.4.3) so that $q \rightarrow q_d$. These controllers are typically embedded within the robot's hardware, translating the desired position commands into motor torques. Here are some advantages and disadvantages of using internal servomotor controllers:

Advantages

- **Simplicity**: The internal servomotor controllers are already integrated into the robot's hardware, making them readily available for control. They are designed specifically for the robot's actuators and often come with predefined control modes, making it easier to implement and configure the control strategy.

- **Real-Time Responsiveness**: The internal servomotor controllers operate directly within the servomotors, allowing for high control frequencies and fast response times. The control signals are generated and executed within the robot's hardware, resulting in minimal delays and low-latency control. This real-time responsiveness is crucial for achieving precise and dynamic movements in locomotion tasks.

Disadvantages

- **Limited Flexibility**: The internal servomotor controllers often have limited control algorithms and customization flexibility. They may not support advanced control techniques or allow fine-grained control parameter adjustments. This limitation can restrict the ability to achieve highly specialized control behaviors.

- **Model Mismatch**: The internal controllers are typically designed based on simplified robot dynamics models. While these models capture the essential dynamics, they may not capture all the intricacies and complexities of the robot's behavior. This model mismatch can lead to suboptimal performance, especially in challenging locomotion tasks that require precise control.

First Order Kinematic Control - Jacobian Pseudo-inverse

Kinematic control is another approach to controlling humanoid robots that do not rely on position control or torque control but instead focuses on velocity control ($\dot{q} \rightarrow \dot{q_d}$). It involves generating and tracking desired joint velocities to achieve the desired robot motion. To do so, this type of controller employs the de Jacobian matrix pseudo-inverse to determine the required joint velocities for reducing the error in the task space e_x:

$$e_x = x_d - x \tag{2-27}$$

$$\dot{e_x} = \dot{x_d} - \dot{x} \tag{2-28}$$

$$\dot{e_x} = \dot{x_d} - J_x\dot{q} \tag{2-29}$$

Setting $\dot{q} = \dot{q_d} = J_x^+(\dot{x_d} + K_p e_x)$

$$\dot{e_x} = -K_p e_x. \tag{2-30}$$

As long as the proportional gain K_p is greater than 0, the error dynamic converges exponentially to 0, obtaining the desired tracking.
Here are some advantages and disadvantages of using kinematic control:
Advantages

- **Simplified Control**: Kinematic control simplifies the control problem by directly controlling joint velocities rather than dealing with complex dynamics or torque calculations. This can reduce computational complexity and enable faster control execution, making it suitable for real-time control applications.
- **Hardware Compatibility**: Since kinematic control operates at the velocity level, it can be compatible with a wide range of robotic hardware and actuators. It is not as dependent on specific servo motor controllers or hardware characteristics, making it easier to implement on different robot platforms.
- **Reduced Sensitivity to Modeling Errors**: Kinematic control relies primarily on geometric relationships and kinematic equations, which are typically more robust to modeling errors than dynamic models used in torque control or feedback linearization. This reduced sensitivity to modeling errors can make kinematic control more robust and less prone to performance degradation in the presence of uncertainties.

Disadvantages

- **Limited Dynamic Capability**: Kinematic control ignores the dynamic aspects of the robot's motion, focusing solely on joint velocities. This limitation can restrict the robot's ability to handle dynamic tasks requiring accurate force interactions or environmental compliance.
- **Inability to Handle Underactuation**: If the humanoid robot has underactuated joints (joints with fewer actuators than degrees of freedom), kinematic control alone may not achieve desired motions. Additional control strategies, such as operational space control or task prioritization, may be necessary to compensate for the underactuation and achieve the desired control objectives.
- **Lack of Force Control**: Kinematic control does not directly address force control or physical interaction with the environment. This can limit the robot's ability to perform tasks that require force-sensitive manipulation or contact tasks, where precise force control is essential.

Second Order Kinematic Control - Hessian Matrix

Kinematic control based on the Hessian matrix considers the second-order derivatives of the robot's kinematic equations. This approach approximates the system's dynamics more closely than traditional kinematic control. However, it comes with the computational cost of calculating the Hessian and Jacobian matrix pseudo-inverse:

$$\ddot{x} = \left(\frac{\partial^2 x}{\partial q^2}\frac{dq}{dt}\right)\frac{dq}{\mathrm{dt}} + \frac{\partial x}{\partial q}\frac{d^2 q}{\mathrm{dt}}, \tag{2-31}$$

where $\frac{\partial^2 x}{\partial q^2}$ corresponds to the Hessian matrix of the robot H_x. Note that there is no standard form for the Hessian matrix, and in this work, we will consider it as an $[m \times n \times n]$ matrix:

$$\ddot{\mathbf{x}} = H_x \dot{q}^2 + J_x \ddot{q}. \tag{2-32}$$

With these dynamics at hand, we can evaluate the error dynamics:

$$e_x = x_d - x \tag{2-33}$$

$$\dot{e_x} = \dot{x_d} - \dot{x} \tag{2-34}$$

$$\ddot{e_x} = \ddot{x_d} - \ddot{x} \tag{2-35}$$

$$\ddot{e_x} = \ddot{x_d} - \ddot{x}H_x\dot{q}^2 - J_x\ddot{q} \tag{2-36}$$

setting $\ddot{q} = \ddot{q_d} = J_x^+(\ddot{x_d} - \ddot{x}H_x\dot{q}^2 - K_p e_x - K_d\dot{e_x})$

$$\ddot{e_x} = -K_p e_x - K_d\dot{e_x}. \tag{2-37}$$

That corresponds to second-order dynamics, which with the proper tuning of the proportional gain K_p and the derivative gain K_d, can lead to exponential convergence in the error dynamics obtaining the desired tracking.
Here are some advantages and disadvantages of using kinematic control based on the Hessian matrix:

Advantages

- **Improved Dynamic Accuracy**: Using second-order derivatives, kinematic control based on the Hessian matrix accurately represents the robot's dynamic behavior. This can improve tracking performance, especially in tasks requiring precise motion control or complex dynamic interactions.
- **Enhanced Stability**: The inclusion of higher-order derivatives can contribute to improved stability properties of the control system. By considering the system's dynamics more comprehensively, kinematic control based on the Hessian matrix can offer increased robustness and stability margins, leading to more reliable and safer robot operation.

Disadvantages

- **High Computational Cost**: Calculating the Hessian matrix involves complex mathematical operations, which can be computationally expensive. The size of the matrix grows with the number of degrees of freedom, resulting in a significant computational burden, particularly for complex humanoid robots with many joints. This computational cost can limit the real-time applicability of the control method, potentially leading to delays or reduced control performance.
- **Increased Model Complexity**: Kinematic control based on the Hessian matrix requires an accurate model of the robot's kinematics and dynamics. Developing and maintaining such a model can be challenging, as it often involves complex mathematical derivations and accurate estimation of system parameters. Any discrepancies or uncertainties in the model can introduce errors in the control calculations and affect overall system performance.

- **Sensitivity to Modeling Errors**: Kinematic control based on the Hessian matrix highly depends on the model's accuracy and the assumptions' validity. Inaccuracies or uncertainties in the model can lead to suboptimal or unstable control behavior. Robust model identification and parameter estimation techniques are necessary to mitigate the effects of modeling errors and ensure reliable control performance.

Feedback Linearization

Another approach to controlling humanoid robots is to employ non-linear control methods to directly control the torque applied to the servomotor, such as feedback linearization. Feedback linearization is a technique that transforms the non-linear dynamics of the robot into a linear system, allowing for the application of linear control techniques. For robots, it is possible to simplify the dynamics of the system to independent double integrators for each degree of freedom as long as the kinetic model matrices $M(q)$ and $C(q, \dot{q})$ are accurately known. Let v be the new input for the desired behavior setting the generalized torque input as:

$$\tau = C(q, \dot{q}) + M(q)v \tag{2-38}$$

then the feedback-linearized system takes the form:

$$\ddot{q} = v \tag{2-39}$$

This is valid as long as the constraints over the input and contact forces are fulfilled, allowing the use of any appropriate linear control technique, such as PID controllers or sliding mode controllers. Note also that this torque controller can be employed with any of the previous strategies using it to enforce the desired joint positions, velocities, or accelerations:

- For position control $v = -K_{p\tau}(q_d - q) - K_{d\tau}(\dot{q_d} - \dot{q})$, allows the tracking.
- For velocity control $v = -K_{p\tau}(\dot{q_d} - \dot{q})$, allows the tracking.
- For acceleration control $v = \ddot{q_d}$, allows the tracking.

where $K_{p\tau}$ corresponds to an appropriately tuned proportional torque gain and $K_{d\tau}$ corresponds to an appropriately tuned derivative gain.
Here are some advantages and disadvantages of using feedback linearization:
Advantages

- **Model-Based Control**: Feedback linearization relies on an accurate model of the robot's dynamics. Using this model makes possible to design control laws that explicitly consider the robot's underlying physical properties. This model-based approach can improve control performance, especially when dealing with complex locomotion tasks.

- **Control Flexibility**: Feedback linearization provides more flexibility in terms of control design. It allows for applying various control techniques and strategies, such as optimal control, robust control, or adaptive control. This flexibility enables the development of control strategies that can handle different operating conditions and adapt to varying task requirements.

Disadvantages

- **Complexity and Computational Demands**: Feedback linearization requires an accurate model of the robot's dynamics, which can be complex and time-consuming to develop. The process of modeling the robot's dynamics and deriving the linearizing transformation can be challenging, particularly for complex humanoid robots with multiple degrees of freedom. Additionally, the control algorithms based on feedback linearization often involve computationally intensive calculations, which can pose challenges in real-time control implementation.

- **External Dependency**: Non-linear control methods, such as feedback linearization, often rely on external computation and communication channels to implement control algorithms. The control commands and calculations are typically executed on an external computer and then communicated to the robot's actuators. This introduces additional latency due to the need for data transmission and processing, which can limit the real-time responsiveness of the control system. The external dependency on the computer and communication channels can be a potential source of delays and instability in control execution, particularly in scenarios that require high-speed and precise movements.

- **Tuning and Robustness**: Feedback linearization relies heavily on the model's accuracy for control design. Any discrepancies or uncertainties in the model can lead to performance degradation or instability. Achieving robust and reliable control with feedback linearization often requires careful model identification, parameter tuning, and robust control techniques to handle uncertainties and disturbances.

Ultimately, the control model choice depends on the locomotion task's specific needs and the available resources. Internal servomotor controllers excel in tasks that require a high-speed and low-latency response, while feedback linearization provides more sophisticated model-based control with increased performance potential. On the other hand, Kinematic control offers simplicity and hardware compatibility but may be limited in handling specific dynamic and force-sensitive tasks.

2.2.2. Parametrizing Trajectories

Bézier Polynomials

Bézier polynomials provide a flexible and intuitive approach for parametrizing smooth trajectories that low-level controllers can follow. These polynomials are widely used in computer graphics and animation due to their ability to generate aesthetically pleasing curves. In controlling robots, Bézier polynomials offer a convenient representation for defining trajectories that exhibit desired properties, such as smoothness and customizable shape. Works as [Simba et al., 2016] show applications of this kind of trajectory.

Bézier Polynomial Definition and Derivatives
A Bézier polynomial of degree k is defined as:

$$B(s) = \sum_{i=0}^{k} \binom{k}{i} (1-s)^{k-i} s^i P_i, \tag{2-40}$$

where s is the parameter ranging from 0 to 1, and $\binom{k}{i}$ is the binomial coefficient. The control points P_i can be interpreted as the positions of the robot's end-effector or key points along the trajectory.
The derivatives of the Bézier polynomial relative to s can be computed as follows:

$$\frac{dB(s)}{ds} = k \sum_{i=0}^{k-1} \binom{k-1}{i} (1-s)^{k-1-i} s^i (P_{i+1} - P_i) \tag{2-41}$$

$$\frac{d^2B(s)}{ds^2} = k(k-1) \sum_{i=0}^{k-2} \binom{k-2}{i} (1-s)^{k-2-i} s^i (P_{i+2} - 2P_{i+1} + P_i). \tag{2-42}$$

These derivatives provide information about the velocity and acceleration of the trajectory, which can be used to control the robot's motion. However, we want to use the Bézier Polynomials in arbitrary time intervals, therefore for each time interval of interest j, we define $s^j(t) = \frac{t - t_0^j}{t_f^j - t_0^j}$ so that:

$$q_d(t) = B(s(t)), \tag{2-43}$$

$$\dot{q_d}(t) = \frac{dB(s(t))}{dt} = \frac{dB(s(t))}{ds} \frac{ds(t)}{dt}, \tag{2-44}$$

$$\ddot{q_d}(t) = \frac{d^2B(s(t))}{dt^2} = \frac{d^2B(s(t))}{ds^2} \left(\frac{ds(t)}{dt} \right)^2, \tag{2-45}$$

where t_0^j is the initial time of the interval j, t_f^j is the final time of the interval j, and t is the time where the polynomial is evaluated.

Fixing Position, Velocity, and Acceleration at Extreme Points
One advantage of using Bézier polynomials is the ability to fix position, velocity, and acceleration at specific extreme points of the trajectory. This feature allows precise control of the robot's motion characteristics at critical moments. We must connect multiple Bézier polynomials in arbitrary consecutive intervals (that we will index with j) so that their first and second derivatives are continuous.

Position: To fix the position at the start $B^j(t_0^j)$ and end $B^j(t_f^j)$ points of any segment j, the control points P_0^j and P_k^j can be set to the desired positions. By doing so, the trajectory will pass through these points, ensuring the desired positional accuracy. Moreover, if we want that two consecutive Bézier polynomials $B^j(t)$ and $B^{j+1}(t)$ are continuous at the transition time $t_f^j = t_0^{j+1}$, we have to guarantee that

$$B^j(t_f^j) = B^{j+1}(t_0^{j+1}), \tag{2-46}$$

which is trivially obtained setting $P_k^j = P_0^{j+1}$.

Velocity: Similarly, to specify the velocity at the start and end points, the derivatives of the Bézier polynomial can be evaluated at transition time according to:

$$\frac{dB^j(t_0^j)}{dt} = \frac{k}{t_f^j - t_0^j}(P_1^j - P_0^j), \tag{2-47}$$

$$\frac{dB^j(t_f^j)}{dt} = \frac{k}{t_f^j - t_0^j}(P_k^j - P_{k-1}^j). \tag{2-48}$$

Then, we require the final velocity at the end of the segment j to be equal to the initial velocity at the start of the segment $j+1$, i.e., $\dot{B^j}(t_f^j) = \dot{B^{j+1}}(t_0^{j+1})$ which is obtained through:

$$\frac{k}{t_f^j - t_0^j}(P_k^j - P_{k-1}^j) = \frac{k}{t_f^{j+1} - t_0^{j+1}}(P_1^{i+1} - P_0^{j+1}). \tag{2-49}$$

Note that we determined $P_k^j = P_0^{j+1}$ in the position setting so that we must determine the P_{k-1}^j and P_1^{j+1} in this step so that the transition velocity corresponds to the desired value.

Acceleration: Finally, to specify the acceleration at the start and end points, the second derivatives of the Bézier polynomial can be evaluated at transition time according to:

$$\frac{d^2B^j(t_0^j)}{dt^2} = \frac{k(k-1)}{(t_f^j - t_0^j)^2}(P_0^j - 2P_1^j + P_2^j), \tag{2-50}$$

$$\frac{d^2B^j(t_f^j)}{dt^2} = \frac{k(k-1)}{(t_f^j - t_0^j)^2}(P_{k-2}^j - 2P_{k-1}^j + P_k^j). \tag{2-51}$$

Then, we require the final acceleration at the end of the segment j to be equal to the initial acceleration at the start of the segment $j+1$, i.e., $\ddot{B^j}(t_f^j) = \ddot{B^{j+1}}(t_0^{j+1})$ which is obtained through:

$$\frac{k(k-1)}{(t_f^j - t_0^j)^2}(P_{k-2}^j - 2P_{k-1}^j + P_k^j) = \frac{k(k-1)}{(t_f^{j+1} - t_0^{j+1})^2}(P_0^{j+1} - 2P_1^{j+1} + P_2^{j+2}). \tag{2-52}$$

Note that we determined the values for P_k^j, P_{k-1}^j, P_0^{j+1}, and P_1^{j+1} in the previous steps so that we must determine P_{k-2}^j and P_2^{j+1} to achieve the desired transition acceleration.

Number of Segments, Minimum Order k and Additional Coefficients: Depending on the number of control points that we required m, we will require $m-1$ segments described by Bézier polynomials to connect them all with the imposed constraints. To set the position, velocity, and acceleration at each segment's beginning and end, we require a minimum order of $k = 6$ for each one. In the case of $k = 6$, the solution for the coefficients P^j is uniquely defined, and no further changes can be made, but for $k > 6$, there will be free coefficients that can be arbitrarily defined. Those extra coefficients can be used to modify the behavior of the curve, altering its higher-order derivatives.

Control Points for Bipedal robots Independent of the number of joints of the robot and its structure, it would be required to control at least the position of its CoM, the position of its balancing foot, and the rotation of its balancing foot. Given the hybrid nature of the bipedal gait, the phase transition instants arise as a natural choice for the control points in the robot's reference definition. For determining the robot references, we consider four (possibly iterative) stages:

1. Determine footprints planning: For determining the step locations, it is required to be aware of the geometric limitations of the robot (maximum step length and width) and the areas where the feet can be located. One common scenario corresponds to flat terrain with no obstacles where a fixed step geometry can be followed. In this case, it would be required to determine four parameters, the step length, the step width, the single support duration, and the double support duration. With these parameters, the position of the footprints is set as presented in Figure **2-7**.

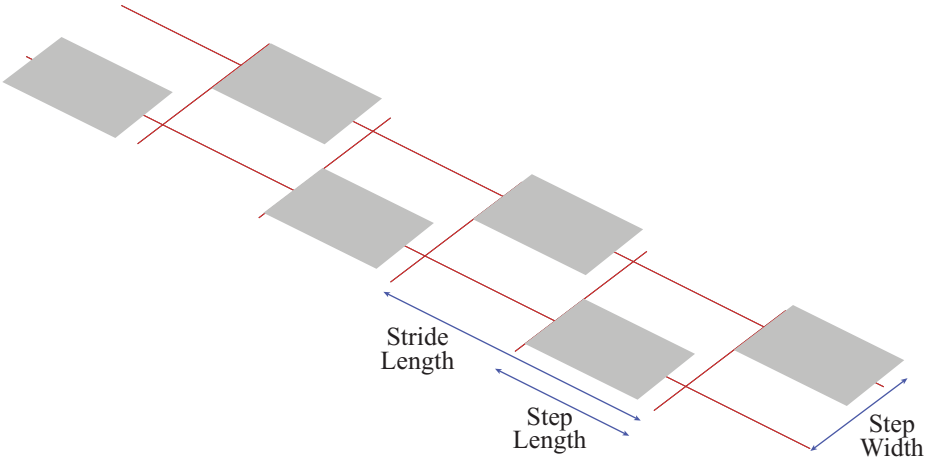


Figure 2-7.: Footprints planning

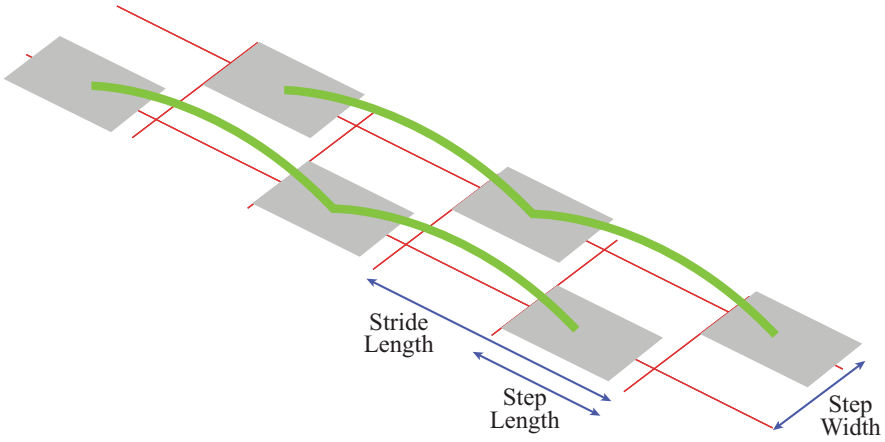


Figure 2-8.: Feet Trajectories planning

2. Determine the trajectory of the feet: Setting initial conditions with the known initial and final positions with velocity and acceleration set to 0 allows for the calculus of the three initial and final coefficients of the Bézier polynomial for the feet position at each phase, and it is required at least an additional coefficient to guarantee that the foot rises over the ground (without it, the trajectory results flat which is infeasible).

3. Determine the trajectory of the CoM: Setting the initial conditions to be the final conditions of the previous phase and the final to be desired at the end of the current, the whole trajectory can be entirely determined without additional coefficients (only the six required for the boundary conditions). Note that contrary to the feet, except for the initial and final steps, the velocity and acceleration of the CoM are free. An example trajectory is presented in Figure **2-9**.

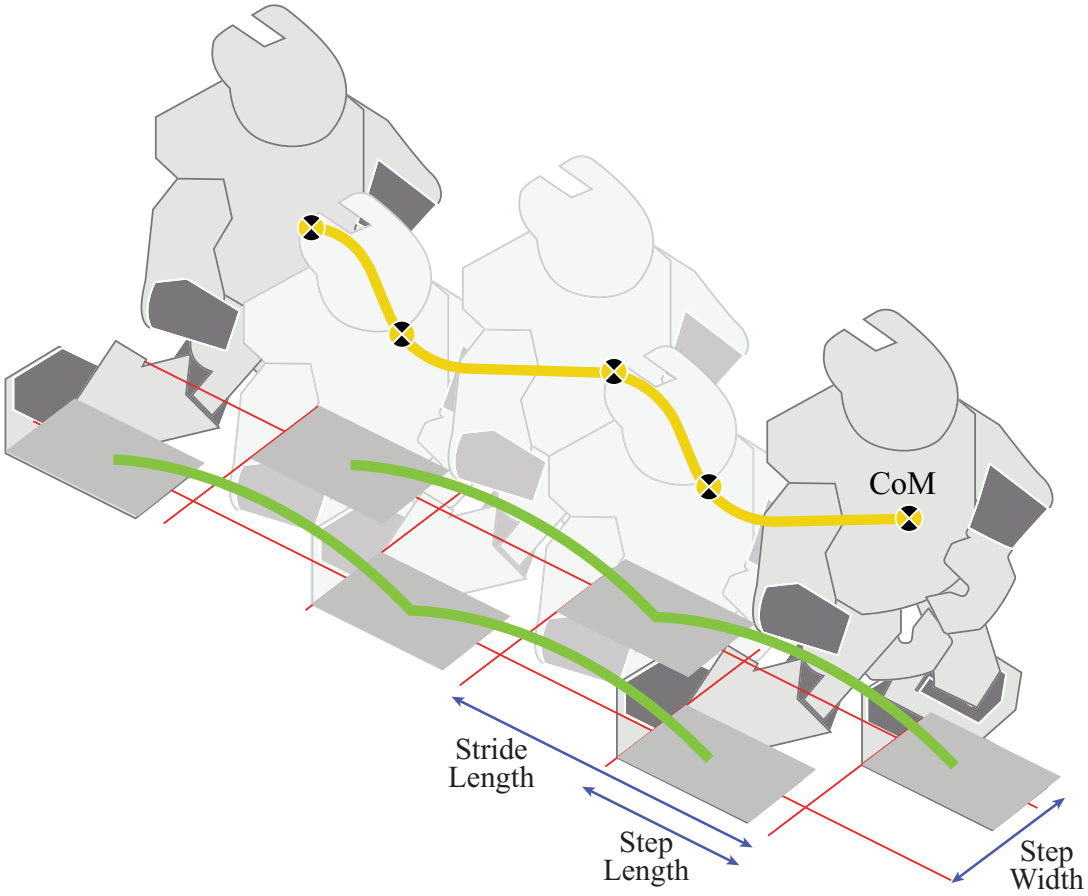


Figure 2-9.: Bézier trajectory planning

4. Check the stability criteria of the resulting trajectory: Once the trajectory is determined, the chosen stability criterion must be verified. If the criterion is fulfilled, the process is complete, but if it is not, the trajectory must be updated so that it does. For instance, if the criterion corresponds to the ZMP criterion, the condition of $x_{ZMP} \in SP$ must be fulfilled at each instant. If not, the coefficients and/or time parameters must be adjusted accordingly. Figure **2-10** shows the ZMP trajectory for a certain CoM Bézier Trajectory.

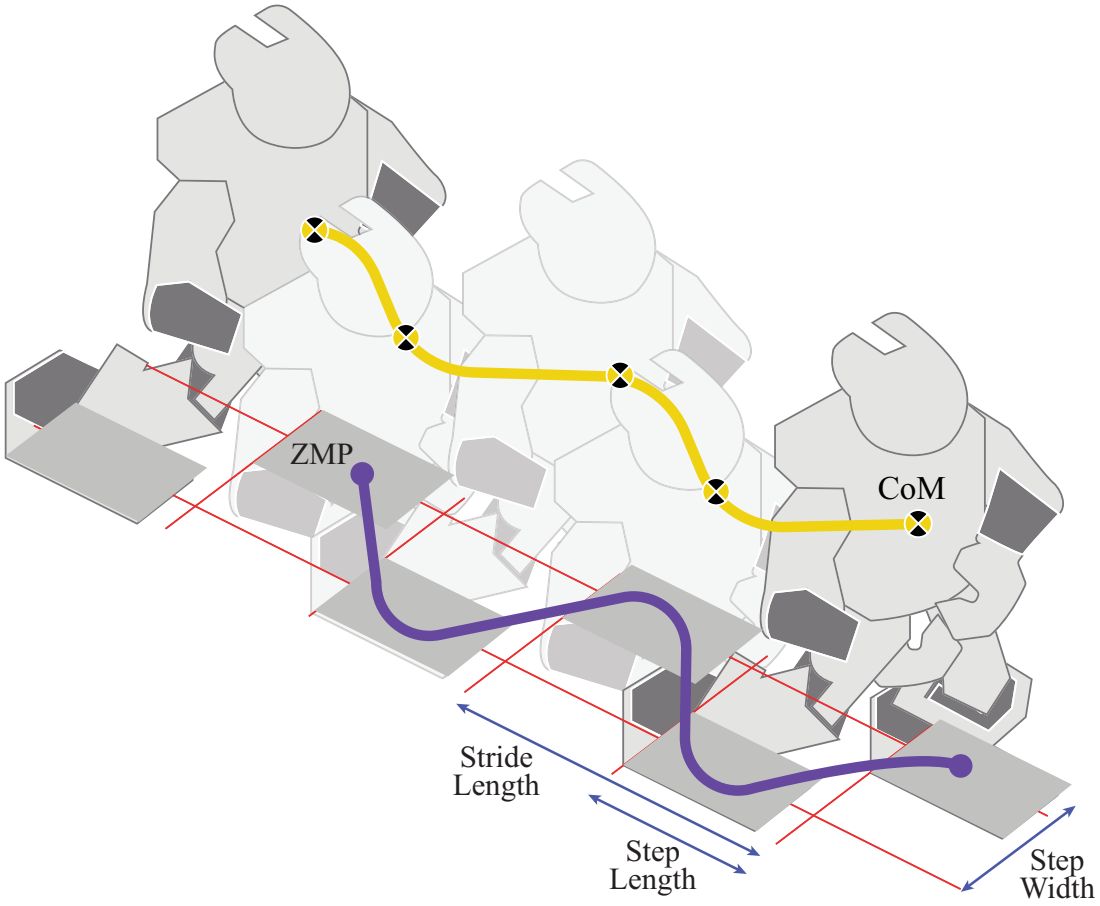


Figure 2-10.: Bézier trajectory stability checking

ZMP-Based Trajectories

Since the earliest works on bipedal robot gait, one of the most popular frameworks is related to the consideration of the ZMP criterion as presented in [Vukobratovic and Borovac, 2004] (see Section 2.1.6). In these frameworks, the references for the robot are set in three stages:

1. Determine footprints planning: This step is the same as for the Bézier case (see Figure **2-7**).solving the differential equation corresponding to x_{CoM} is possible

2. Determine a trajectory for the ZMP of the robot consistent with the steps planning: As the ZMP criterion requires the ZMP to be always within the support polygon, the footprints plan determines its feasible region at each instant. This information makes tracing a continuous trajectory over that feasible region possible. It is common to use linear segments that go from the center of one foot to the next, as presented in Figure **2-11** although more complex trajectories may also be considered.

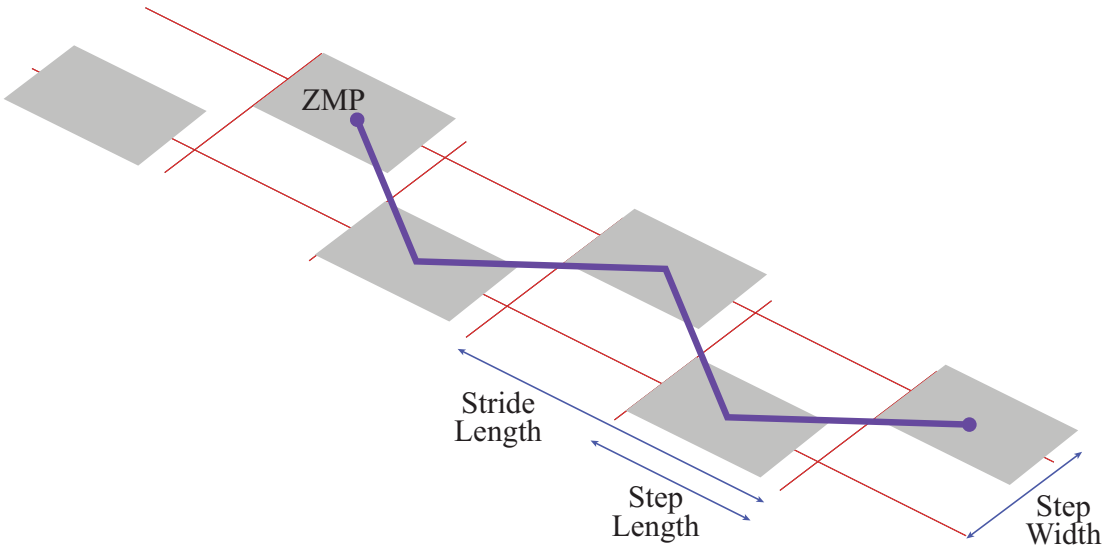


Figure 2-11.: ZMP Planned Trajectory

3. Evaluate the CoM and feet trajectories that are consistent with the steps planning and the ZMP trajectory: Once the footprints and the ZMP trajectories have been determined, the last step is to determine what are the corresponding trajectories for the CoM of the robot and its feet. Whether we are using the complete ZMP equation or the simplified one, solving the differential equation corresponding to x_{CoM} is possible based on the dynamics of x_{ZMP} determining the CoM trajectory. On the other hand, the trajectory of the feet can be determined with Bézier polynomials or with any other soft parametrization as long as the footprints planning is followed. Figure **2-12** presents two example resulting trajectories when the resulting motion is slow and fast. Note that as the velocity goes to 0, the CoM trajectory becomes very similar to the ZMP trajectory, and as the velocity grows, the trajectory gets smoother and closer to a linear path.

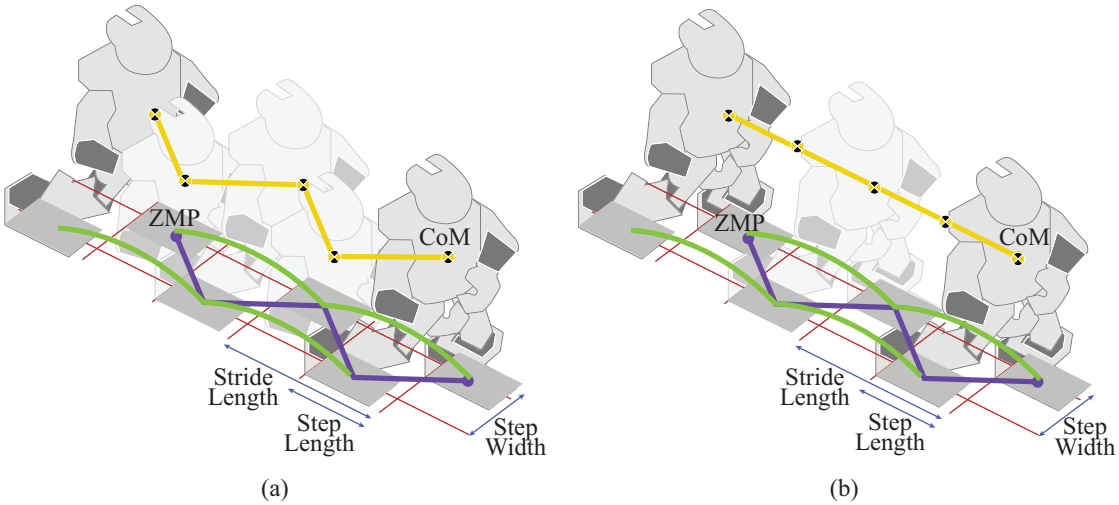


Figure 2-12.: CoM and feet trajectories obtained from the ZMP trajectory - (a) slow resulting trajectory (b) fast resulting trajectory

Trajectory planning methods using Bézier polynomials and ZMP-based trajectories offer distinct approaches for generating desired robot trajectories in humanoid robot locomotion. Both methods ensure stability, although they differ in their approach. Bézier polynomials provide a flexible and intuitive way to parametrize trajectories, allowing for smooth and customizable motion profiles. While stability is checked a posteriori in the Bézier approach, it offers more flexibility and the potential to achieve more efficient performances. On the other hand, ZMP-based trajectories prioritize stability from the start by fixing the ZMP trajectory and solving the differential equation to obtain the Center of Mass (CoM) and foot trajectories. This approach guarantees stability throughout the robot's motion. The choice between these trajectory planning techniques ultimately depends on the specific requirements, motion characteristics, and desired trade-offs between flexibility and guaranteed stability in the humanoid robot locomotion task.

2.3. Darwin Mini

The Darwin Mini humanoid robot is a product of the company Robotis which counts with multiple servomotors, sensors, an internal computer, and structural pieces that correspond to a 16 DoF humanoid structure with the following joints:

- 2 Frontal plane ankle joints
- 2 Sagittal plane angle joints
- 2 Sagittal plane Knee joints
- 2 Sagittal plane hip joints

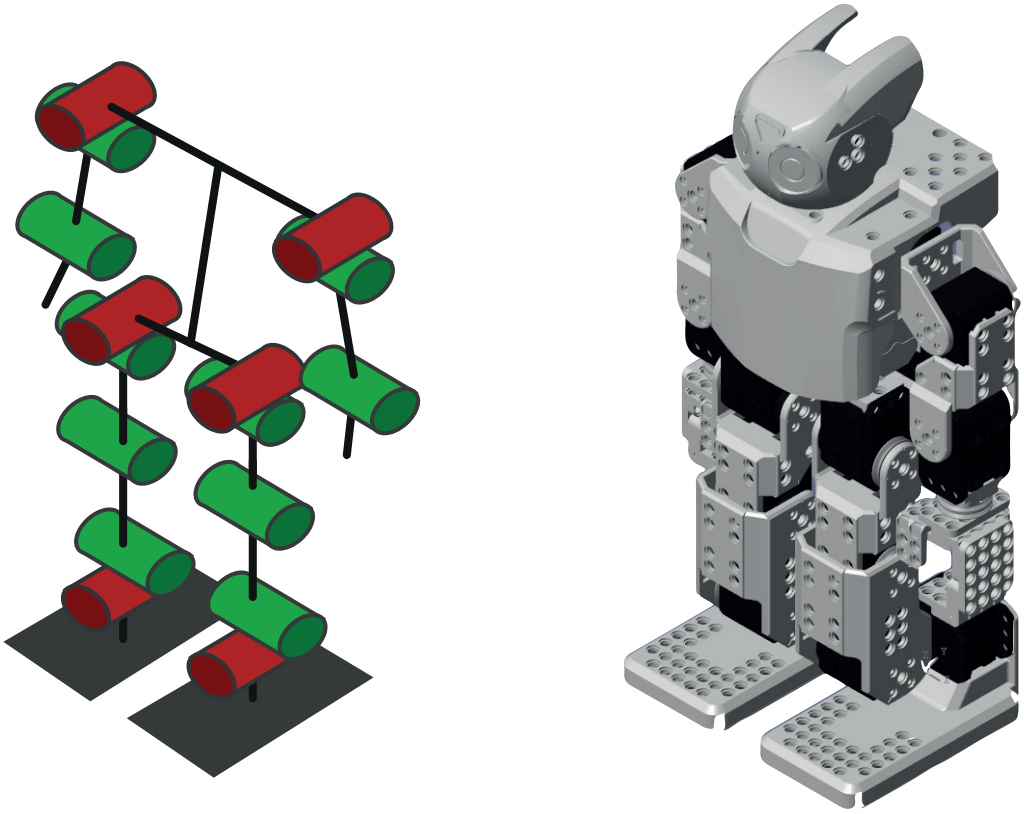


Figure 2-13.: Humanoid Robot Robotis Darwin Mini

- 2 Frontal plane hip joints
- 2 Sagittal plane shoulder joints
- 2 Frontal plane shoulder joints
- 2 Frontal plane elbow joints

2.3.1. Kinematic Model

To describe the robot's pose according to Section 2.1.3, we set reference frames at each of the 16 joints and the end of the limbs as presented in Figure **2-14**.

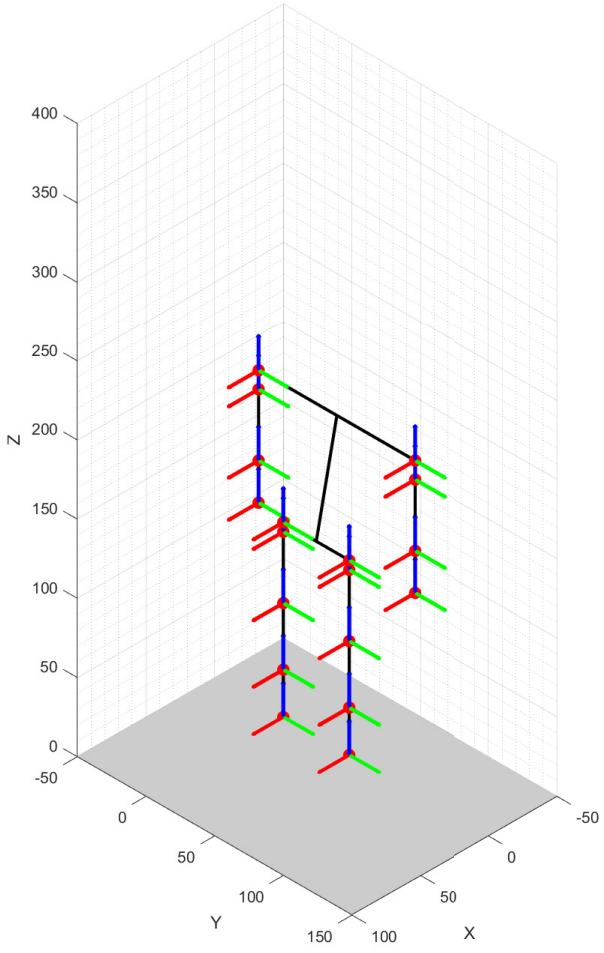


Figure 2-14.: Neutral position frames of Darwin mini

The homogeneous transformation matrices that describe these reference frames are evaluated sequentially from the support foot to each of the remaining limbs' ends, assuming that the support foot is static ($z_{sf} = \dot{z_{sf}} = \ddot{z_{sf}} = \theta_{sf} = \omega_{sf} = \alpha_{sf} = 0$) and does not rotate (which is valid as long as there is no slipping and the ZMP criteria are fulfilled). There are three kinematic chains to consider as follows:

- Support foot, frontal support ankle, sagittal support ankle, support knee, sagittal support hip, frontal support hip, frontal balancing hip, sagittal balancing hip, balancing knee, sagittal balancing ankle, frontal balancing ankle, and balancing foot.
- Support foot, frontal support ankle, sagittal support ankle, support knee, sagittal support hip, frontal support hip, frontal balancing hip, support sagittal shoulder, support

frontal shoulder, support elbow, and support hand.

- Support foot, frontal support ankle, sagittal support ankle, support knee, sagittal support hip, frontal support hip, frontal balancing hip, balancing sagittal shoulder, balancing frontal shoulder, balancing elbow, and balancing hand.

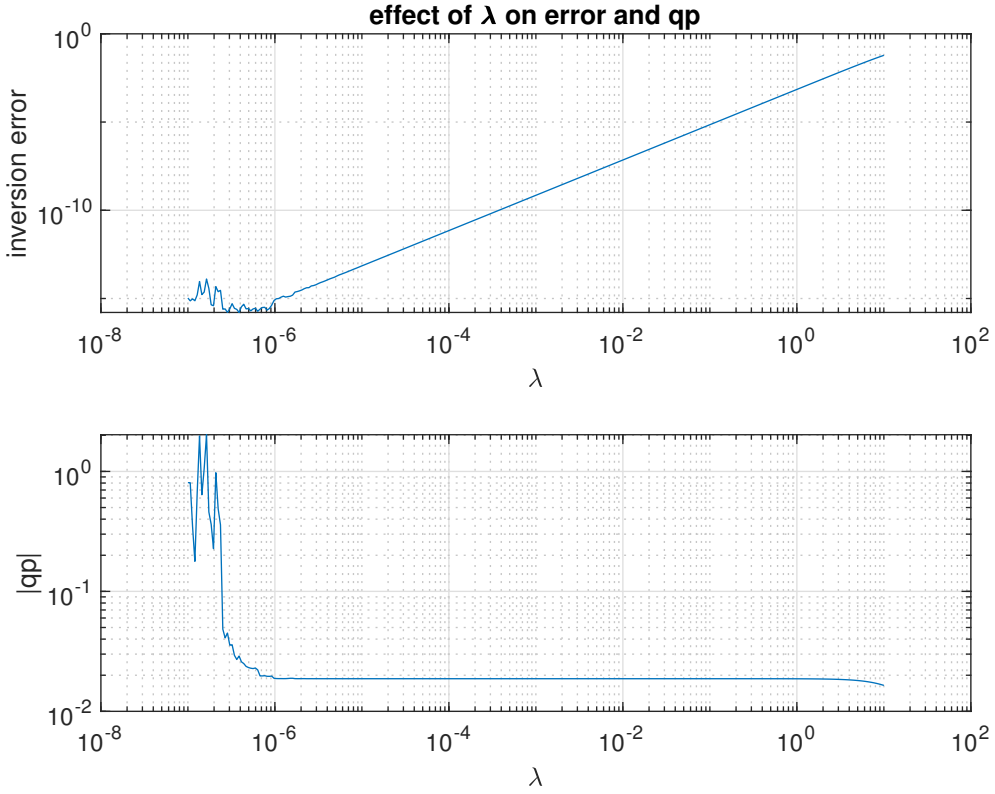


Figure 2-15.: Variation of the tracking error and solution magnitude when varying the damping coefficient λ for the Darwin Mini Robot

As mentioned in Section 2.1.3, the inversion of the robot's Jacobian matrix can be performed with a damping factor λ ($\lambda = 0$ corresponds to the Moore Penrose Pseudo-inverse method and $\lambda > 0$ corresponds to the damped pseudo-inverse method). To determine the appropriate value for this constant, we evaluated the behavior of the inversion error and the norm of the output $\dot{q}$ in different positions for a unitary input $|\dot{x_d}| = 1$ as presented in Figure **2-15**. As expected, the inversion error is very low for small λ values. However, as $\lambda \to 10^{-6}$, the magnitude of the output grows significantly and exhibits numerical instability. On the other hand, for greater values of λ, the inversion error grows exponentially (linearly in the logarithmic plot), imposing an upper bound on the damping coefficient. For instance, for a determined inversion error magnitude to be lower than 10^{-5} observing onto the graphic, it can be determined that λ must be lower than 0.1, i.e., $|\dot{x_d} - J^{+}\dot{x_d}| <= 10^{-5} \to \lambda <= 10^{-1}$.

Under this criteria, any damping factor that satisfies $10^{-6} < \lambda \leq 10^{-1}$ is considered feasible, and we employed a value of 10^{-4} for most of our experiments.

2.3.2. Kinetic Parameters

To accurately parametrize the kinetic model of the robot according to Section 2.1.4, we considered the CAD models of each of the main components of the robot, the density of the material of the links, and the specific weight of the available parts to build a detailed model in Matlab's Simscape Multibody that allows estimating their inertial properties using the Inertia Sensor block. Appendix B presents the inertia matrices of each of Darwin's links relative to their corresponding CoMs.

2.3.3. Actuation

According to Darwin's Manual [ROBOTIS, 2022c], it has 16 Dynamixel servomotors XL-320. Each of them has a stall torque of 0.39 N.m (at the recommended operation voltage of 7.4V), has its internal micro-controller and can be operated in 2 different modes:

- Wheel Mode: The servomotor tracks the MOVING SPEED registry.
- Position Control Mode: The servomotor tracks the GOAL POSITION registry from 0 to 300 degrees.

The internal controller of the XL-320 servomotors presented in Figure **2-16** corresponds to a standard PID controller whose constants can be tuned by the user modifying the correspondent registers onto the servomotor's microcontroller.

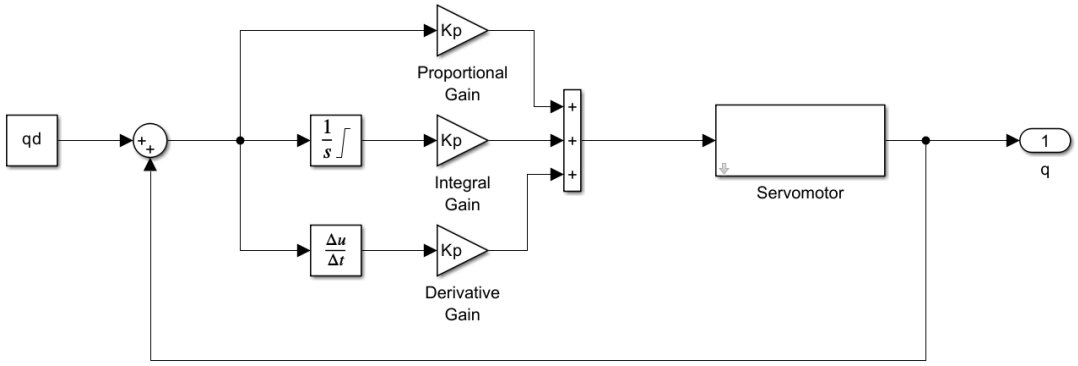


Figure 2-16.: Darwin Servomotors' PID Control Diagram (adapted from [ROBOTIS, 2022c])

2.3.4. Sensing

Darwin only has proprioceptive sensors. Each one of the servomotors present in Darwin [ROBOTIS, 2022d] has an internal 10-bit encoder (resolution about 0,29 deg for position and about 0.11 rpm for velocity) to determine its current state according to the control mode. Additionally, one of the registries of the servomotor has an estimate of the load over the motor with a resolution of 11 bits (resolution of about 0.1 %). Although the manual states that this sensor may be inaccurate for estimating torque or weight.

2.3.5. Internal PC

MAXE counts with an internal OpenCM 9.04 board that has the following specifications:

Processor	STM32F103CB 32 bits (ARM Cortex-M3 @72MHz)
Memory	148kbytes (128 Flash and 20 SRAM)
Wireless Connectivity	Bluetooth 4.2, BLE, onboard antenna
Input power	5V DC

2.4. Robotis Engineer Kit (MAX-E2)

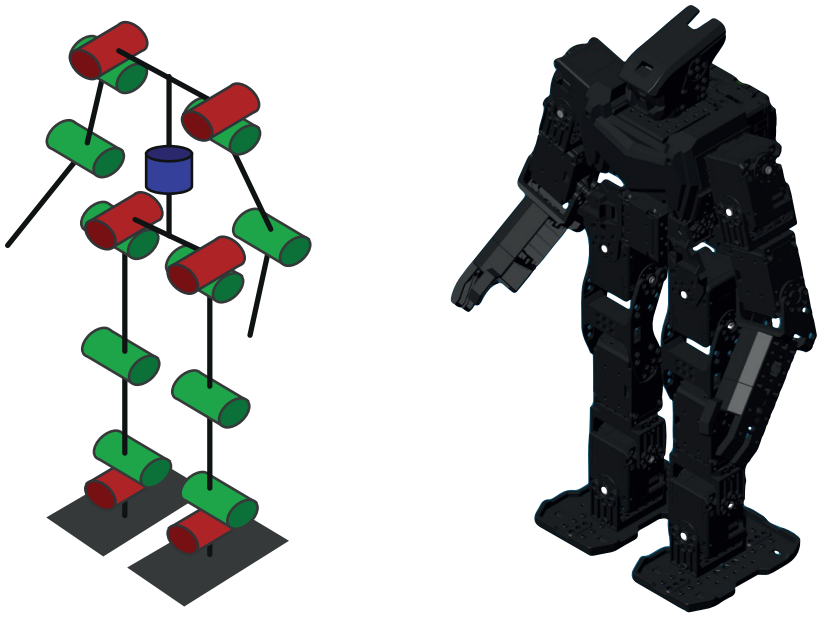


Figure 2-17.: Humanoid Assembly of the Robotis Engineer Kit

The Robotis Engineer Kit counts with multiple servomotors, sensors, an internal microcontroller, and structural pieces that can be assembled in many shapes. Particularly, it can be assembled as MAX-E2, a humanoid robot with 16 links and 17 joints:

- 2 Frontal plane ankle joints
- 2 Sagittal plane angle joints
- 2 Sagittal plane Knee joints
- 2 Sagittal plane hip joints
- 2 Frontal plane hip joints
- 1 Coronal plane waist joint
- 2 Sagittal plane shoulder joints
- 2 Frontal plane shoulder joints
- 2 Sagittal plane elbow joints

2.4.1. Kinematic Model

To describe the robot's pose according to Section 2.1.3, we set a reference frame at each of the 17 joints and limbs as presented in Figure **2-18**.

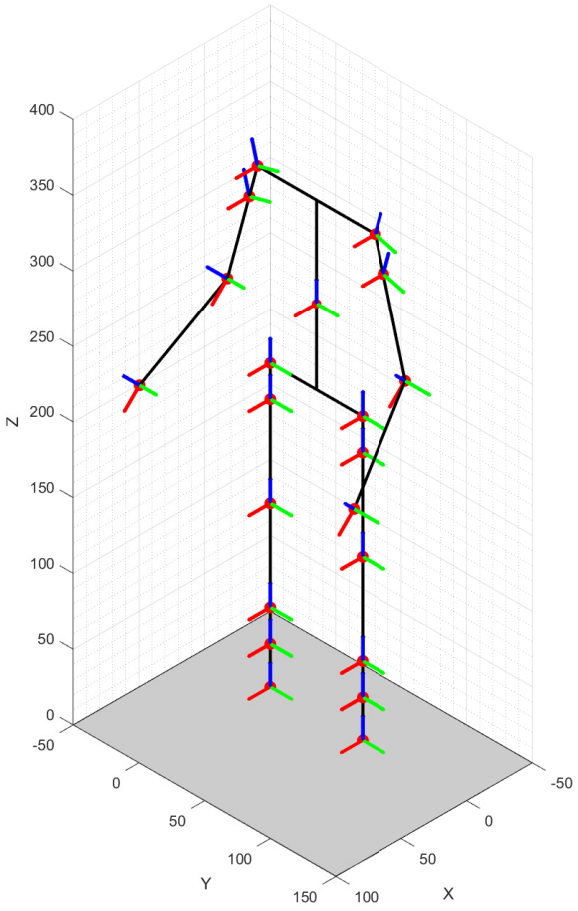


Figure 2-18.: Neutral position frames of MAXE

The homogeneous transformation matrices that describe these reference frames are evaluated sequentially from the support foot to each of the remaining limbs' ends, assuming that the support foot is static ($z_{sf} = \dot{z_{sf}} = \ddot{z_{sf}} = \theta_{sf} = \omega_{sf} = \alpha_{sf} = 0$) and does not rotate (which is valid as long as there is no slipping and the ZMP criteria are fulfilled). There are three kinematic chains to consider as follows:

- Support foot, frontal support ankle, sagittal support ankle, support knee, sagittal support hip, frontal support hip, frontal balancing hip, sagittal balancing hip, balancing knee, sagittal balancing ankle, frontal balancing ankle, and balancing foot.
- Support foot, frontal support ankle, sagittal support ankle, support knee, sagittal support hip, frontal support hip, frontal balancing hip, waist, support sagittal shoulder, support frontal shoulder, support elbow, and support hand.

- Support foot, frontal support ankle, sagittal support ankle, support knee, sagittal support hip, frontal support hip, frontal balancing hip, waist, balancing sagittal shoulder, balancing frontal shoulder, balancing elbow, and balancing hand.

Note that the feet switch roles depending on the walking phase, assuming a gait model consisting of single and double support (instantaneous or not) interleaved phases. Also, it is worth mentioning that models of jogging or running (that won't be considered in this work) may include flying phases where none of the feet is touching the ground, with the center of mass of the robot presenting a ballistic motion. Regarding the damping coefficient λ, the criteria considered are the same as the employed for Darwin, and based on our numerical experiments, the bounds are very similar, as can be expected when observing the behavior of the inversion error and the solution norm in Figure **2-19**.

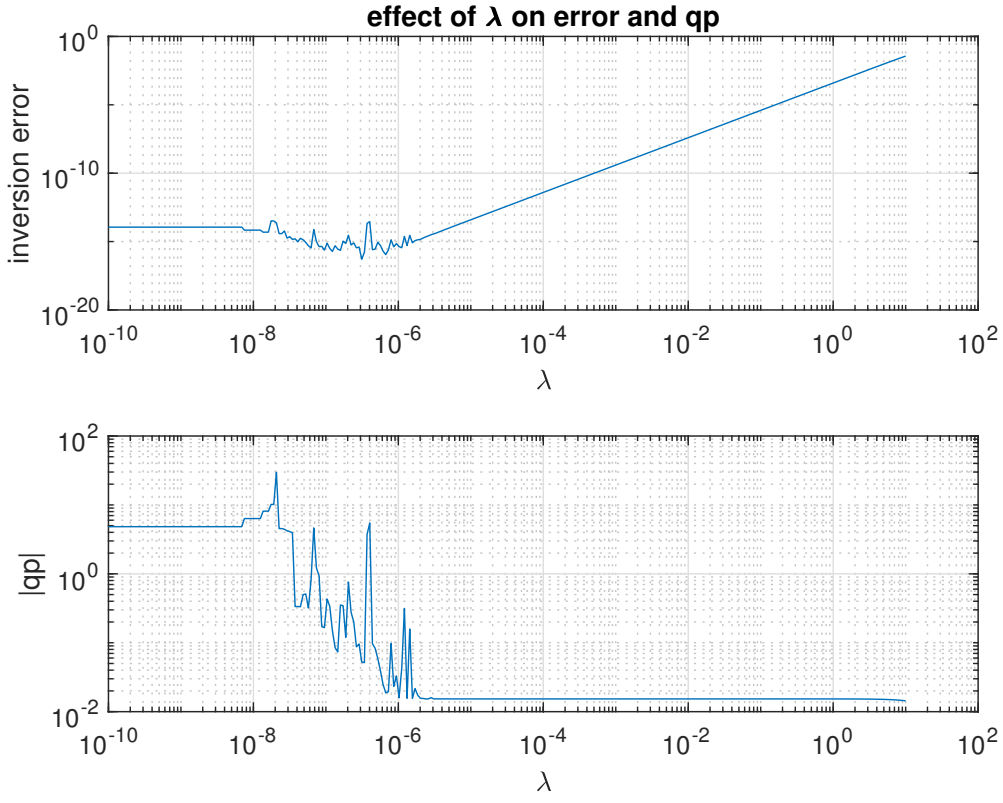


Figure 2-19.: Variation of the tracking error and solution magnitude when varying the damping coefficient λ for the Robot MAX-E2

2.4.2. Kinetic Model

Following the same process employed for Darwin's inertial properties according to Section 2.1.4, we obtained the corresponding ones for MAX-E2. Appendix C presents the kinetic

parameters of each one of the MAX-E2's links.

2.4.3. Actuation

According to MAXE's Manual [ROBOTIS, 2022a][ROBOTIS, 2022b], it counts with two versions of the same Dynamixel servomotors. 6 dual servomotors 2XL430-W250T and five single servomotors XL430-W250T. Each of them has a stall torque of 1.4 N.m (at the recommended operation voltage of 11.1V), has its internal micro-controller and can be operated in 4 different modes:

- Velocity Control Mode: The servomotor tracks the GOAL VELOCITY registry. This mode uses a standard PI controller with tunable gains.
- Position Control Mode: The servomotor tracks the GOAL POSITION registry from 0 to 360 degrees. This mode employs a PID controller with feedforward considering the references set by a trapezoidal profiler as presented in Figure **2-20**.
- Extended Position Control Mode (Multi-turn): The servomotor tracks the GOAL POSITION registry from -256 to 256 revolutions.
- PWM Control Mode (Voltage Control Mode): The registry GOAL PWM directly determines the input voltage applied to the servomotor without internal control.

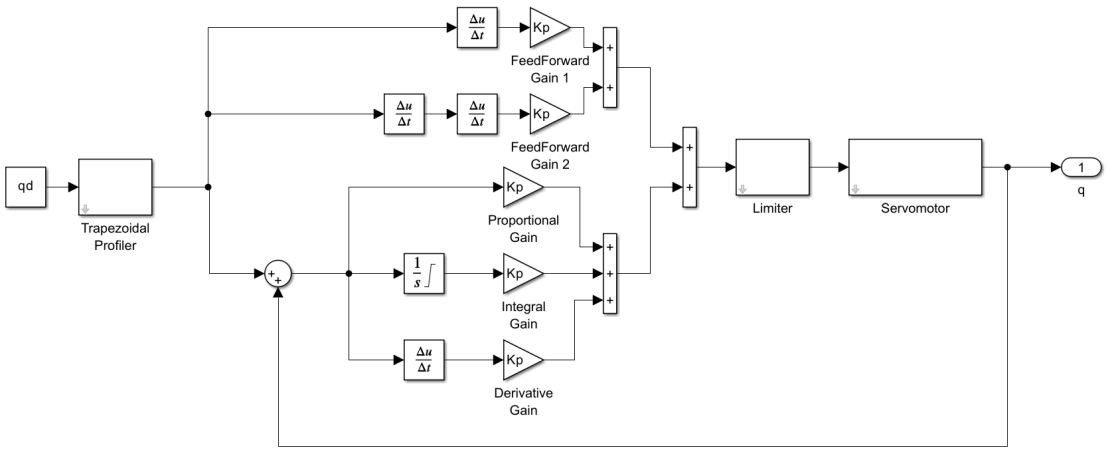


Figure 2-20.: MAX-E2 Servomotors' PID + Feedforward Control Diagram (adapted from [ROBOTIS, 2022a])

There is a small servomotor on the neck of MAXE, but its specifications are not available in the product documentation.

2.4.4. Sensing

Each servomotor in MAXE has an internal 12-bit encoder to determine its current position. Depending on the control mode, the internal controller can have a registry of up to 4 bytes to preserve the position information. Additionally, we equipped our MAXE with an Inertial Measurement Unit (IMU) MPU9255 that registers:

- Linear accelerations between $\pm 2, \pm 4, \pm 8, and \pm 16g$ on each axis with a resolution of 16 bits.
- Angular velocity in between $\pm 250, \pm 500, \pm 1000, \pm 2000 \, \mathrm{deg}/s$ on each axis with a resolution of 14 or 16 bits.
- Angular position (based on a 3-axis magnetometer) with a resolution of 14 or 16 bits depending on the configuration.
- Barometric pressure between 300 and 1100 hPa with a resolution of about 0.0016 hPa.
- Room temperature with a resolution of about 0.01 Celsius degrees.

And with 8 Interlink Electronics 0.2" circular force sensing resistors disposed of in the soles of MAXE's feet to estimate the contact forces and their distribution to determine contact and to estimate the ZMP position onto the support Polygon. Each can register forces between 0.2 and 20 N with analog variations. This project won't consider the barometric and temperature information for control purposes.

2.4.5. Internal PC

MAXE counts with an internal Raspberry Pi Zero W board that has the following specifications:

Processor	Broadcom BCM2710A1, quad-core 64-bit SoC (Arm Cortex.A53 @1Ghz)
Memory	512MB LPDDR2
Wireless Connectivity	2.4GHz IEEE 802.1b/g/n wireless LAN, Bluetooth 4.2, BLE, onboard antenna
Input power	5V DC 2.5A
Operation Temperature	-20 C to +70 C

3. Imitation Learning

In recent years, there have been significant advancements in robotics, particularly in developing autonomous systems capable of performing complex tasks. Within this domain, imitation learning (IL) has emerged as a critical research area, enabling robots to learn from human demonstrations and imitate their behavior. In applying artificial learning to humanoid gait, imitation learning plays a crucial role in providing an effective initialization for the learning process, thereby enhancing sample efficiency and optimizing overall system performance.

While humanoid robots are designed to resemble humans in form and structure, inherent differences exist in their physical properties and dynamics. These disparities pose a challenge when directly applying human data to humanoid robot locomotion. Despite this, human demonstrations are a natural source of inspiration for humanoid locomotion. However, the mapping between human behavior and robot actions is not straightforward. It requires careful consideration of the divergences between humans and robots, necessitating the adaptation of learned behavior to account for the unique characteristics and capabilities of the robot platform. This calls for developing sophisticated techniques that bridge the gap between human demonstrations and the execution of fluid and stable locomotion on humanoid robots.

This chapter aims to delve into the realm of imitation learning and its application in the context of humanoid gait. It focuses on harnessing the power of imitation learning to distill efficient policies for humanoid robots. By leveraging the expertise and experience in human demonstrations, imitation learning is a powerful tool to guide the learning process and accelerate convergence toward optimal solutions. By imitating the behavior of an expert, a humanoid robot can acquire locomotion patterns and benefit from the acquired knowledge, leading to a more efficient learning process.

The primary purpose of incorporating imitation learning into the thesis framework is to provide an effective initialization for the reinforcement learning algorithm. The exploration-exploitation trade-off in reinforcement learning is improved by starting from a policy that closely resembles the expert's behavior. This initialization is expected to significantly enhance the sample efficiency of the overall optimization process, enabling the humanoid robot to converge to optimal policies faster and with fewer samples.

This chapter will explore various imitation learning techniques, including direct imitation, geometrical, optimization-based, and Generative Adversarial Networks (GANs) based approaches. Each technique has advantages and considerations, which will be thoroughly examined within the humanoid gait. Analyzing and comparing these techniques aims to identify

the most suitable approach for initializing the reinforcement learning process and achieving efficient and effective locomotion in humanoid robots.

3.1. What is Imitation Learning?

Let us consider a task space for the humanoid robot, denoted as $x_{robot} \in X_{robot}$, which represents the states of the robot during the locomotion task. Similarly, the states observed in the human expert are denoted as $x_{human} \in X_{human}$. Imitation learning aims to establish a mapping or transformation between the human expert's and the robot's states, enabling the robot to imitate the desired behavior.

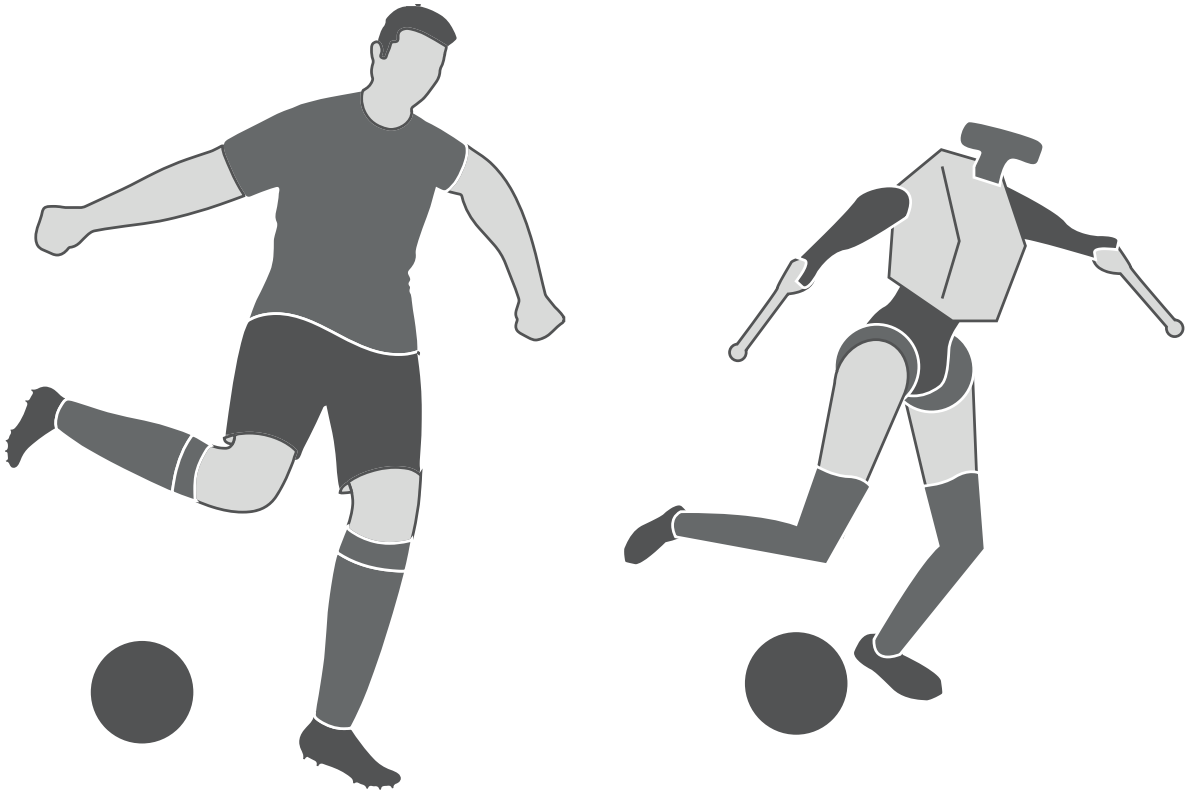


Figure 3-1.: Example of Humanoid Robot imitating a human pose - Adapted from ARTEMIS Robot

Formally, imitation learning seeks to learn a policy $\pi(\theta) : \mathcal{X}robot \rightarrow \mathcal{U}robot$ that maps the robot's states to its corresponding actions in the robot's action space $\mathcal{U}robot$. The policy parameters θ are learned by leveraging a set of expert demonstrations, where each demonstration consists of a sequence of state-action pairs (x_{human}, u_{human}) or, more often, only the state information x_{human}. The states observed in the human demonstrations provide valuable information about desired behavior and can guide the robot in achieving similar performance.

The goal of imitation learning can be formulated in different ways, depending on the specific objectives of the task. One approach is to minimize the difference between the robot's states (x_{robot}) and the human expert's states (x_{human}) in a common feature space:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=0}^{N_f} D(\phi(x_{robot}(k)), \phi(x_{human}(k))) \\ \text{subject to} \quad & x_{robot} \text{ follows } \Sigma \\ & u = \pi_\theta(x_{robot}) \end{aligned}, \tag{3-1}$$

where $\phi(\cdot)$ corresponds to a function that maps both the human's space and the robot's space to a common domain, $D(\cdot, \cdot)$ corresponds to a distance function (as the Euclidean norm) between the arguments, N_f corresponds to the number of frames present in the interest demonstration, Σ corresponds to the model of the robot presented in Chapter 2 (Section 2.1), and $\pi_\theta(\cdot)$ corresponds to a tunable parametric policy.
This approach aims to align the robot's behavior with the expert's behavior, enabling the robot to imitate the desired locomotion patterns exhibited by the expert.
Alternatively, imitation learning can focus on maximizing an external performance measure or target, such as movement speed with low energy consumption. In this case, the objective is to learn a policy that maximizes a reward function that captures the desired performance criteria. The reward function can be modified based on the discrepancy between the robot's states (x_{robot}) and the human expert's states (x_{human}), as well as other task-specific objectives:

$$\begin{aligned} \text{maximize} \quad & \sum_{k=0}^{N_f} R(x_{robot}(k)) - D(\phi(x_{robot}(k)), \phi(x_{human}(k))) \\ \text{subject to} \quad & x_{robot} \text{ follows } \Sigma \\ & u = \pi_\theta(x_{robot}), \end{aligned} \tag{3-2}$$

where $R(\cdot)$ corresponds to the reward function to be maximized.
This combination allows for a trade-off between mimicking the expert's behavior and optimizing the robot's performance.

Key Components of Imitation Learning

Imitation learning encompasses several key components that contribute to its successful implementation:

- Expert Demonstrations: The availability of high-quality expert demonstrations is crucial for imitation learning. These demonstrations provide the robot with examples of the desired behavior and serve as a reference for learning the task.

- Feature Mapping: To establish a meaningful correspondence between the robot's and the human expert's states, a feature mapping function
  phi is often employed. This function transforms the states into a common feature space, facilitating the comparison and alignment of behaviors.

- Policy Learning: The core of imitation learning lies in learning a policy π that maps the robot's states to its actions. Various techniques, such as behavioral cloning, inverse reinforcement learning,

3.2. Expert Demonstrations: The Motion Capture Database

To obtain expert demonstrations for the imitation learning process, the repository of Motion Capture data from the CMU Graphics Lab is utilized [Lab, 2003]. This database provides a collection of free motions that can be downloaded and used for research. Among many others, there are several instances of walking gaits. These expert demonstrations serve as a valuable resource for imitation learning, provide real-world examples of desired behavior, and serve as a reference for the robot's learning process. By leveraging the captured motion data, the robot can learn to imitate the movements and behaviors exhibited by human experts.

3.2.1. Motion Capture Process

The motion capture process at CMU involved using 12 Vicon infrared MX-40 cameras placed around a rectangular area in the motion capture lab. These cameras record the motion at a 120HZ and capture images of the subject wearing a black jumpsuit with markers taped on. The markers, seen by the cameras in infrared, are triangulated to obtain 3D data of the subject's movements.

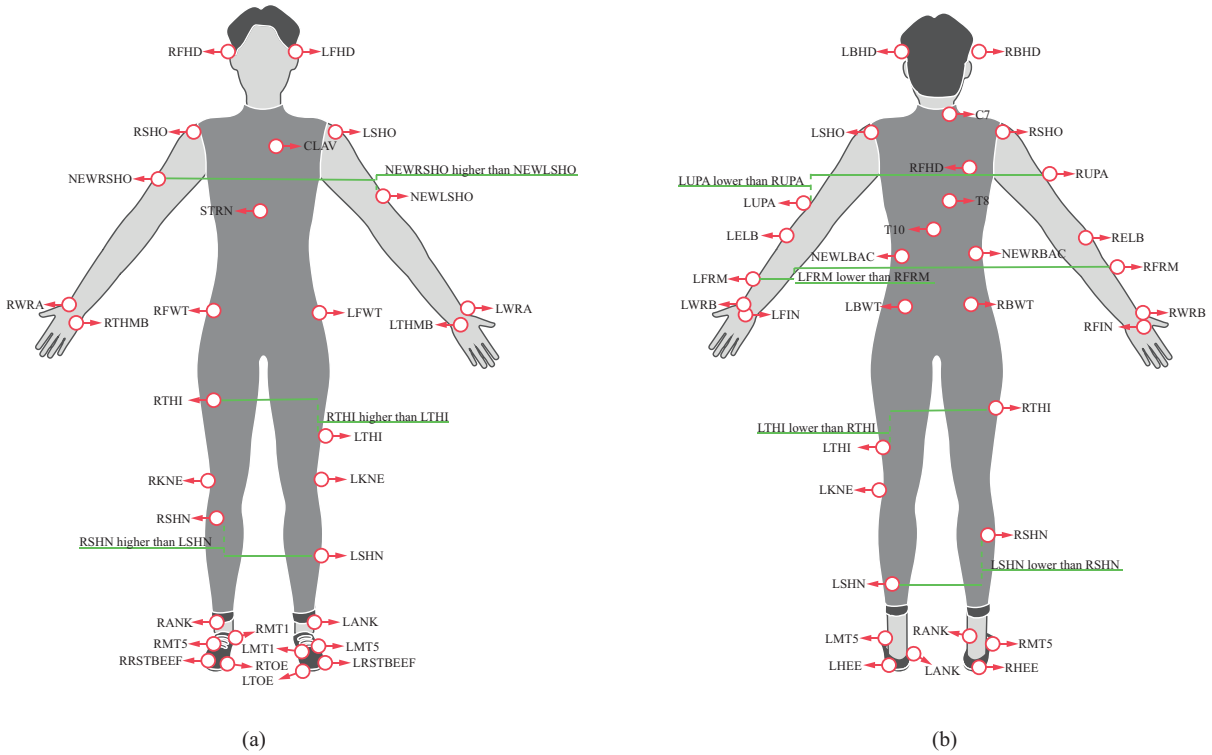


Figure 3-2.: Diagram of the markers' positions for motion capture - adapted from [Lab, 2003]

3.2.2. Data Types and Formats

The CMU motion capture database offers two types of data: marker positions and skeleton movement.

- **Marker Positions:** The database provides files containing 3D marker positions in the .c3d format. Each file represents a sequence of recorded marker positions over time, enabling the reconstruction of the subject's movements.

- **Skeleton Movement:** The database also offers files in the form of a pair, consisting of a .vsk/.v file or a .asf/.amc file. The .vsk/.v pair describes the skeleton and its joints, including their connections, lengths, degrees of freedom, and transformations. The .asf/.amc pair contains the actual movement data. The .v file format provides information about bone rotations and translations in six values for each bone. The .asf/.amc format, on the other hand, represents bone rotations in Euler angles.

We will focus on the Skeleton Movement format since the .asf/.amc format provides information about the skeleton's movements by describing its joints' connections, degrees of freedom, and transformations. This representation closely aligns with the structure and movements

of the humanoid robot, allowing for a more direct mapping between the demonstrated human movements and the robot's own task space. By leveraging the .asf/.amc format, we can establish a clear correspondence between the joint movements observed in human demonstrations (represented by the .amc files) and the joint movements of the humanoid robot. This correspondence is essential for developing a practical imitation learning framework, as it facilitates knowledge transfer from human experts to the robot.
Note: all the demonstrations that we considered present models with 62 degrees of freedom (DoF) and corresponded to floating base models (all of the kinematic chains start from an arbitrary point next to the waist of the subject) where the six first parameters correspond to the floating base parameters and the remaining 56 to the subjects' joints.

Data Preprocessing

Data smoothing: In our case, where the motion capture data contains noticeable noise that can significantly impact the analysis of velocity and acceleration behaviors, applying the Savitzky-Golay filter can help to mitigate the noise and provide a cleaner representation of the underlying motion. The Savitzky-Golay filter is a smoothing technique that applies a weighted least-squares polynomial regression to the data. Unlike other filters that typically average or interpolate the data points, the Savitzky-Golay filter performs a local polynomial regression on a sliding window of data points. The filter works by fitting a polynomial to a subset of neighboring data points within the window and using this polynomial to estimate the smoothed value for the central data point. This process is repeated for each data point in the time series, resulting in a smooth trajectory. The Savitzky-Golay filter is particularly well-suited for our task as it effectively reduces noise while preserving important features and characteristics of the original data. It offers advantages over simple moving average or median filters by better preserving sharp transitions and capturing subtle variations in the motion.
Units scaling: As mentioned in the database documentation, the units on the .amc files are converted to meters using the conversion factor $(1{,}0/0{,}45) * 2{,}54/100{,}0$.
Forward direction correction: To ensure consistency and alignment between the human motion data and the humanoid robot's task space, we rotated the data to establish a consistent reference frame. The rotation aligned the advanced direction of the captured human motion with the X+ axis, ensuring a standardized orientation across all demonstrations. To do so, we considered the data's forward direction as the direction of the mean velocity of the root frame in the interest demonstration:

$$\hat{x}_{MoCap} = \frac{\overrightarrow{v^{Root}_{MoCap}}}{|\overrightarrow{v^{Root}_{MoCap}}|}, \tag{3-3}$$

where $\hat{x}_{MoCap}$ corresponds to the estimated forward direction of the data and $\overrightarrow{v^{Root}_{MoCap}}$ corresponds to the velocity of the root frame in the motion capture interest demonstration.

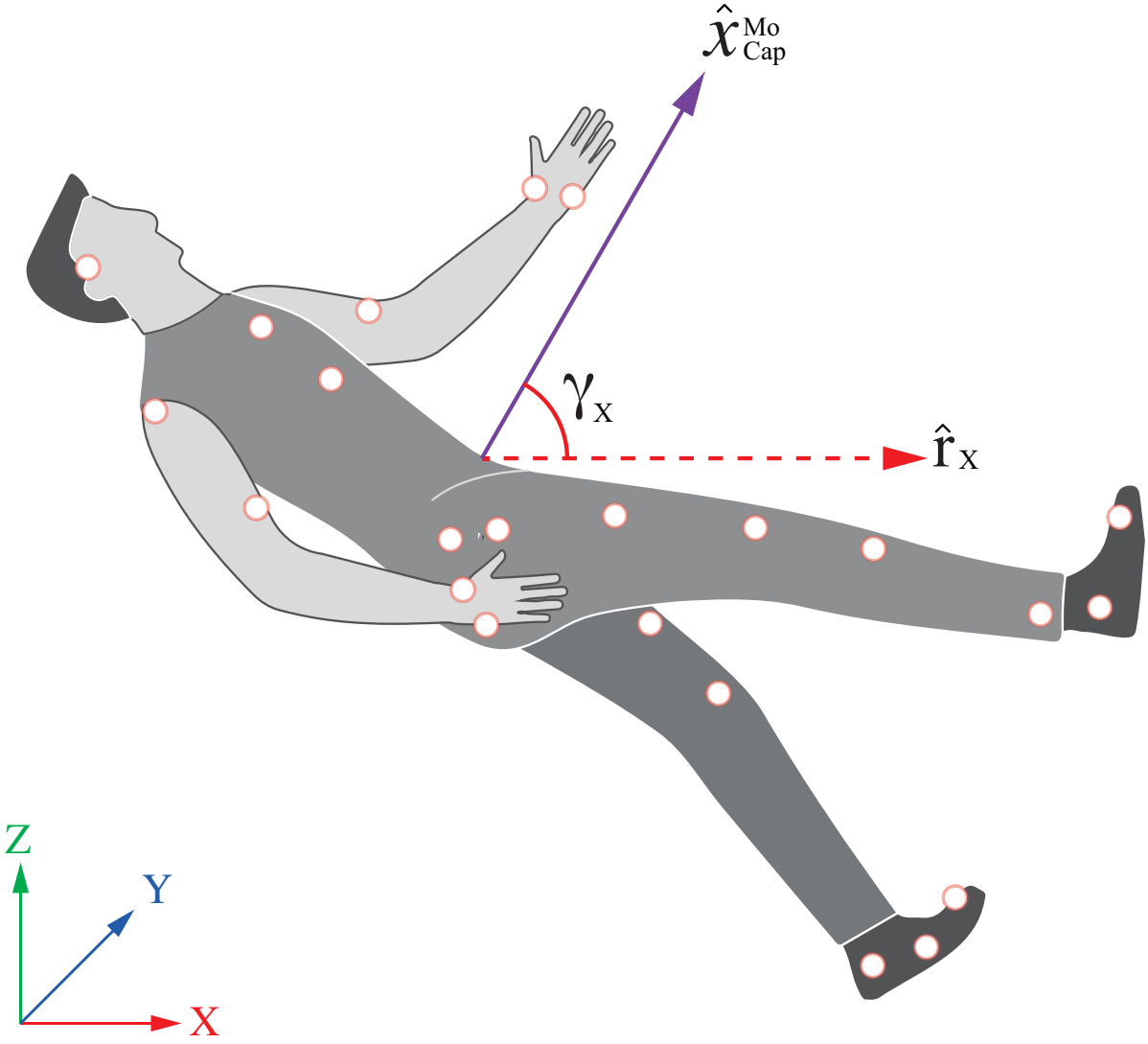


Figure 3-3.: Forward direction alignment of the MoCap data Towards the X axis

We determined the required rotation axis $\hat{r}$ using the cross product between the desired advance direction and the estimated forward direction of the data:

$$\hat{r} = \frac{\begin{pmatrix}1\\0\\0\end{pmatrix} \times \hat{x}_{MoCap}}{\left|\begin{pmatrix}1\\0\\0\end{pmatrix} \times \hat{x}_{MoCap}\right|}. \tag{3-4}$$

We obtained the required rotation angle γ_x using the dot product relation between the two vectors:

$$\gamma_x = acos\left(\begin{pmatrix}1\\0\\0\end{pmatrix} \cdot \hat{x}_{MoCap}\right). \tag{3-5}$$

Then we employed the well-known Rodrigues' Rotation Formula [Belongie, 2023] for evaluating the required rotation matrix to be applied to all of the position vectors within the interest demonstration:

$$\begin{pmatrix} cos(\gamma_x) + \hat{r_x}(1 - cos(\gamma_x)) & \hat{r_x}\hat{r_y}(1 - cos(\gamma_x)) - \hat{r_z}sin(\gamma_x) & \hat{r_x}\hat{r_z}(1 - cos(\gamma_x)) + \hat{r_y}sin(\gamma_x) \\ \hat{r_y}\hat{r_x}(1 - cos(\gamma_x)) + \hat{r_z}sin(\gamma_x) & cos(\gamma_x) + \hat{r_y}(1 - cos(\gamma_x)) & \hat{r_y}\hat{r_z}(1 - cos(\gamma_x)) + \hat{r_x}sin(\gamma_x) \\ \hat{r_z}\hat{r_x}(1 - cos(\gamma_x)) - \hat{r_y}sin(\gamma_x) & \hat{r_z}\hat{r_y}(1 - cos(\gamma_x)) - \hat{r_x}sin(\gamma_x) & cos(\gamma_x) + \hat{r_z}(1 - cos(\gamma_x)) \end{pmatrix}, \tag{3-6}$$

where $\hat{r_x}$, $\hat{r_y}$, and $\hat{r_z}$ are the x, y, and z components of the $\hat{r}$ unitary direction vector.

Head and Feet Orientation: With the previous rotation, the forward advance direction corresponds to the X+ direction onto the world frame. However, we need to adjust the rotation around the new X-axis to set the head towards the Z+ direction and the feet to be supported at the same height. We determined the mean angle between the Y+ axis and the vector connecting both feet. We applied a final rotation concerning the new X-axis to guarantee the desired orientation.

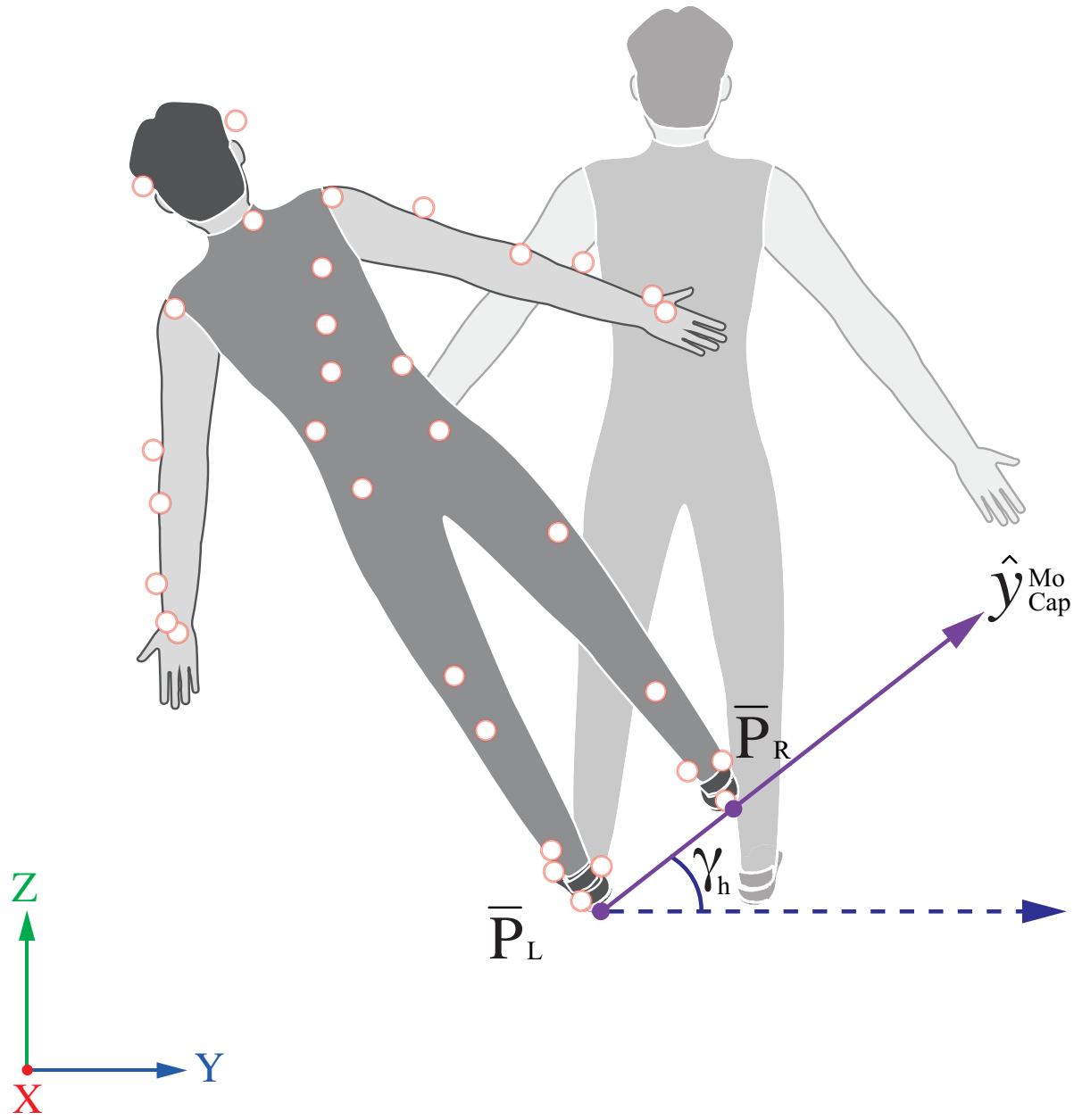


Figure 3-4.: Head and feet orientation alignment

By standardizing the orientation, we establish a consistent reference for the humanoid robot's perception of the environment. This alignment allows for a more straightforward interpretation and mapping of the human motions to the robot's coordinate system. We establish a unified and standardized reference frame for the captured human motions by performing these preprocessing steps. This alignment enables a seamless transfer of expertise from the human demonstrators to the humanoid robot, as the motions are now represented in a consistent and compatible format within the robot's task space.

3.3. Features Mapping - The Correspondence Problem

The Correspondence Problem in imitation learning addresses the challenge of mapping human demonstrations to the task space of humanoid robots [Nehaniv and Dautenhahn, 2002]. While humans and robots share similar characteristics and motions, their geometries are not identical. The differences in dimensions, Degrees of Freedom (DoFs), inertia, and structural variations pose significant obstacles in establishing a direct correspondence between human and robot states.

3.3.1. Dimensional Disparities

Humans possess various physical characteristics, including varying body sizes, limb proportions, and joint ranges. On the other hand, humanoid robots are designed with specific dimensions and mechanical properties which may differ from those of humans. For instance, the area of the feet of small-size robots tends to be larger (in proportion) than the area of human feet. These differences can result in disparities in joint angles, limb lengths, and overall body structure. In particular, both robots considered for this project are significantly smaller than humans. An average American adult's height is about 1.75 meters, while the robot Darwin Mini's height is about 0.27 meters, and the robot MAXE-2's height is about 0.40 meters. Considering this, even simple parameters such as the step length of a human, which is about 0.7 meters, are impossible to directly imitate by the robots evidencing the requirement of normalizing the features to be imitated. Figure **3-5** compares human and robot proportions.

3.3.2. DoF and Joints Misalignment

The Degrees of Freedom (DoFs) of human and robot joints may not align perfectly, leading to challenges in accurately transferring human motion to the robot's kinematic chain. Humans have complex musculoskeletal systems with numerous DoFs in their joints. The human body consists of more than 200 bones and approximately 230 individual joints, although not all are independent. In contrast, humanoid robots are designed with a predetermined set of DoFs, which is generally significantly smaller than that of humans.

For instance, in the context of this project, the Darwin robot possesses 16 individual joints, while MAXE has 17. However, it's important to note that even more anthropomorphic robots do not necessarily correspond precisely to the number and type of joints found in humans. Task requirements, mechanical constraints, and technological limitations often drive a humanoid robot's specific choice of DoFs.

Apart from the differences in DoF counts, the structure of joints can also vary between humans and robots. While some joints may share the same number of DoFs, such as the wrist with three DoFs, each joint's location, and intrinsic mechanics can differ. For example,

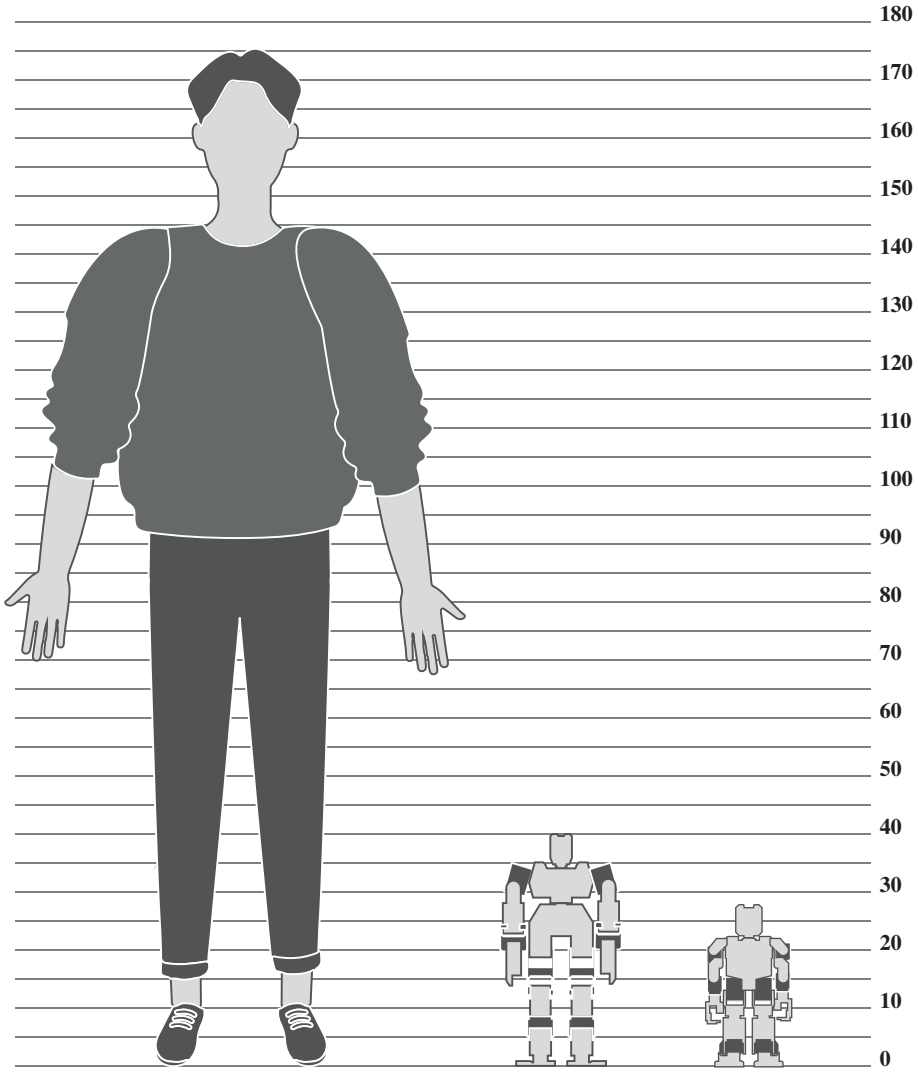


Figure 3-5.: Human vs. Robots' height

the wrist joint in a human and a robot may possess three rotational DoFs, but the specific arrangement and range of motion may differ.

3.3.3. Inertia and Mass Distribution Differences

Even when robots' geometry closely resembles humans, differences in the mass distribution can significantly affect the stability of movements and poses. Due to variations in body composition and mechanical design, poses or movements that are stable for humans may not be stable for robots, and vice versa. Therefore, when performing imitation learning, it is crucial to consider the robot's stability throughout the motion.
The mass distribution differences between humans and robots can result in variations in the

dynamics of motion. These differences can lead to differences in joint torques, accelerations, and overall stability. For example, a human may naturally maintain balance in a specific pose due to mass distribution within their body. In contrast, a robot with a different mass distribution may require additional control efforts to achieve the same stability.
When mapping human demonstrations onto a robot, it becomes essential to account for the differences in mass distribution. This may involve adjustments in the control strategies or motion planning to ensure the robot's stability is preserved throughout the imitation process. By considering the specific mass distribution characteristics of the robot, the imitation learning algorithm can generate motions similar to human demonstrations and are dynamically feasible and stable for the robot. Similarly, by carefully considering the impact of inertia and mass distribution differences, imitation learning algorithms can facilitate the transfer of human expertise to humanoid robots while ensuring stability during task execution. These considerations are vital in enabling robots to perform tasks with human-like characteristics while maintaining their unique stability requirements.

3.3.4. Normalized Features Imitation

In imitation learning, using normalized features can facilitate the mapping of the human demonstration space X_{human} and the humanoid robot's task space X_{robot} to a common imitation space $X_{imitation}$. Normalization techniques aim to establish mappings that take the data from the human demonstrators and from the robots to a shared space where they become comparable, accounting for variations in subjects' heights and proportions (i.e., $\phi_{human} : x_{human} \in X_{human} \rightarrow x_{imitation} \in X_{imitation}$ and $\phi_{robot} : x_{robot} \in X_{robot} \rightarrow x_{imitation} \in X_{imitation}$ as presented in figure **3-6**). This subsection explores scaling functions and limb-based scaling approaches to address these challenges.
One approach to handle scaling is to utilize a mapping function $\phi_{subject}$ that considers the subject's height, whether a human or a robot. Applying this function allows interest points to be scaled proportionally, ensuring consistency across different subjects. For example, if P represents an interest point, the normalized position $\hat{P}$ can be obtained as:

$$\hat{P} = \phi_{subject}(P, h_{subject}) = \frac{P}{h_{subject}}, \tag{3-7}$$

where $h_{subject}$ represents the height of the subject and $\phi_{subject}$ is the scaling function.
However, scaling based solely on height may not be sufficient, as the proportions of the links in the human body and the robot may differ. Fully extended limbs in humans may not align with fully extended limbs in robots. An alternative is to apply scaling on a limb-by-limb basis to address this. In this approach, a fully extended limb corresponds to a value of 1, while a full contraction (which may be unfeasible in practice) corresponds to 0. This limb-based scaling accounts for differences in limb proportions between humans and robots:

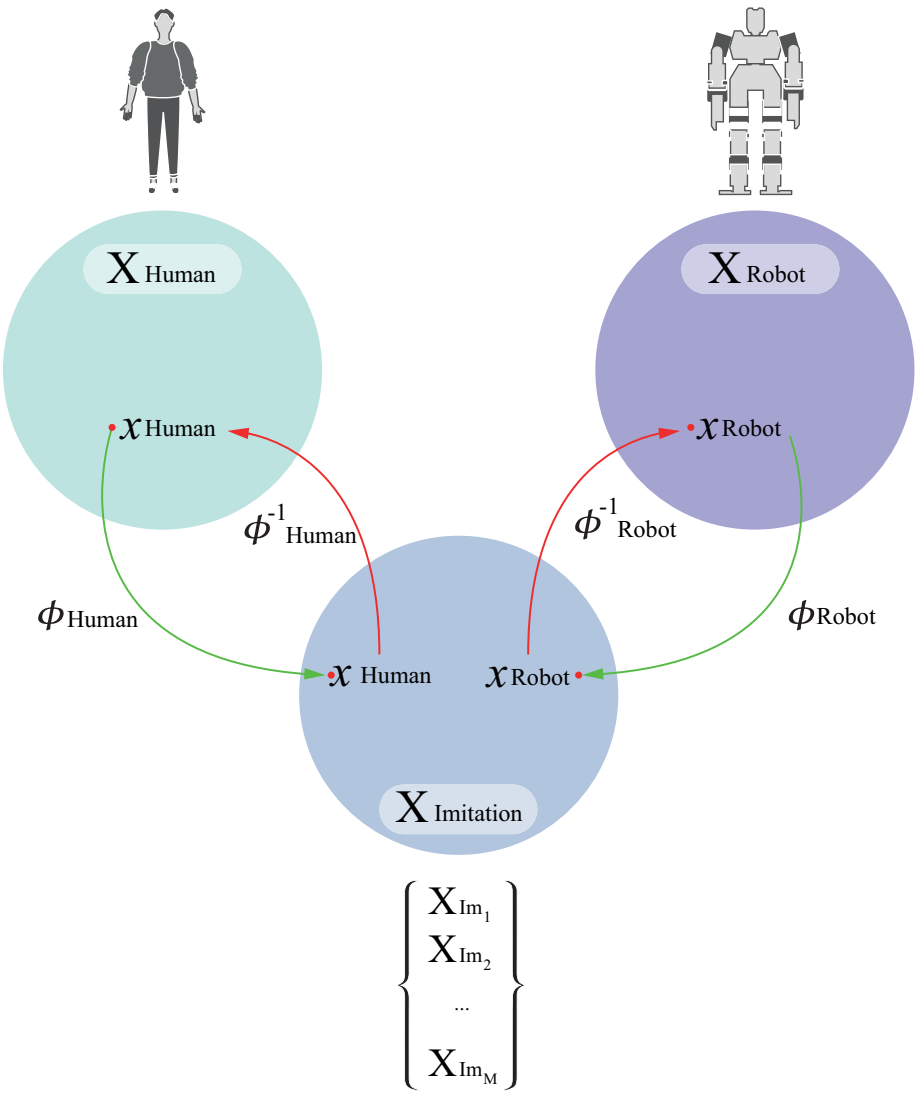


Figure 3-6.: Human, Robot, and Imitation Spaces and their relating functions

$$\hat{P} = \phi(P, L_{limb}) = \frac{P}{L_{limb}}, \tag{3-8}$$

where L_{limb} represents the length of the fully extended limb.
Note that whichever mapping function is chosen implies the existence of the Imitation Jacobian $J_{imitation}$ for the robot:

$$\dot{x}_{imitation}(q) = \frac{\partial x_{imitation}(q)}{\partial q}\frac{dq}{dt} = J_{imitation}(q)\dot{q}. \tag{3-9}$$

This matrix enables kinematic control (as presented in Section 2.2.1) onto the robot, allowing

the setting of the interest positions in the imitation space. However, this *direct approach* may not be the best given the previously mentioned limitations on imitation.

In our specific case, considering the characteristics and requirements of our robots, an appropriate scaling approach is crucial for achieving accurate imitation. The limb-based scaling method proposed in [Koenemann et al., 2014] is the most promising among the various approaches discussed. This approach considers the differences in proportions between humans and robots, scaling the interest vectors based on the height ratio and considering the relative differences from a reference pose. By adopting this approach, we aim to address the challenges posed by the geometry and dimensions of our robots, enabling a more accurate and effective transfer of human motion to the robot's kinematic chain.

3.4. Policy Learning

Once the vector to be imitated has been determined (i.e., what to imitate), the next step is determining the policy responsible for achieving that imitation (i.e., how to imitate). Several approaches to performing imitation learning are broadly categorized into three main branches: Geometric methods, Optimization-based methods, and GANs-based methods.

3.4.1. Geometric Methods

Geometric methods establish a closed mapping between the human and robot domains using linear or non-linear transformations. These methods often rely on defining a set of correspondences between critical points or features in the human and robot spaces. By mapping the human motion to the robot's kinematic chain, geometric methods aim to achieve accurate imitation. However, they may face challenges in handling complex dynamics and capturing fine-grained details of the motion. One of the most straightforward approaches of this category implies the imposition of specific high-level step parameters such as step length, height, width, and duration.

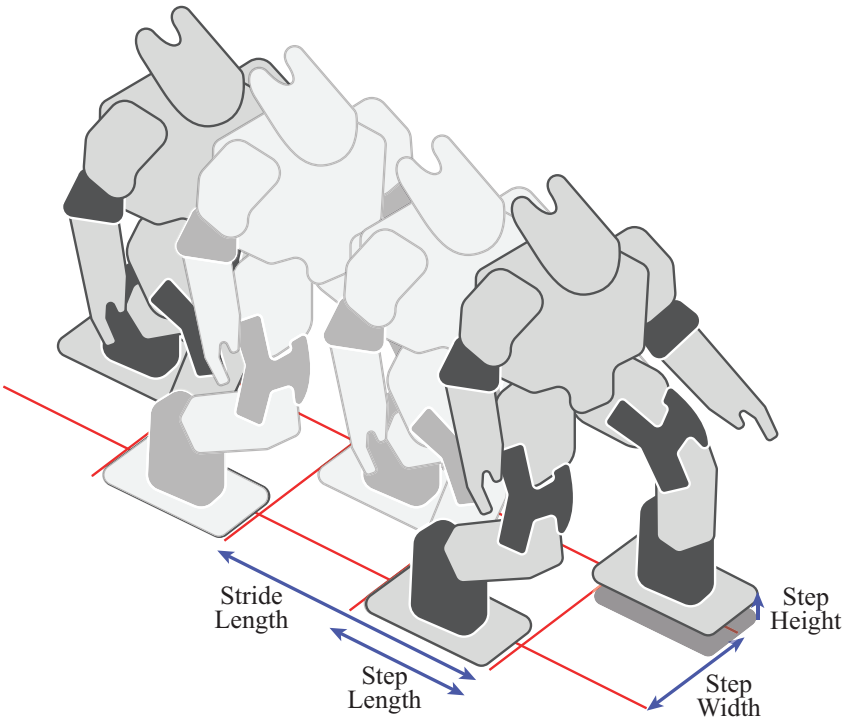


Figure 3-7.: Example of step parameters that may be directly imitated (in an appropriate common space)

3.4.2. Optimization-based Methods

Optimization-based methods approach imitation learning as a problem of maximizing a reward function. These methods involve defining a parametric structure for the robot's policy and iteratively tuning the robot's policy parameters to maximize the interest function as Equation (3-2). By formulating the imitation problem as an optimization task, these methods can account for various constraints and objectives. However, the optimization process can be computationally demanding, and the choice of the reward function and optimization algorithm plays a crucial role in achieving successful imitation. This approach can be integrated with others to fine-tune previous results and/or consider the similitude metric to guide the overall process.

3.4.3. GANs-based Methods

GANs (Generative Adversarial Networks) have also emerged as a powerful approach to imitation learning. GANs consist of two neural networks, a generator, and a discriminator, that compete against each other in a minimax game. The generator network learns to generate samples that mimic the training data distribution (human demonstrations), while the discriminator network learns to distinguish between the real and generated samples. GAN-based methods offer the potential for generating highly realistic and diverse imitation behaviors by leveraging the generative capabilities of neural networks. However, they often require much training data and are more computationally demanding than other approaches.

3.5. Proposed approach: Dynamic ZMP-Based Retargeting

We will take as base the work of [Koenemann et al., 2014]. In our work, we aim to extend Koenemann's approach by incorporating the Zero Moment Point (ZMP) criterion for dynamic stability and employing a trajectory-based retargeting process. By incorporating these modifications, we seek to achieve dynamically stable imitation while considering the specific characteristics of our robots.

3.5.1. Original Koenemann's Approach

In [Koenemann et al., 2014], they proposed a geometric method for imitation learning, which initially focuses on imitating static poses based on scaled vectors based on a so-called T-pose, utilizes the criterion of Center of Mass (CoM) projection to ensure stability during imitation, and applies a *Retargeting* process to fix the resulting position.

Imitation Vectors from T-Pose

To have a common initial position compatible with both the robot x^{ref}_{robot} and the human x^{ref}_{human}, they consider it as a T-pose where both legs and arms are fully extended. They use a limb scaling method as in Equation (3-8), where the scale factor corresponds to the relation between the robots and the corresponding human limb at this T-Pose. However, they do not map the interest vectors directly but consider the difference between the T-Pose and the interest position:

$$x_{imitation} = \phi_{human}(x_{human}) = \frac{x_{human} - x^{ref}_{human}}{L^{H}_{limb}}, \tag{3-10}$$

where L^{H}_{limb} is the fully extended length of the interest limb of the human demonstrator.

$$x_{robot} = \phi^{-1}_{robot}(x_{imitation}) = \phi^{-1}_{robot}(\phi_{human}(x_{human})) = \left(\frac{x_{human} - x^{ref}_{human}}{L^{H}_{limb}} \right) L^{R}_{limb} + x^{ref}_{robot}, \tag{3-11}$$

where L^{R}_{limb} is the fully extended length of the interest limb of the robot.

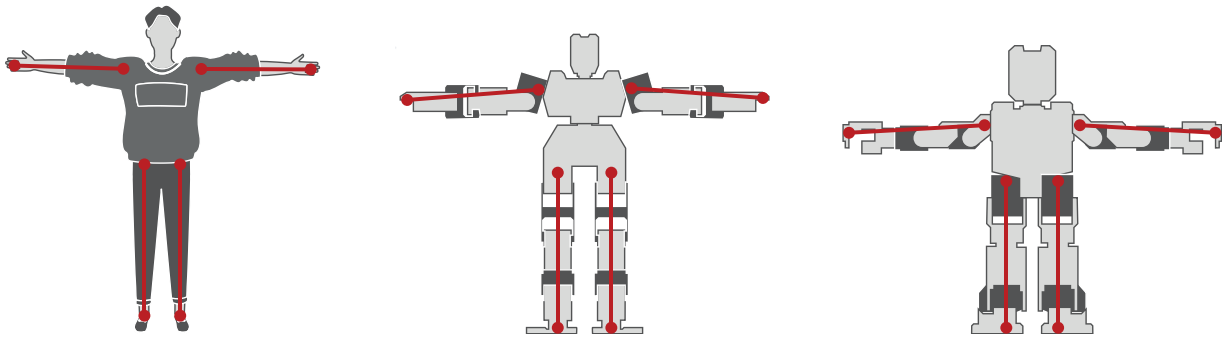


Figure 3-8.: T-Pose for Human and the two interest robots

Statically-stable Retargeting

The position of the end effectors obtained by directly tracking the interest vectors may not be stable. Therefore, the robot's state must be altered to guarantee stability. For this purpose, the work of [Koenemann et al., 2014] proposes two scenarios with a finite states machine that regulates the transition between them.

The first scenario is the case of a double support stance, where both feet are in contact with the ground. In this case, a normalized CoM offset o_{CoM} is defined:

$$o_{CoM} = \frac{(p_{CoM} - p_{L,foot}) \cdot (p_{R,foot} - p_{L,foot})}{||p_{R,foot} - p_{L,foot}||^2}, \tag{3-12}$$

where o_{CoM} is the normalized CoM offset, p_{CoM} is the position of the subject's CoM, $p_{L,foot}$ is the position of the subject's left foot, and $p_{R,foot}$ is the position of the subject's right foot.

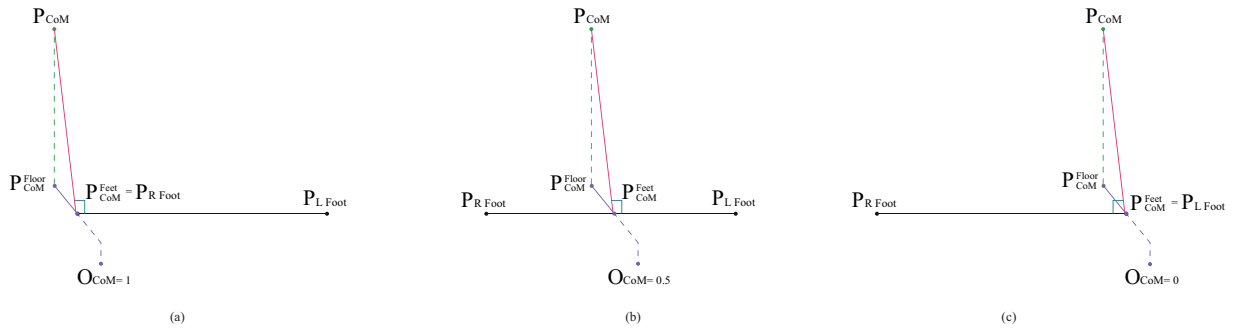


Figure 3-9.: Limit cases of the CoM offset - (a) $O_{CoM} = 1$ projection of the CoM on the right foot, (b) $O_{CoM} = 0{,}5$ projection of the CoM on the center of the feet, (c) $O_{CoM} = 0$ projection of the CoM on the left foot

Considering this offset, the outermost foot is retargeted to ensure that the normalized CoM offset of the robot corresponds to one of the human demonstrators (Note that this offset is attainable in an infinite number of points, but we are only interested in the point that belongs to the line that connects both feet).

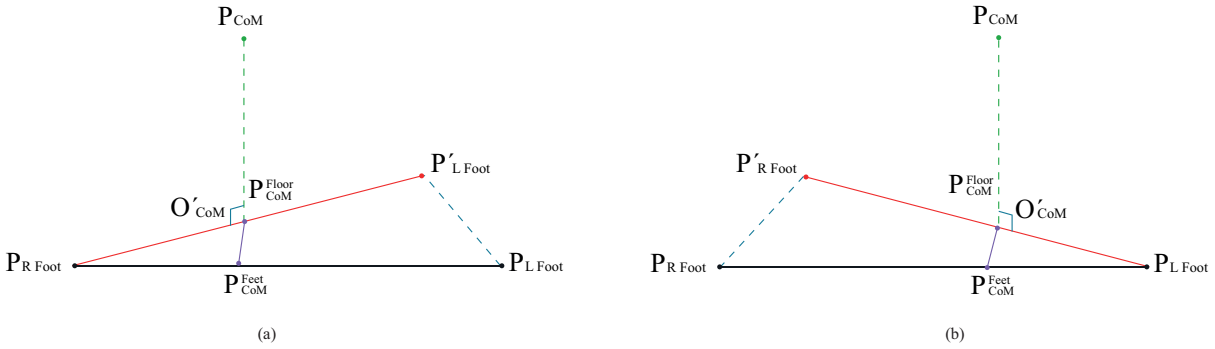


Figure 3-10.: Retargeting of the farthest foot for imposing the desired offset - (a) left foot retargeting, (b) right foot retargeting

The second scenario is the case of poses involving a simple support stance, where only one foot supports the body while the other is in the air. In this case, the CoM position is set to lie precisely at the center of the supporting foot (this would be equivalent to having a normalized CoM offset of $o_{CoM} = 0$ for the right foot and $0_{CoM} = 1$ for the left foot). This adjustment ensures the robot maintains balance and stability during single support phases. Concerning the transitions, when the robot is in double support, and the subsequent interest pose corresponds to a simple support stance, a soft transition for the offset is determined so that in a determined time, the offset changes from its current value either to 0 or 1 (depending on the corresponding simple support foot) and only then, the new balancing foot is freed to track its desired state.

Limitations

While the Koenemann approach provides a suitable pose imitation method, it has certain limitations. One limitation lies in its stability criterion, which focuses solely on low acceleration situations and overlooks the dynamic aspects of the robot's motion. As a result, the approach may not adequately account for higher acceleration movements or scenarios that require rapid changes in the robot's pose. The transition from double support to simple support stances also introduces a delay that can lead to a desynchronization between the desired reference poses and the robot's current state. This delay may cause undesired abrupt changes in the robot's motion, resulting in a lack of smoothness and coherence during the imitation process. To address these limitations and improve upon the Koenemann approach, we propose modifications and enhancements that consider both the dynamic nature of the robot's movements and the need for seamless transitions between different stances. These refinements enhance the imitation process's stability, fluidity, and synchronization, ensuring a more accurate and natural replication of human motion.

3.5.2. Support Modes Identification

The dynamic nature of the robot's gait and the specific challenges presented in different stances influence the imitation process. Notably, the simple support phases, characterized by a smaller support polygon, imply a narrower margin for stability. Conversely, the double support phases necessitate maintaining a consistent position of the feet throughout the motion. To achieve robust imitation, it is crucial to estimate the gait phases within the data accurately. For this purpose, we employ a method based on the detection of changes in the speed of the feet. Specifically, when the speed of the feet drops below a certain threshold, it indicates a transition into a support phase. We leverage this criterion to determine the timing and duration of the simple and double support phases. Figure **3-11** shows an example of this process.
In our estimation, we rely on two critical points located at the heel and the metatarsus. By analyzing the contact of these critical points with the ground and their velocities, we can infer whether the feet are in a supporting position. By precisely estimating the gait phases, we ensure compatibility between the human data and the hybrid model utilized for the robots. This compatibility allows for a more effective and reliable imitation process, enabling the humanoid robots to replicate human motion with enhanced precision and fidelity.

3.5.3. Key Frames Imitation

Once the Support mode has been identified, we set the phase transition frames and the frames when the balancing foot is at its highest as keyframes. Following the ideas presented in the work of [Ma et al., 2021], we aim to establish the dynamic imitation fixing space-time bounds around the reference behavior so that the robot's behavior is very close to the

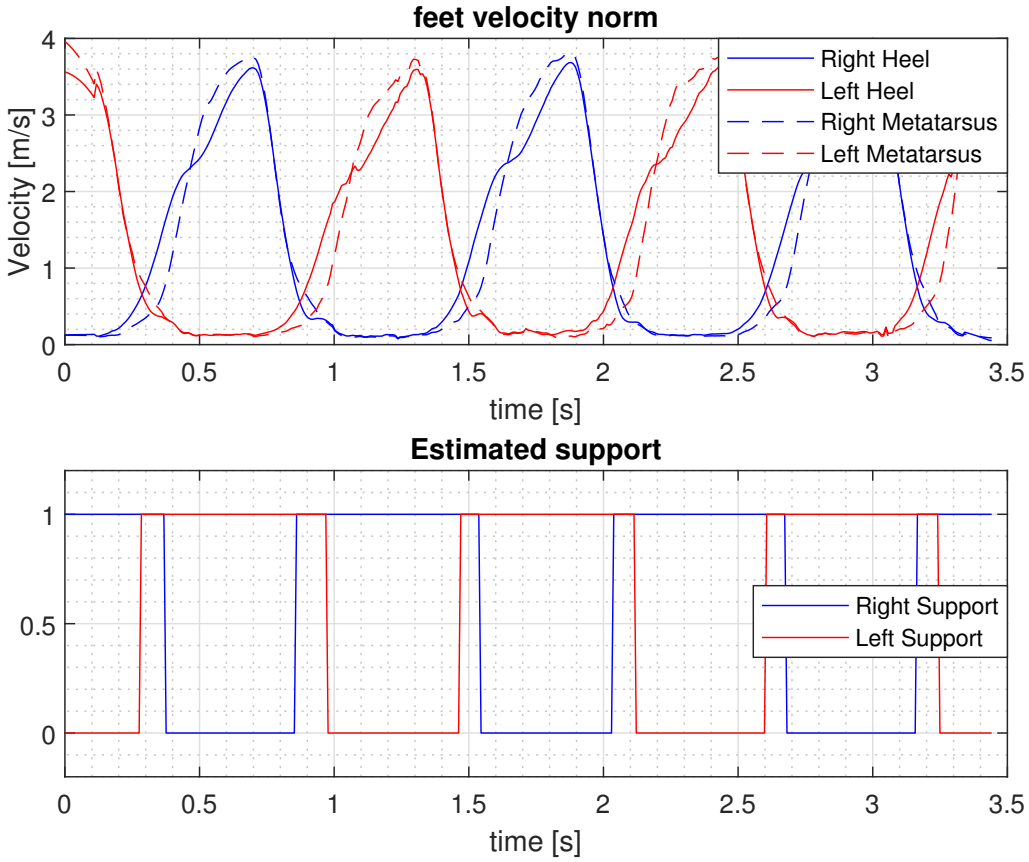


Figure 3-11.: Example of step phases estimation according to the feet key points velocities

human's at the critical points while still having certain freedom to adjust its movement at the rest of the trajectory.

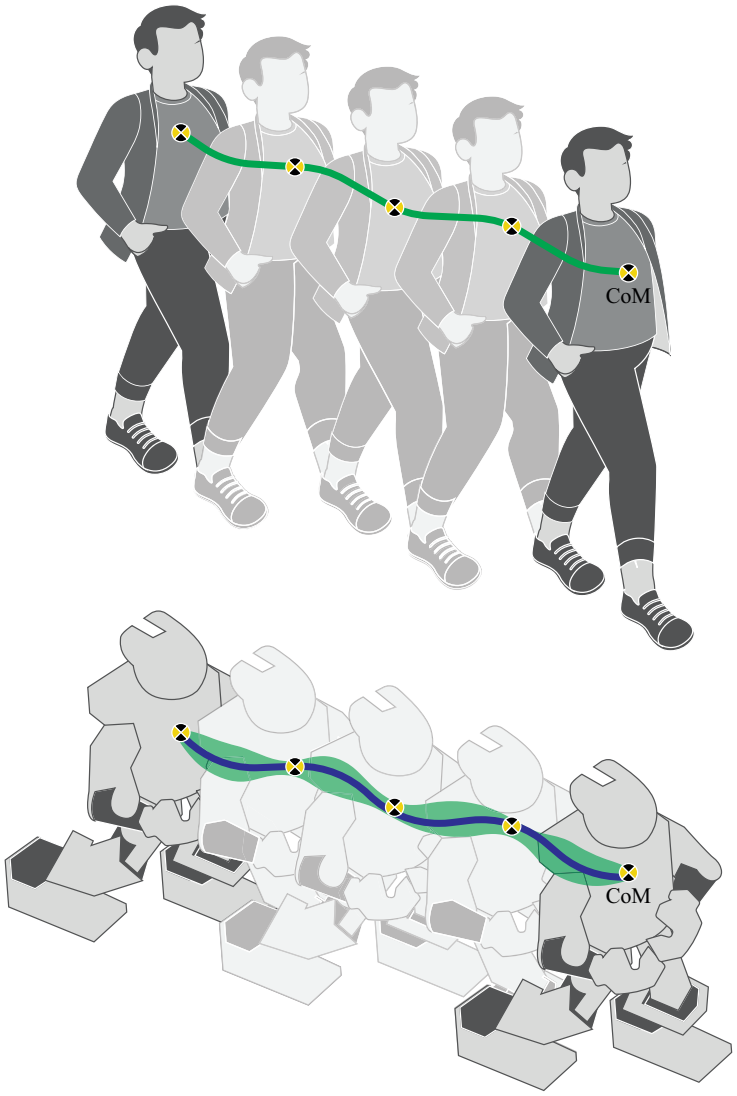


Figure 3-12.: Imitation of Keyframes and spacetime bounds around them

To determine the robot's position corresponding to the imitation vectors, we define the robot's imitation task space similarly as in Section 3.5.1. x_{robot} corresponds to the vectors that map the position of the end of each limb relative to its beginning (hands relative to shoulders and feet relative to hips).

$$x_{robot} = \begin{bmatrix} x_{Rfoot}^{Rhip^T} & x_{Lfoot}^{Lhip^T} & x_{Rhand}^{Rshoulder^T} & x_{Lhand}^{Lshoulder^T} \end{bmatrix}^T, \tag{3-13}$$

where x_{robot}^{im} corresponds to the vector in the robot's imitation task space, x_{Rfoot}^{Rhip} corresponds to the position of the robot's right foot relative to the right hip, x_{Lfoot}^{Lhip} corresponds to the position of the robot's left foot relative to the left hip, $x_{Rhand}^{Rshoulder}$ corresponds to the position

of the right-hand relative to the right shoulder, and $x_{Lhand}^{Lshoulder}$ corresponds to the position of the left hand relative to the left shoulder.

With this vector defined, now we require to determine the joint values corresponding to this vector (inverse kinematics). For this purpose, we use the calculus of the imitation Jacobian $J_{imitation}$ mentioned in Section 3.3.4. With this Jacobian, we can iteratively take any arbitrary initial position towards the interest one through

$$q_{k+1} = q_k + K_{ik} J_{imitation}^{+}(x_{robot}^{im} - x_{robot}(q_k)), \tag{3-14}$$

where K_{ik} corresponds to an appropriately tunned positive constant, $J_{imitation}^{+}$ corresponds to the pseudo-inverse of the imitation Jacobian, $x_{robot}(q_k)$ corresponds to the actual robot position, and x_{robot}^{im} corresponds to the desired position determined by the imitation data.

With the keyframes established, we adjust Bézier polynomials (see Section 2.2.2) to match the position at these keyframes maintaining the duration between keyframes as in the data. This initial approach in general does not lead to feasible trajectories, but it allows to generate an initial estimate for the center of mass accelerations faced during the execution of the gait. With this acceleration it is possible to establish a bound on the distance between the CoM and the ZMP:

$$d_{bound} = \max_{t} |x_{CoM}(t) - x_{ZMP}(t)|. \tag{3-15}$$

Note that although it is mathematically possible to adjust velocities and accelerations, these values present much noise, making the imitation process more challenging instead of improving it. Therefore, we preserved only the position information.

3.5.4. Dynamic Retargeting

The resulting imitated trajectories may be unfeasible for the robot's geometry (as self-collisions caused by the difference in feet areas) and/or may correspond to unfeasible behaviors (unstable in the sense of the ZMP stability criterion or hard-to-transfer ones). This may result in failed imitations, as presented in Figure **3-13**.

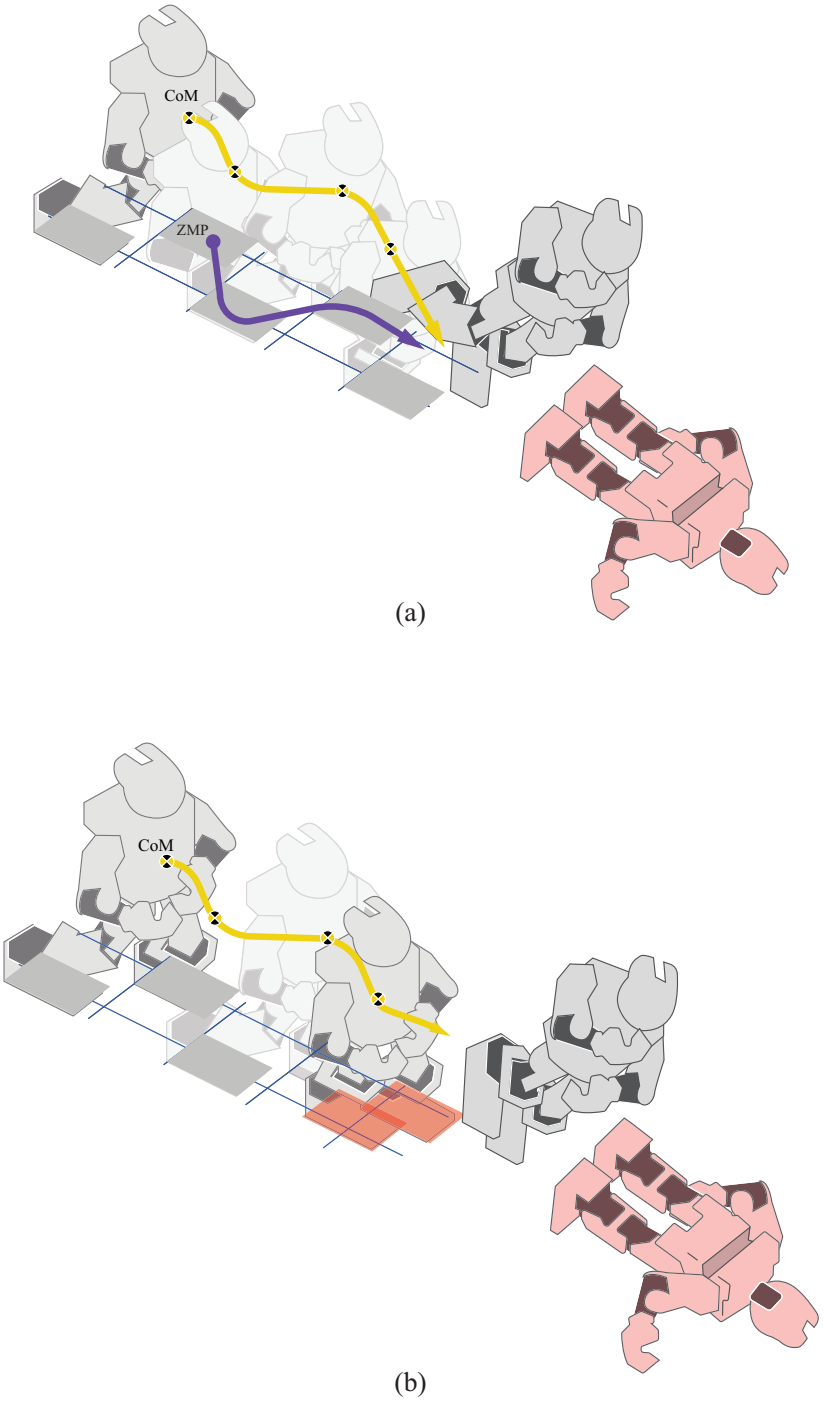


Figure 3-13.: Failed imitation - (a) Falling for self-collision, (b) falling for unstable trajectory

To correct this, we consider a retargeting process that aims to guarantee that:

- The robot's ZMP always lies within the support polygon

- There are no collisions of the robot against itself
- The steps correspond to soft impacts (see **2-3**)
- The feet remain still and flat during the double support phases

For this purpose, we define new vectors onto a new task space for the robot $X_{retarget}$ that we will denominate as:

$$x_{retarget} = \begin{bmatrix} x_{CoM}^{0^T} & x_{Bfoot}^{Sfoot^T} & \theta_{Bfoot}^{0^T} & \theta_{W}^{0^T} \end{bmatrix}^T, \tag{3-16}$$

where $x_{retarget} \in X_{retarget}$ corresponds to the new robot's task space vector, x_{CoM}^{0} corresponds to the position of the Robot's CoM relative to the world frame, x_{Bfoot}^{Sfoot} corresponds to the position of the robot's balancing foot relative to the support foot, θ_{Bfoot}^{0} corresponds to the orientation of the robot's balancing foot relative to the world frame, and θ_{W}^{0} corresponds to the orientation of the torso relative to the world frame.
Note that as the support phases change, the assignation of support and balancing foot also does. This corresponds to the hybrid nature of the models employed for the robot (see Section 2.1) and requires the proper transition maps.
With this new task space vector and its corresponding Jacobian, we can adjust the positions of the keyframes to fulfill the requirements mentioned above.

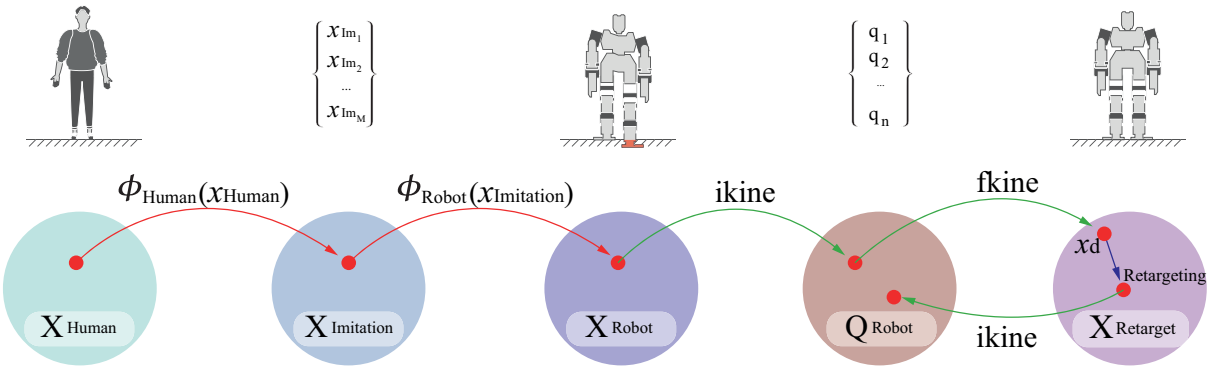


Figure 3-14.: Imitation Workflow - Features extraction, Features Imitation, and Dynamic Retargeting

Depending on the value obtained for d_{bound} (See Equation 3-15), there are two possible scenarios:

- $d_{bound} < f_w$: It is feasible to set a CoM position that maintains the ZMP within the support polygon when followed. This position can be obtained setting the position at the keyFrames within the support polygon respecting the bound (possibly with an additional margin for robustness).

- $d_{bound} > f_w$: It is unfeasible to set a CoM position that when followed, maintains the ZMP within the support polygon. In this case, the overall tracking speed for the gait must be slowed down until we revert back to the previous scenario.

Once the feasible region for the CoM's and the overall tracking speed have been established according to d_{bound}, the correction in $X_{retarget}$ is made in four steps:

1. The relative vertical position of the feet is set to 0 at the start and end of double support phases avoiding floor perforation and improving the smoothness of the trajectory.
2. The relative horizontal position of the feet is set to the width of the foot plus a small margin to avoid self-collisions.
3. The position of the CoM is adjusted at the keyframes so that the ZMP always lies on the support polygon (as the accelerations are bounded, the distance between the CoM and the ZMP is also bounded, therefore if the support polygon within that bound contains the CoM projection, the ZMP will also be contained).
4. If the desired position is geometrically unreachable for the robot, the Z component of the CoM is lowered, maintaining the remaining elements in their previously defined value.

Once the correction is made in the retargeting space, we softly interpolate between the recently obtained positions obtaining a consistent and stable trajectory. However, although stable, this imitated trajectory does not always start from a double support position and may end at any arbitrary position. To solve this issue, we added keyframes at the start and end of the gait to guarantee the initial and final conditions of the gait to be at double support stages with both feet at the same X coordinate. Note that the walking cycle consists on

- double support phase (conventionally over the right foot),
- simple support phase over left foot,
- double support phase (conventionally over the left foot), and
- simple support phase over the right foot.

Therefore, the retargeting process of each frame depends on the corresponding phase, and the additional keyframes will correspond to a transition from a double support phase to the initial phase of the data.

4. Reinforcement Learning

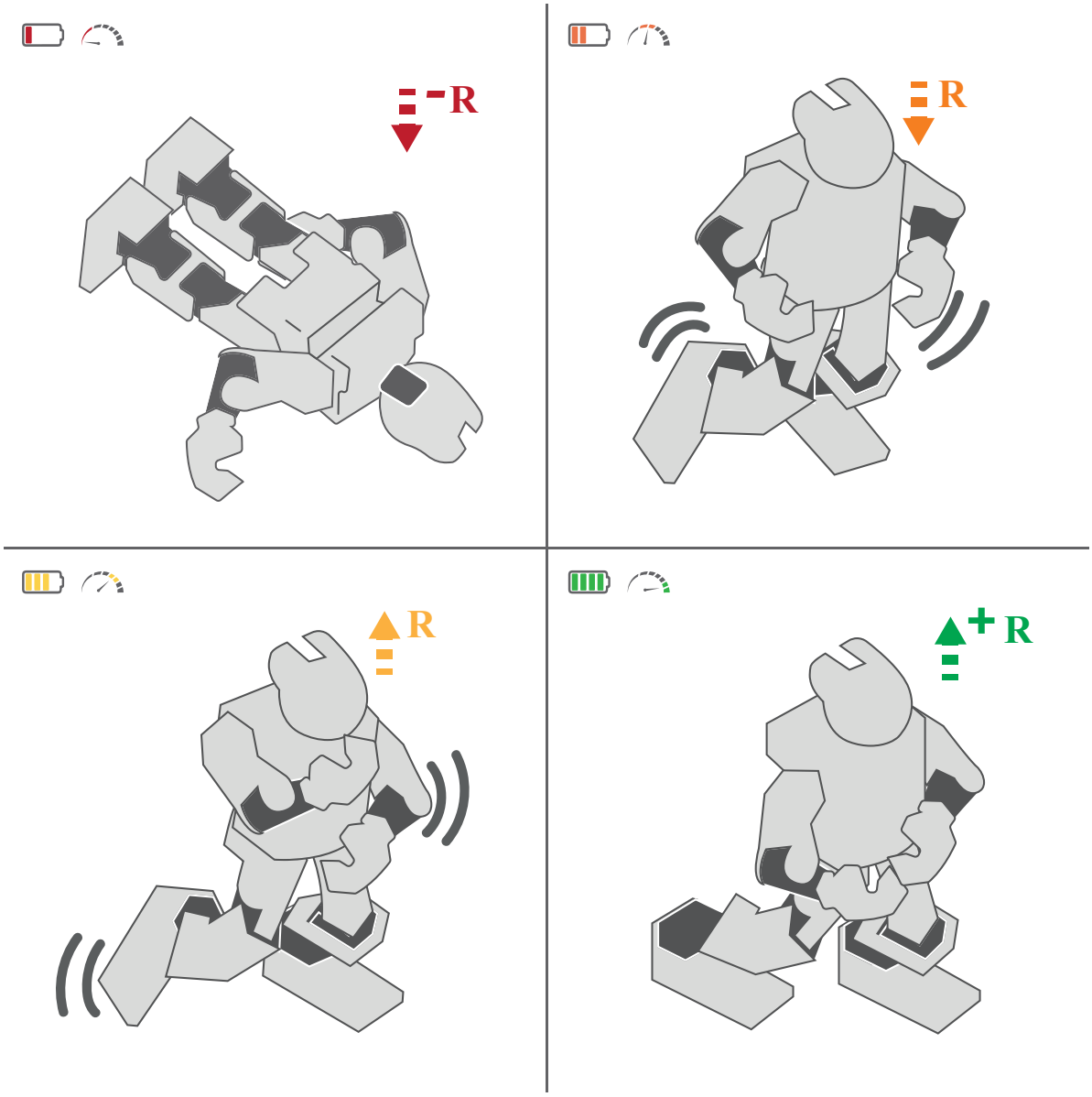


Figure 4-1.: Reinforcement Learning Framework - Iteratively testing different policies to maximize the expected Reward R (Stability, high velocity, and low torque consumption)

This chapter delves into the field of Reinforcement Learning, focusing specifically on its application to the challenging task of humanoid locomotion. Reinforcement learning is a powerful paradigm in artificial intelligence and robotics that enables agents to learn optimized behaviors through interaction with their environment. While reinforcement learning has gained significant attention for its ability to tackle complex tasks and adapt to dynamic environments, we emphasize developing a tailored approach for humanoid locomotion. In this chapter, we aim to provide a comprehensive exploration of reinforcement learning techniques and methodologies specifically designed to address the requirements of humanoid locomotion. Our approach utilizes a parametric structured policy incorporating well-established robotics principles to control the robot's movements effectively. Furthermore, we employ the Augmented Random Search algorithm to optimize the policy's parameters, ensuring the agent learns efficient and adaptive locomotion strategies. Additionally, we draw inspiration from [Tan et al., 2018] approach to transferability in reinforcement learning. We incorporate ideas from robust control, refine our models, and actively avoid hard-to-transfer states to enhance the transferability of learned policies. Considering these factors, we strive to develop robust and versatile locomotion controllers that can be successfully deployed on physical humanoid robots.

The chapter begins with an overview of reinforcement learning in Section 4.1, where we define the core concepts and terminology necessary for a solid understanding of the topic. Section 4.2 focuses on designing and constructing reward functions, a crucial aspect of reinforcement learning. We delve into the intricate process of defining appropriate reward functions that incentivize desired agent behavior while considering trade-offs and balancing conflicting objectives. By carefully engineering reward functions, we can shape the learning process and guide agents toward optimal behaviors. Moving forward, Section 4.3 explores different policy structures, which determine the agent's decision-making mechanism. We discuss two prominent approaches: Q-learning and parametric structured policies. A detailed examination of these policy structures gives us insights into their strengths, limitations, and suitability for various tasks and domains. Section 4.4 shifts the focus to optimization algorithms employed in reinforcement learning. We present a brief overview of optimization techniques for learning policies that maximize cumulative rewards. Specifically, we delve into the Augmented Random Search algorithm and its adaptive parameter update mechanism, which enables efficient exploration and exploitation in the learning process. Lastly, in Section 4.5, we comment on how to tackle a significant challenge in reinforcement learning, the reality gap. This section explores the discrepancy between training in simulated environments and deploying learned policies in the real world. We discuss the complexities, limitations, and potential solutions to bridge this gap, enabling successful policy transfer and deployment of autonomous systems in real-world scenarios.

4.1. What is Reinforcement Learning (RL)

Reinforcement learning (RL) is a subfield of artificial intelligence that addresses the challenge of learning how to make decisions in an environment to maximize cumulative rewards. Unlike other machine learning approaches that rely on labeled data, reinforcement learning is based on an interactive learning paradigm where an agent learns through trial and error [Sutton and Barto, 2008]. In RL, an agent interacts with an environment sequentially. At each time step, the agent observes the current state of the environment and selects an action to perform. The environment then transitions to a new state, and the agent receives a numerical reward signal reflecting the state-action pair's desirability. The agent's goal is to learn a policy, a mapping from states to actions, that maximizes the expected cumulative reward over time.

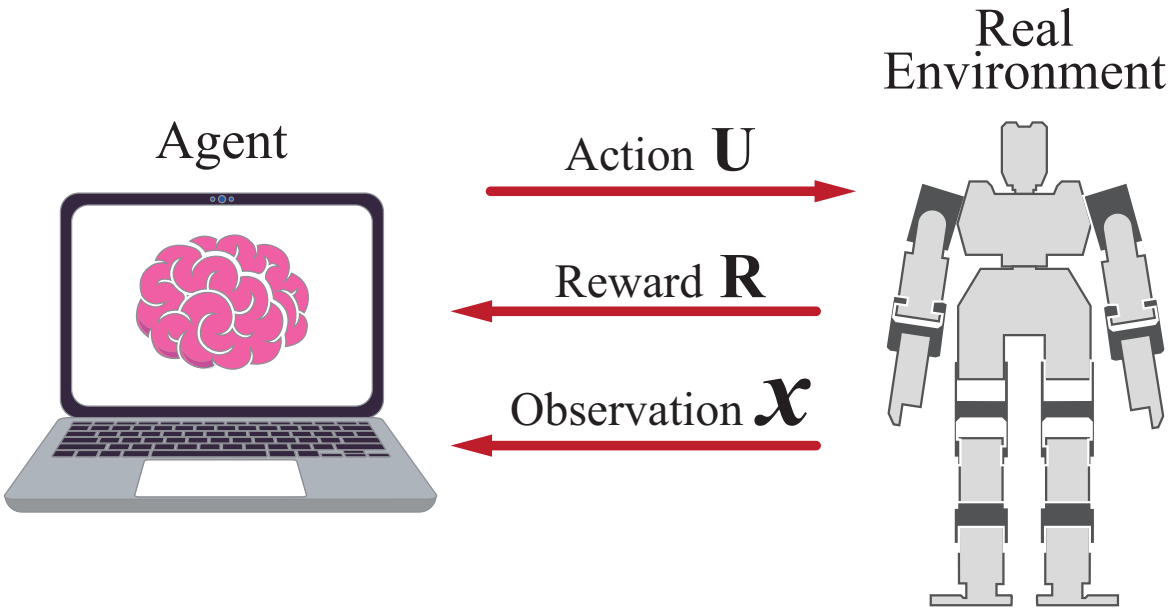


Figure 4-2.: Reinforcement Learning Standard Diagram - The agent decides which actions to take based on the information of the state of the system and the received rewards

One key distinction of reinforcement learning is its focus on learning from experience rather than relying on explicitly labeled data. The agent learns by repeatedly exploring the environment, taking action, and observing the consequences. Through this iterative process, the agent gradually improves its decision-making capabilities based on the received rewards and the observed state-action transitions.
Reinforcement learning operates in both discrete and continuous domains. In this context, we are specifically interested in the continuous domain, where the state and action spaces are continuous and can have infinite possible values. Continuous RL presents additional challenges compared to discrete RL due to the need for function approximation and the

complexity of optimizing policies in high-dimensional spaces. RL algorithms often rely on function approximators, such as neural networks, to estimate the value function or the policy in the continuous domain. These function approximators allow agents to generalize knowledge across similar states or actions and effectively handle vast, continuous state-action spaces. While many RL algorithms are based on Markov Decision Processes (MDPs), in this thesis, we are mainly focused on the continuous domain and less concerned with the formal MDP framework. Instead, we emphasize the practical aspects of RL in continuous spaces, such as algorithm design, policy representation, and optimization techniques.
When considering the essential elements of a reinforcement learning algorithm in the continuous domain, three key components stand out:

- **Reward function:** The reward function is crucial in reinforcement learning as it defines the goal and guides the learning process. It assigns a numerical value to each state-action pair, indicating the desirability or utility of that particular combination. Designing an appropriate reward function is crucial for shaping and driving the agent's behavior toward the desired objectives. A well-designed reward function should align with the problem's objectives and provide meaningful feedback to guide the learning process effectively.

- **Policy structure:** The policy structure determines how the agent selects actions based on the observed states. It represents the agent's decision-making strategy and defines the mapping between states and actions. The policy structure can take various forms, such as Q-Learning, which estimates the value of taking action in a given state, or parametric structured policies that directly map states to actions using parameterized functions. The choice of policy structure impacts the agent's exploration and exploitation trade-off and influences the learning efficiency and convergence properties.

- **Optimization algorithm:** The optimization algorithm drives the learning process by iteratively updating the policy based on observed rewards and state transitions. It determines how the agent adapts behavior to maximize cumulative rewards over time. Various optimization algorithms exist, such as the Augmented Random Search (ARS-V1) algorithm, Proximal Policy Optimization (PPO), or Trust Region Policy Optimization (TRPO). These algorithms leverage mathematical techniques to find the optimal policy by iteratively adjusting the policy parameters in response to observed rewards and environmental interactions.

4.2. Reward Function

The reward function is a critical component of the reinforcement learning algorithm as it gives the agent feedback on its actions and guides its learning process. In this section, we present the design of our reward function, which aims to promote the desired behaviors

and penalize the undesired ones in the context of bipedal locomotion. Our proposed reward function, denoted as $R(\theta)$, considers two primary factors: the movement velocity and the torque consumption. The balance between these factors is crucial for achieving effective and efficient walking patterns. The reward function can be defined as follows:

$$R(\theta) = \beta v_x(t) - (1-\beta)u(t)'Qu(t), \tag{4-1}$$

Where $v_x(t)$ represents the robot's velocity in the advanced direction, and $u(t)$ denotes the torque applied to the robot's joints. The term β captures the relative importance of torque consumption compared with the velocity. Q corresponds to a weighting matrix that reflects the individual cost of using each servomotor. In our case, all the values of matrix Q are identical since the motors of both interest robots have the exact specifications and are considered equally relevant.
However, relying solely on these initial terms proved insufficient for valid walking patterns. Initialized with random values for these parameters, many experiments resulted in early falling or failing to move. To address this challenge, we incorporated conditional rewards and penalizations into the reward function:

$$r(\theta) = \beta v_x(t) - (1-\beta)u(t)'Qu(t) + C_{end}(t), \tag{4-2}$$

These conditional terms, denoted as $C_{end}(t)$, are applied at the end of the experiment based on the termination conditions and the time elapsed since its start. The purpose of these terms is to further guide the agent towards desired behaviors and discourage undesired ones. Specifically, if the robot falls, a penalization term is assigned an enormous negative value that increases as the fall occurs earlier. Conversely, if the robot reaches the goal position stably, a reward term is assigned a considerable positive value that increases as the goal is achieved earlier. This incentivizes the agent to develop fast and stable gait patterns.
In our approach, we incorporated significant information in the policy structure to reduce the need for extensive tuning of the reward function. By designing a policy structure that captures essential aspects of the desired behavior, we can achieve a reward function that requires fewer adjustments and fine-tuning compared to approaches that iteratively adjust the reward function based on partial results, as observed in previous studies as [Xie et al., 2019].

4.3. Policy Structure

There are several well-established methods in the state of the art to determine the mapping between the state of the environment x and the actions of the agent $u = \pi(x)$. Between them, we identify four main categories: Value-based methods, Policy Based methods, Model-Based methods, and Hybrid methods. We chose a policy-based method that employs several elements described in Section 2.2 instead of a more common agnostic approach.

4.3.1. Value-Based Methods

Value-Based Methods in reinforcement learning focus on estimating the value function $q(x, u)$, which represents the expected cumulative reward for each state or state-action pair. These methods aim to find an optimal value function or Q-function that guides decision-making. This function assigns a Q-value to each pair state action corresponding to the expected reward to be obtained following that action at the end of the experience. These methods usually have two stages, an initial training stage where the value function is iteratively updated, exploring the states-action space extensively, and a deployment stage where the value function has converged, and the agent performs at each state the action with the highest Q-value.

$$\pi(x) = \arg\max_{u} q(x, u) \tag{4-3}$$

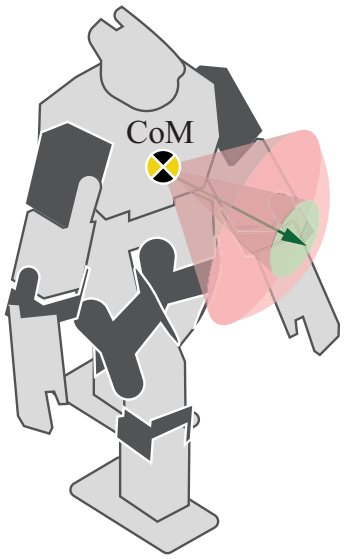


Figure 4-3.: Q-Learning Paradigm - a q-value is iteratively assigned to each pair state action so that after the training, the best action for each state corresponds to the corresponding highest q-value. For instance, the target movement direction of the CM of the robot (action space) at a specific state may correspond to expected rewards (q-values)

Examples of Value-Based Methods include:

- Q-Learning: An off-policy algorithm that learns the optimal Q-values by iteratively updating the Q-function based on observed rewards and state transitions.
  [Watkins and Dayan, 1992]

- Deep Q-Networks (DQN): A value-based method that utilizes deep neural networks to approximate the Q-values, allowing for efficient handling of high-dimensional state spaces. [Mnih et al., 2015]

Advantages:

- Suitable for problems with large state spaces.
- Can handle continuous state and action spaces.
- Can converge to an optimal policy given sufficient exploration.

Disadvantages:

- May suffer from convergence issues in high-dimensional and continuous action spaces.
- Prone to overestimation or underestimation of Q-values.
- Difficulties in handling exploration-exploitation trade-offs.

4.3.2. Policy-Based Methods

Policy-Based Methods directly learn a parameterized policy that maps states to actions. Rather than estimating value functions, they aim to optimize the policy directly to maximize the expected cumulative reward.

$$u = \pi_\theta(x) = \pi(x, \theta) \tag{4-4}$$

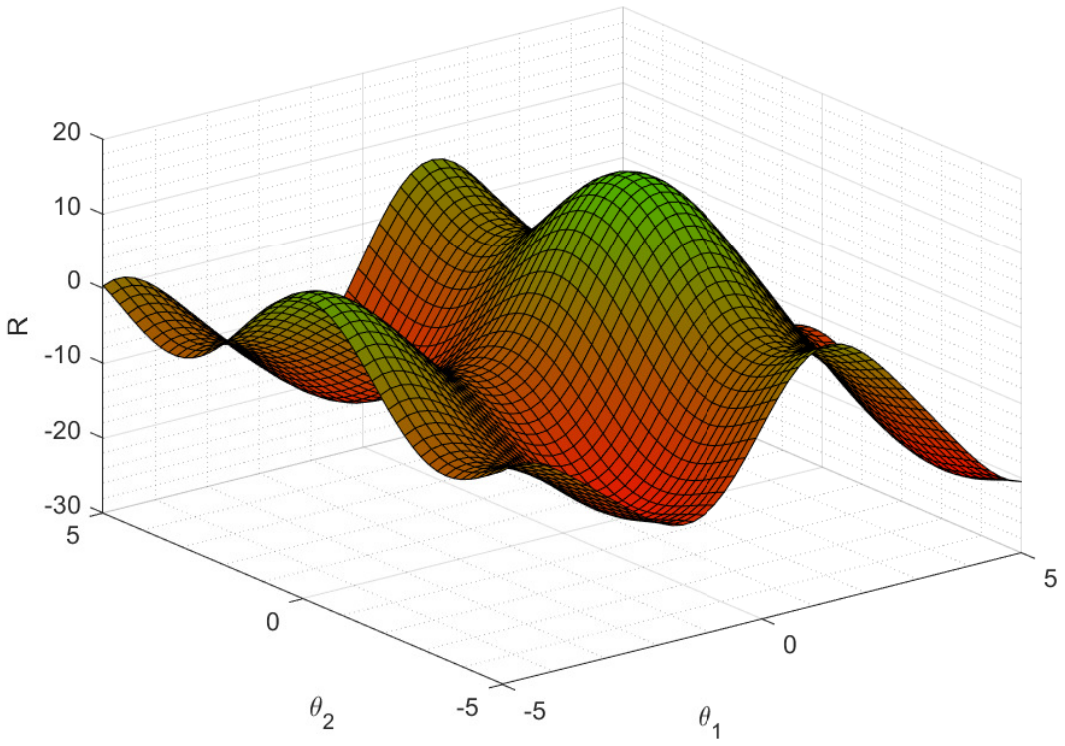


Figure 4-4.: Reward vs θ example landscape - Policy-based methods explore the parameters space Θ aiming to maximize the expected Reward R

Examples of Policy-Based Methods include:

- REINFORCE: A policy gradient method that uses Monte Carlo sampling to estimate the policy gradient and updates the policy parameters accordingly.
- Proximal Policy Optimization (PPO): A policy optimization algorithm that iteratively updates the policy using multiple epochs of stochastic gradient ascent, employing a surrogate objective function with a clipping mechanism for stability. [Heess et al., 2017]

Advantages:

- Direct Policy Optimization: Policy-based methods optimize policies directly, allowing for more flexibility in learning complex and stochastic policies.
- Continuous Action Spaces: These methods naturally handle continuous action spaces without requiring additional modifications or discretization.
- Convergence to Local Optima: Policy-based methods are less prone to getting stuck in local optima than value-based methods.

Disadvantages:

- High Variance: Policy-based methods often suffer from high variance during learning, leading to slow convergence and noisy updates.
- Sample Efficiency: These methods require more samples to learn an optimal policy than value-based methods.
- Exploration Challenges: Policy optimization can struggle with exploration, especially in large action spaces or when faced with sparse rewards.

4.3.3. Model-Based Methods

Model-Based Methods focus on learning a model of the environment dynamics, which includes the transition probabilities between states and the reward function. These methods utilize the learned model to plan and decide actions to maximize future rewards.

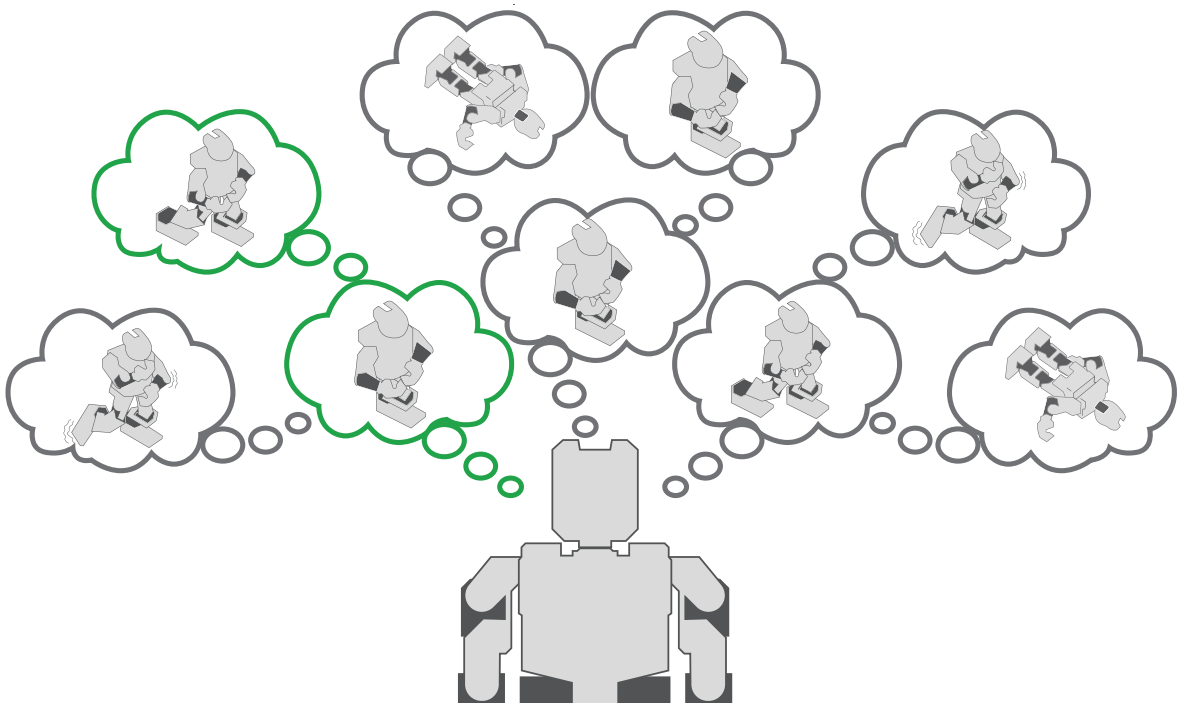


Figure 4-5.: Model-Based Methods Framework - The agent uses model information to decide the best actions in a defined forecast window

Examples of Model-Based Methods include:

- Monte Carlo Tree Search (MCTS): A planning algorithm that uses a tree structure to simulate and explore possible actions and outcomes, utilizing the learned model to guide the search process. [Świechowski et al., 2022]

- Model Predictive Control (MPC): A control method that optimizes a sequence of actions over a finite horizon using the learned model, considering the future consequences of actions. [García et al., 1989]

Advantages:

- Planning and Optimization: Model-based methods can utilize the learned model to plan and optimize actions, enabling efficient decision-making.
- Sample Efficiency: These methods can achieve higher sample efficiency by utilizing the learned model to simulate and explore possible outcomes.
- Handling Complex Dynamics: Model-based methods are well-suited for environments with complex dynamics, as they explicitly model the environment's transition probabilities.

Disadvantages:

- Model Inaccuracy: The accuracy of the learned model can impact the performance of model-based methods, especially in environments with stochastic or unknown dynamics.
- Computational Complexity: Planning and optimization in model-based methods can be computationally intensive, limiting their applicability in real-time or resource-constrained scenarios.
- Model Bias: Errors or biases in the learned model can propagate and impact the quality of the derived policies and value estimates.

4.3.4. Hybrid Methods (Actor-Critic Methods)

Hybrid Methods combine elements of both value-based and policy-based approaches. They learn both a value function and a policy simultaneously, often using separate components for estimation and improvement.
Examples of Hybrid Methods (including Actor-Critic Methods) include:

- Advantage Actor-Critic (A2C): An algorithm that maintains an actor (policy) and a critic (value function), where the actor updates the policy based on the advantage function derived from the critic's value estimates. [Mnih et al., 2016]
- Deep Deterministic Policy Gradient (DDPG): A hybrid method that combines DQN with policy gradients, using a deterministic policy to learn a value function and an actor network to optimize the policy deterministically. [Lillicrap et al., 2019]

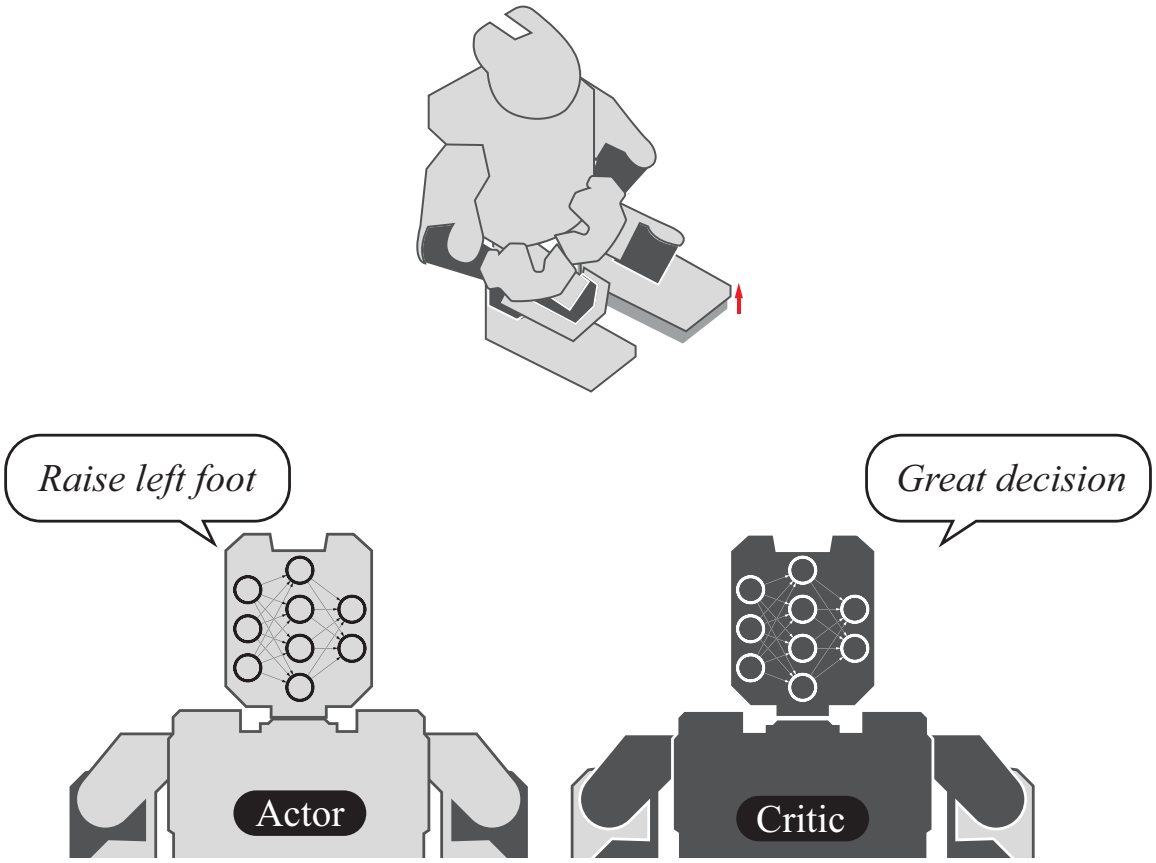


Figure 4-6.: Actor Critic Framework - Two independent entities (usually Neural Networks) are iteratively trained to decide how to act based on the state of the system (actor) and to evaluate the actions performed (critic).

Advantages:

- Combined Strengths: Hybrid methods combine the benefits of both value-based and policy-based approaches, leveraging value estimation for policy improvement.
- Improved Stability: Actor-critic methods often exhibit improved stability during training compared to pure policy gradient methods.
- Continuous and Discrete Actions: These methods can handle continuous and discrete action spaces, making them versatile for various problems.

Disadvantages:

- Complexity: Hybrid methods can be more complex to implement and tune due to the combination of value estimation and policy optimization components.
- Hyperparameter Sensitivity: These methods often have more hyperparameters to tune, increasing the difficulty of finding the right parameter settings.

- Exploration-Exploitation Tradeoff: Similar to value-based methods, hybrid methods face challenges balancing exploration and exploitation during learning.

4.3.5. Approach Based on Bézier Trajectories

In our proposed approach, we adopt a policy-based method incorporating critical elements discussed in Section 2.2. The approach, illustrated in Figure **4-7**, draws inspiration from the workflow for achieving dynamic gaits with the Atlas robot [Kuindersma et al., 2016] but with notable modifications. Instead of a floating-base-model approach, we employ a hybrid model approach to characterize and control the system dynamics. Additionally, we focus on a more straightforward locomotion task designed explicitly for flat terrain.

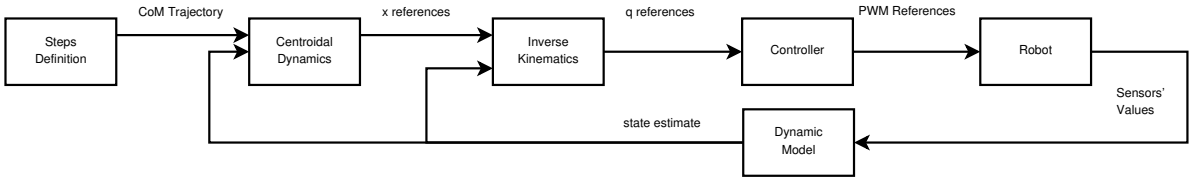


Figure 4-7.: Predefined gait control structure

Building upon the concepts presented in Section 2.2.2, we parameterize a nominal trajectory for the interest frames in the task space using Bézier polynomials, following the approach of hybrid zero dynamics. The boundary conditions of the Bézier trajectories are set to match the actual after-collision requirements, as seen in the work of [Arcos-Legarda et al., 2019], and we fix the pre-collision targets to soft collisions as mentioned in Section 2.1.5. Subsequently, we map these trajectories to the joint space using a damped pseudo-inverse inverse-kinematics approach, as mentioned in Section 2.2.1 [Wampler, 1986]. At the low-level tracking, we employ servomotor controllers. Finally, we leverage the encoder information and the forward kinematic equations of the robots to estimate the system's state and close the loop.
Based on Bézier trajectories, this approach combines elements from control models discussed earlier to achieve locomotion control. By utilizing Bézier polynomials and the hybrid-model approach, we can parameterize trajectories in the task space and map them to joint space for control execution. Integrating low-level servomotor controllers and state estimation further enhances the effectiveness of our approach.
To fully characterize the gait of the robots, we consider several decision variables that determine the step parameters, Bézier coefficients, and duration for each walking phase. Instead of directly treating the Bézier coefficients as decision variables, we leverage the known differentiability of the Bézier curves. This allows us to impose desired boundary conditions on the model's explored solutions, ensuring their feasibility and efficiency [Chevallereau et al., 2014]. The following decision variables are used to parameterize the Bézier curves in our model:

- Duration of single and double support phases (t^{SS} and t^{DS}, 2 variables).

- Length, width, and reference height for the steps (p_{BF}^{SS}, three variables).
- CoM position target (relative to the support foot) at the end of the single and double support phases (p_{CoM}^{SS} and p_{CoM}^{DS}, six variables).
- CoM velocity target at the end of the single and double support phases (v_{CoM}^{SS} and v_{CoM}^{DS}, six variables).
- CoM acceleration target at the end of the single and double support phases (a_{CoM}^{SS} and a_{CoM}^{DS}, six variables).

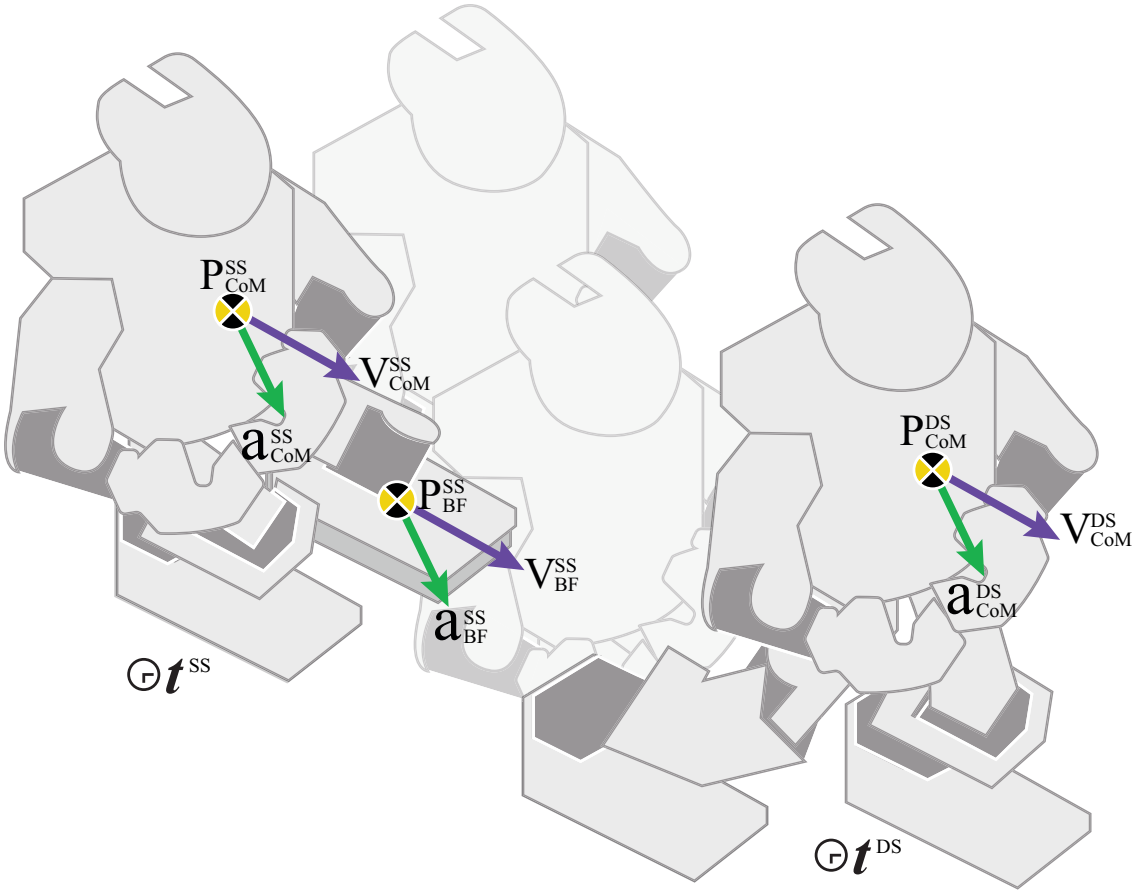


Figure 4-8.: Our Proposed Framework - Optimize the robot's performance by updating the Bézier Coefficients and phases duration that determine the gait imposed by kinematic control.

By incorporating these decision variables represented in Figure **4-8**, denoted as $\theta \in \Theta$, we establish a 23-dimensional constrained search space. It is important to note that to mitigate the impact forces when the foot lands at the end of each single support phase, we set the initial and final velocities and accelerations of the balancing foot to zero. This adjustment is

made to promote smoother foot transitions and enhance the overall stability of the robot's gait.

Advantages:

- Precise Dynamic Model: With the focus on minimizing the velocity of the feet at the collision instant and achieving soft collisions, the proposed method takes into account the specific dynamics of the system. By incorporating accurate modeling of the dynamics, the approach can better capture and represent the robot's behavior during locomotion, leading to more precise control and improved overall performance.

- Trajectory Parameterization and Mapping: Using Bézier polynomials to parameterize trajectories in the task space offers flexibility and control over the desired motion patterns. The method enables the generation of smooth and well-defined joint trajectories by mapping these trajectories to the joint space using a damped pseudo-inverse inverse-kinematics approach. This allows precise control of the robot's movements and enhances the ability to achieve desired locomotion behaviors.

Disadvantages:

- Complexity of Implementation: Implementing the proposed method based on Bézier trajectories can involve more than simpler techniques. Utilizing hybrid-model approaches and incorporating specific boundary conditions to match after-collision requirements may require additional effort in system modeling and control design. This complexity could lead to challenges in implementation, especially when adapting the method to different robot platforms or task scenarios.

- Limited Generalization: The approach's focus on precise modeling and control for minimizing foot velocity at collisions may limit its generalizability to other locomotion tasks or environments. The specific strategies employed may be highly tuned to the dynamics and requirements of the considered task on flat terrain, potentially leading to reduced performance in different scenarios. The method may require substantial modifications or adjustments to adapt to varying locomotion challenges, which could limit its versatility.

4.4. Optimization algorithm

Optimizing the parameters of reinforcement learning algorithms plays a critical role in achieving optimal performance for complex tasks. However, this process is not trivial due to several challenges, including the presence of local minima and the stochasticity of the system. The parameter tuning process in reinforcement learning is intricate due to multiple local minima and the influence of stochasticity. The non-convex nature of the parameter space presents

difficulties in finding the global optimum. At the same time, the stochasticity of the system can result in noisy gradients, affecting the reliability of the optimization process. These challenges demand specialized techniques for efficient and effective parameter tuning.

4.4.1. Gradient-Based Methods

Gradient-based methods utilize the gradients of the objective function (or estimations of them) to optimize the parameters of the policy. Algorithms like stochastic gradient descent (SGD) [Robbins and Monro, 1951], Adaptive Moment Estimation (Adam)
[Kingma and Ba, 2015], Root Mean Square Propagation (RMSprop) [Zou et al., 2019], or Augmented Random Search (ARS) [Mania et al., 2018] update the parameters by descending along the steepest gradient direction. Gradient-based methods are advantageous when the objective function is differentiable, as they can effectively exploit gradient information for precise optimization control. They are computationally efficient, enabling faster convergence and better utilization of computational resources.

4.4.2. Augmented Random Search

In our approach, we employed the Augmented Random Search algorithm V1 (ARS-V1) to optimize the performance of the robot's walking patterns [Mania et al., 2018]. ARS-V1 takes random directions in the search space, evaluates their impact on the fitness function, and updates the policy parameters based on a weighted sum of the top-performing directions. This algorithm adjusts parameter changes according to the standard deviation of the observed rewards, allowing for longer steps in low-variation regions and shorter steps in high-variation regions. The update rule used is as follows:

$$\theta_{k+1} = \theta_k + \frac{\alpha}{b\sigma_R} \sum_{d=1}^{b} [f(\theta_k + \delta_d) - f(\theta_k - \delta_d)]\delta_d, \tag{4-5}$$

Here, θ_k represents the parameter vector at the k-th iteration, α is the base learning rate, σ_R denotes the standard deviation of observed rewards in the current iteration, f refers to the fitness function, and δ_d represents the random unitary vector of the interest direction.
Additionally, our approach addresses the constraints imposed on the solutions by incorporating restriction barrier penalization functions [Luenberger and Ye, 2021]. While the search space is continuous, certain restrictions must be fulfilled, such as step length and height greater than 0, step width exceeding foot width, and joint limits. More complex constraints involve the zero moment point (ZMP) remaining within each joint's support polygon and torque limits. Barrier functions are employed to heavily penalize violations of these restrictions in a quadratic manner, allowing the optimization process to discern the degree of violation for each restriction. The fitness function is defined as follows:

Table 4-1.: Comparison of Optimization Methods

Method	**Approach**	**Strengths**	**Weaknesses**	**Applicability**
SGD	Updates parameters by descending along the steepest gradient direction with a fixed learning rate based on the batch or mini-batch samples	Simple and easy to implement; Computationally efficient for large datasets	Requires careful tuning of the learning rate; Prone to getting stuck in local minima	Large-scale optimization problems in machine learning and deep learning
Adam	Adapts the learning rate based on estimates of first-order and second-order moments of gradients	Efficiently handles sparse gradients and noisy or non-stationary objective functions; Converges faster compared to traditional gradient descent methods	Requires careful tuning of hyperparameters; May converge to suboptimal solutions in certain cases	Training deep neural networks and various optimization tasks
RMSprop	Adjusts the learning rate based on a moving average of squared gradients	Adapts the learning rate dynamically based on gradient magnitudes; Effective in handling sparse gradients and non-stationary objectives	Sensitive to the choice of hyperparameters; May converge prematurely to suboptimal solutions in some cases	Optimization tasks involving deep neural networks, natural language processing, and machine learning
ARS	Approximates the gradient by exploring random directions in the parameter space	Suitable for optimizing non-differentiable or complex objective functions; Does not require explicit gradient calculations	May require a large number of samples to obtain accurate gradient approximations; Sensitive to noise in reward estimates	Reinforcement learning and optimization problems with expensive or non-differentiable objective functions

$$f(\theta) = r(\theta) - \sum_{j=1}^{N_r} M P_j(x), \tag{4-6}$$

In this equation, N_r represents the number of restrictions, $M \gg r(\theta)$ ensures a large constant, and $P_j(x)$ corresponds to the j-th restriction of the model.
The prototype for the barrier functions $P_j(x)$ is illustrated in Figure **4-9**, where the function heavily penalizes violations of the restrictions but has no effect if the restrictions are fulfilled.

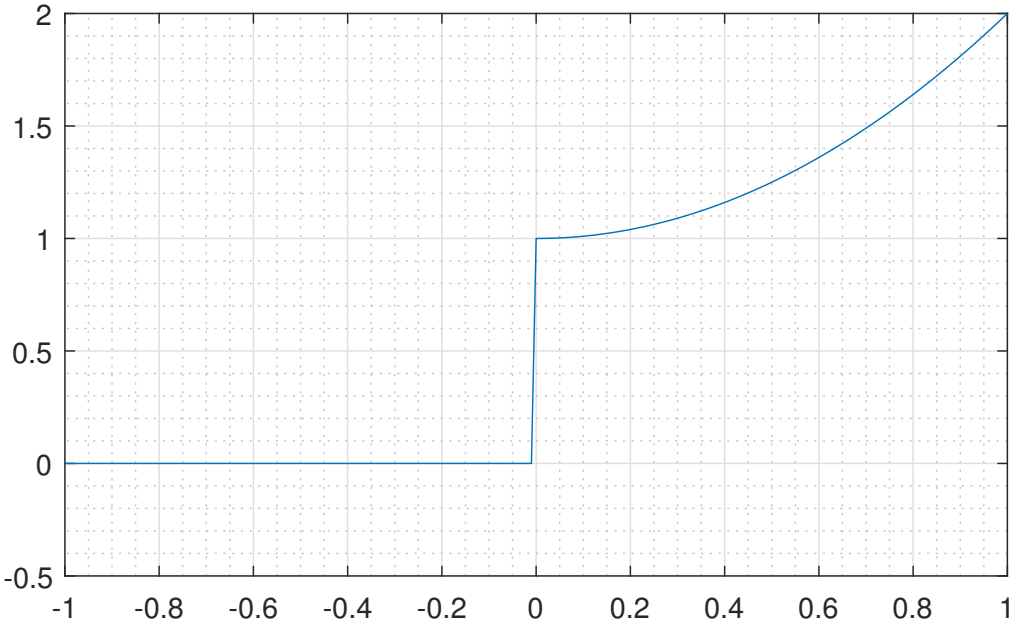
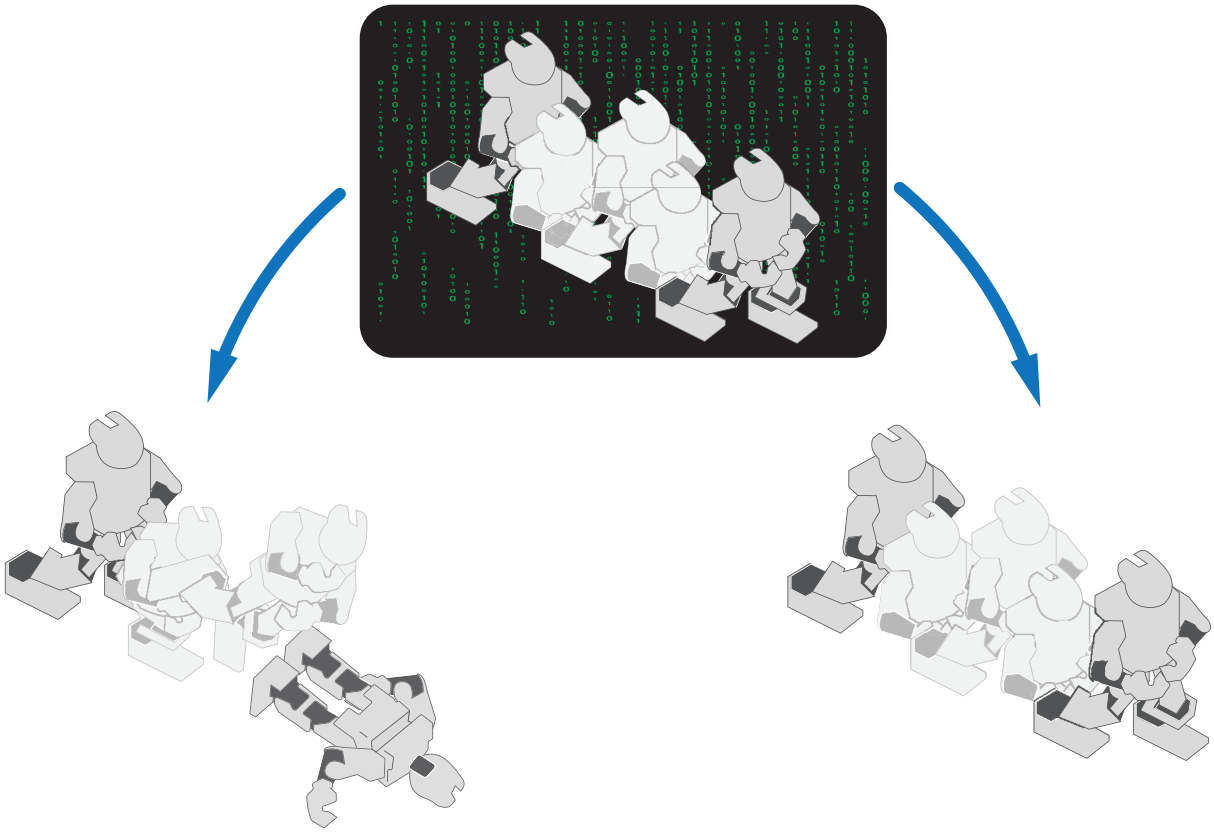


Figure 4-9.: Prototype for the barrier functions $P_j(x)$ - If the restriction is fulfilled, it does not affect the function, but it heavily penalizes the reward function otherwise.

4.5. The Reality Gap - Transferability

Reinforcement learning methods often rely on simulated environments for training due to the ease and efficiency of conducting experiments in virtual settings. However, there exists a significant challenge known as the Reality Gap, which refers to the differences between simulated environments and the real world (See Section 1.2.3) [Salvato et al., 2021]. These disparities can arise from inaccuracies in system characterization, unmodeled dynamics, or errors inherent in the simulation model, such as discretization. Notably, the calculation of impact forces between the robot and the environment, characterizing friction, and other non-linear behaviors remain challenging problems, leading to errors in the simulation of

robotic systems. Consequently, optimization results obtained in simulation may not directly transfer to real-world robots. Highly performing policies in simulation can become unfeasible or require subsequent fine-tuning to be effective in reality.



(a) (b)

Figure 4-10.: Reality Gap Problem - The simulated behaviors may fail when transferred to real environments (a) and usually require additional considerations to obtain an appropriate transferability (b).

Researchers have explored various strategies to mitigate the Reality Gap, each offering distinct approaches to address the challenges. This section discusses three common strategies: Domain Randomization, Gap Closing, and Mixing Virtual and Real Experiments. Additionally, we present our proposed approach for bridging the Reality Gap.

4.5.1. Domain Randomization

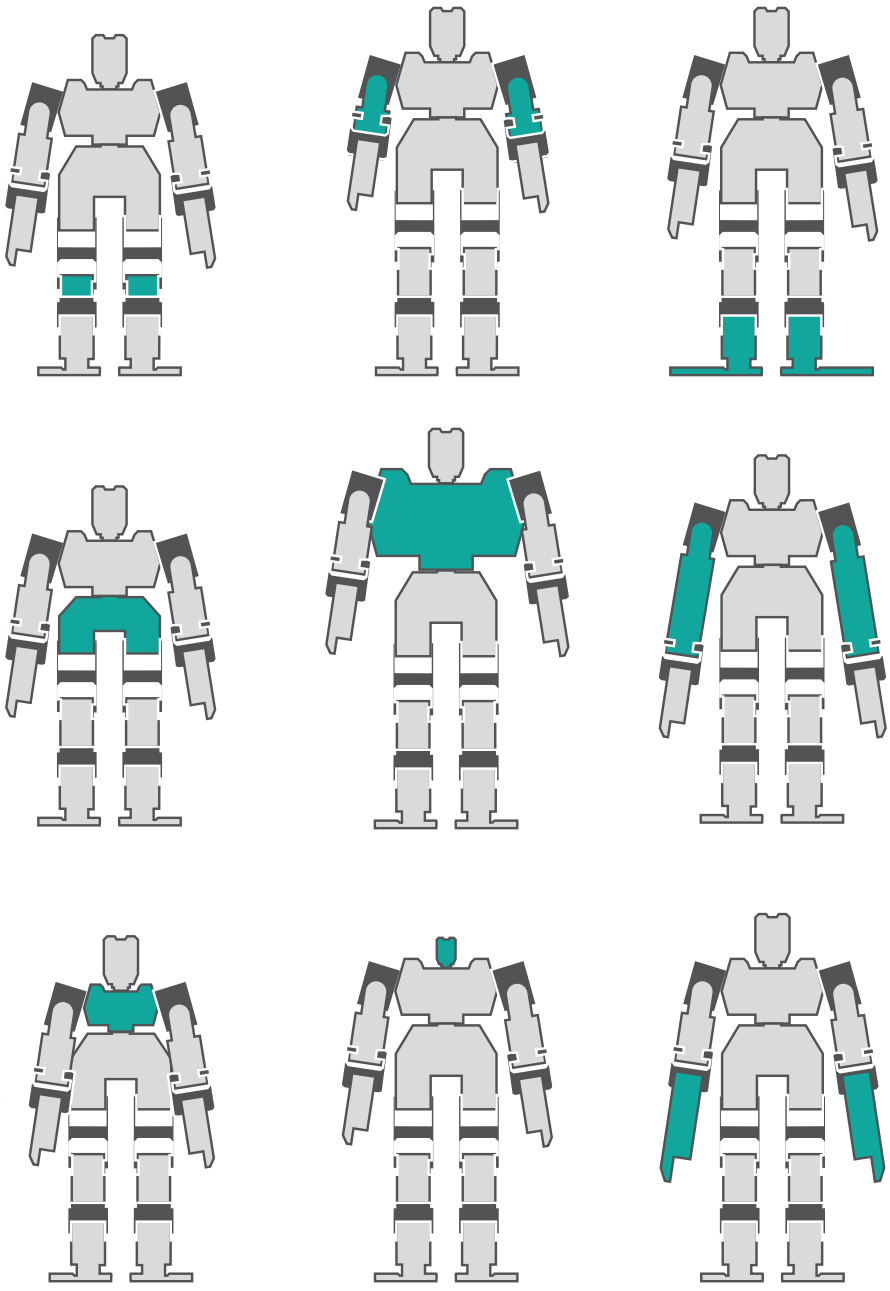


Figure 4-11.: Domain Randomization Approach - By training policies capable of dealing with a large set of conditions, the policy is expected to become capable of dealing with real conditions.

Domain randomization is an effective strategy to enhance the transferability of policies from simulation to reality. It involves introducing variations and randomization into the simulated environment during training. By randomizing factors such as object appearances, tex-

tures, lighting conditions, or physical properties, the policy learns to generalize across a wide range of conditions. This robustness to variations improves the likelihood of the policy performing well in real-world scenarios with different characteristics compared to the simulation (if the agent is capable of properly behaving in many different environments, the reality would be ¨just another environment¨ between its expertise). Domain randomization has been successfully applied in various robotic domains, including manipulation tasks [Tobin et al., 2017], locomotion control [Chebotar et al., 2018], and humanoid robotics for soccer [Liu et al., 2022].

4.5.2. Gap Closing

Gap closing focuses on reducing the disparities between the simulated environment and the real-world system to improve transferability. This approach involves refining the simulation model by incorporating accurate representations of the robot's dynamics, sensors, and actuators. Additionally, accounting for environmental factors such as friction, noise, or disturbances can further align the simulation with reality. By narrowing the gap through more precise modeling and simulation, the learned policies become more applicable to real-world scenarios. However, the challenge lies in accurately capturing all relevant dynamics and factors, which may require iterative refinement and calibration. Gap closing has been successfully applied in various robotic tasks such as quadruped locomotion [Tan et al., 2018], Autonomous driving for racing [Niu et al., 2022], and Manufacturing systems digital cloning [Müller et al., 2022].

4.5.3. Mixing Virtual and Real Experiments

Another strategy is to combine virtual and real experiments during the training process. This approach involves conducting a limited number of real-world experiments alongside simulation experiments. By comparing the performance and behavior of policies in both environments, researchers can assess the transferability and adjust the policies accordingly. The goal is to identify behaviors that closely align with the simulation while demonstrating effectiveness in the real world. This approach allows leveraging simulators' versatility and computational efficiency while incorporating real-world data to enhance policy adaptation. Mixing Virtual and Real Experiments has been successfully applied in various robotic tasks such as quadruped locomotion as [Koos et al., 2013], and biped walking as [Rodriguez et al., 2018] and [Xie et al., 2020].

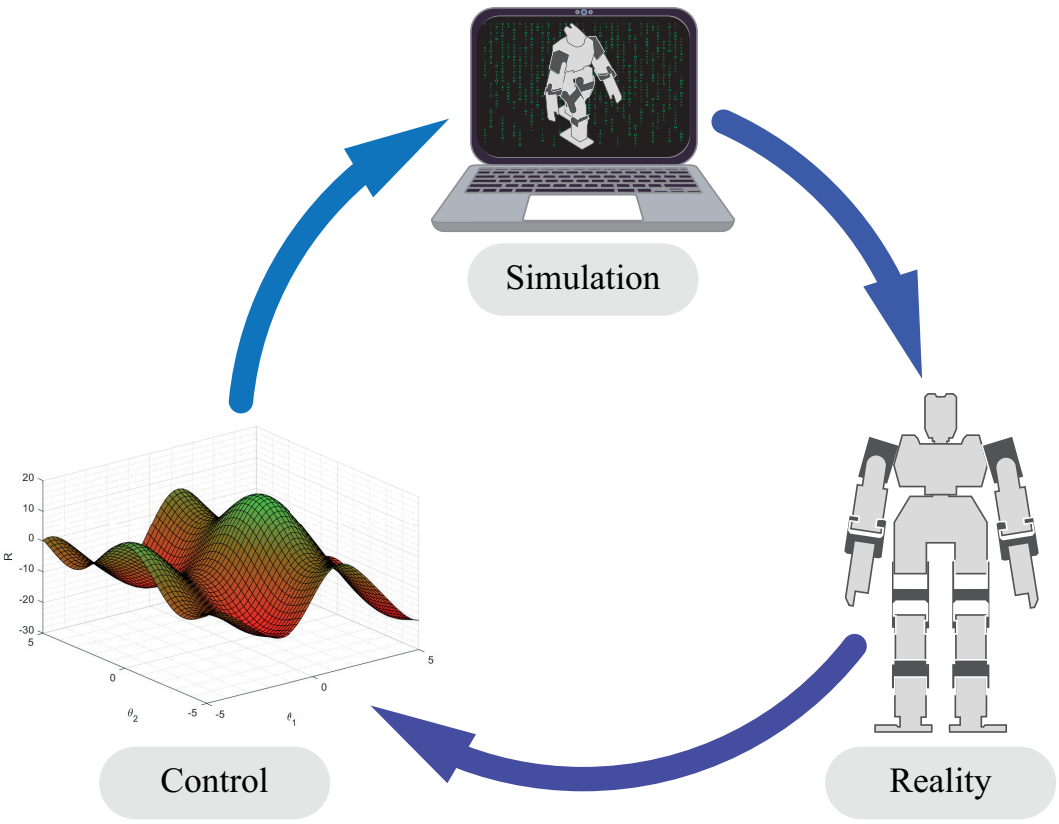


Figure 4-12.: Transferability Approach - By iteratively training policies both in simulation and in reality, the final policies are expected to work properly in both environments.

4.5.4. Proposed Approach

In our research, we propose a novel approach to address the Reality Gap in the context of reinforcement learning. Our approach focuses on three main elements: refined modeling, communication channel characterization, and hard-to-transfer dynamics avoidance.

- **Refined Modeling:** To bridge the Reality Gap, we enhance the accuracy of our simulation model through refined modeling techniques. We obtain the masses and inertia tensors of the robot using the Simscape Multibody tool in MATLAB's Simulink, based on the CAD files of its different parts and component data sheets. While most of the robot's rigid parts are considered in the model, we acknowledge that wires and connectors are not included. By incorporating these detailed representations, we aim to improve the fidelity of our simulation and reduce disparities between the simulated and real-world systems.

- **Characterization of the Communication Channel:** Another crucial aspect in

addressing the Reality Gap is accurately characterizing the communication channel between the robot and its control system. We identify the entire control loop's average time delay, incorporating this delay into both the simulations and the physical control protocol. By ensuring the precise representation and compensation of communication delays, we aim to minimize the discrepancies between simulation and reality, enabling a more effective transfer of learned policies.

- **Avoidance of Hard-to-Transfer Dynamics:** To further reduce the Reality Gap, we focus on avoiding complex and challenging dynamics that are difficult to transfer from simulation to the real world. Specifically, we employ trajectories with soft steps and proper support, avoiding flight phases and supporting foot tilting during bipedal walking (as described in Section 4.3.5). By narrowing the scope of our policies to the specific characteristics of bipedal walking and simplifying the models, we minimize errors and discrepancies, resulting in a smaller Reality Gap. This targeted approach allows for more accurate and reliable simulations, enhancing the transferability of learned policies to real-world robotic systems.

By incorporating these considerations - refined modeling, characterization of communication delays, and avoidance of hard-to-transfer dynamics - our approach enables the use of more accurate and reliable simulation models. This significantly reduces the disparity between the simulation and real-world environments, effectively transferring policies learned in simulation to real-world robotic systems. Our comprehensive approach aims to bridge the Reality Gap and facilitate the development of robust and transferable reinforcement learning policies for practical robotic applications.

5. Results and Modules Integration

The previous chapters have laid the foundation by introducing the theoretical background, control models, imitation learning, and reinforcement learning approaches. In this chapter, we focus on the practical aspect of our research, presenting the outcomes of extensive experimentation and integration of different modules. The main objective of this chapter is to evaluate the effectiveness and applicability of the proposed methods in achieving locomotion tasks. To accomplish this, we employ a combination of virtual and physical experimental setups to assess the performance and transferability of the developed techniques thoroughly.

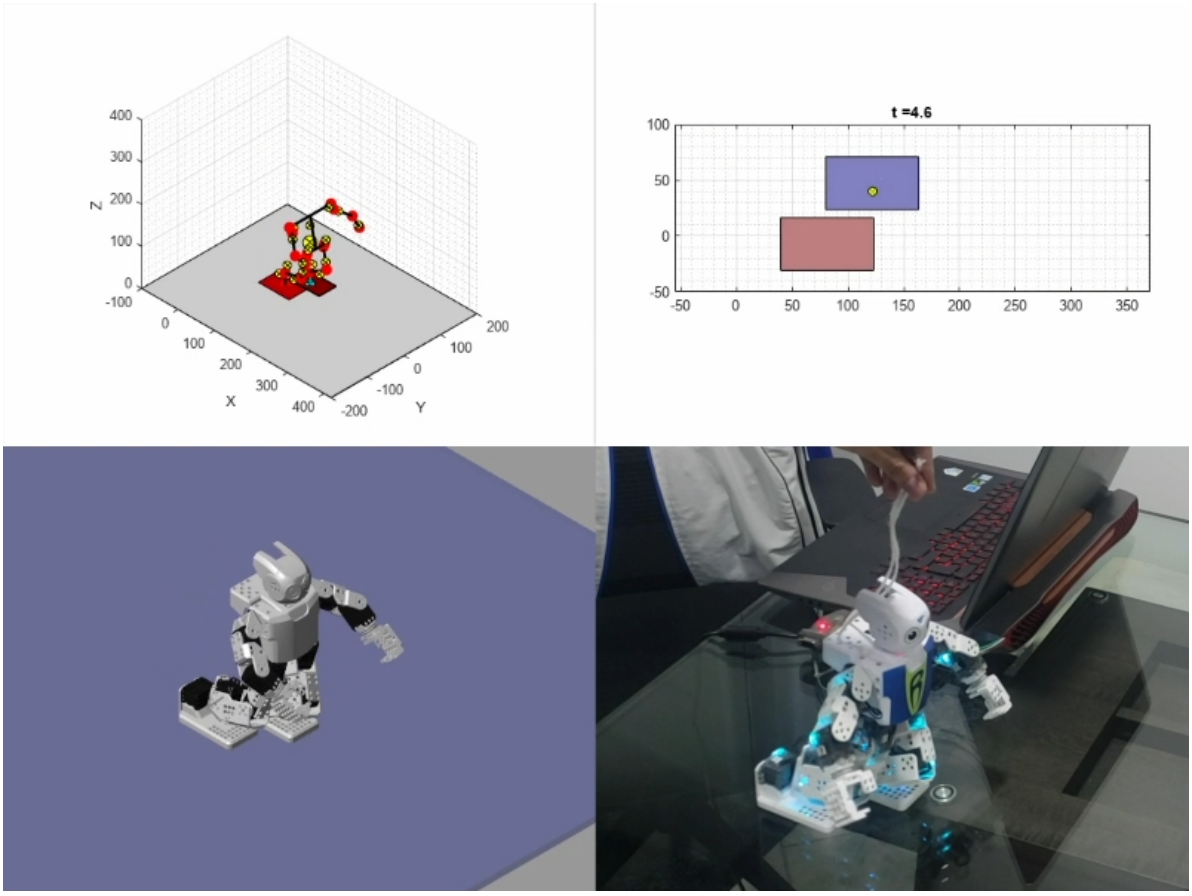


Figure 5-1.: Experimental Mosaic Example - custom simulator (top left), CoM and ZMP behavior (top right), Simulink Simulation (bottom left), and real test (bottom right)

We start by providing an overview of the experimental setups used in the evaluation process in section 5.1. This includes both the virtual experimental setup (Section 5.1.1), where simulations were conducted, and the physical experimental setup (Section 5.1.3), involving real-world experiments on the robotic platforms. These setups are designed to capture the key aspects and challenges associated with locomotion tasks, enabling a comprehensive evaluation of the proposed methods. The chapter then presents the results obtained from virtual and real environments in Section 5.2. We explore various trajectory generation methods, including handcrafted Bézier trajectories (Section 5.2.1), imitation-based trajectories (Section 5.2.3), and reinforcement learning-based trajectories (Section 5.2.2). Each subsection discusses the trajectory generation approach's performance, stability, and efficiency, shedding light on their capabilities and limitations in achieving the desired locomotion tasks within the simulated environment. Furthermore, we integrate reinforcement learning-based trajectories with the lessons learned from imitation learning (Section 5.2.4). By leveraging the expertise acquired through imitation learning as a starting point, we investigate the improvements achieved in the reinforcement learning process. These insights provide valuable guidance for effectively initializing reinforcement learning-based trajectories and enhancing performance. Through this exhaustive evaluation and integration of modules, we aim to provide a comprehensive understanding of the capabilities, limitations, and potential enhancements of the developed techniques. The outcomes presented in this chapter contribute to the overall objectives of this thesis, advancing the field of locomotion control and reinforcement learning in robotics.

5.1. Experimental Setups

To achieve optimized walking capabilities on real robots, this work combines the use of virtual experiments and physical experiments. The primary objective is to develop and validate effective locomotion strategies through a systematic approach. To accomplish this, we focus on a specific gait, which involves the robot starting from an arbitrary position with null velocities and accelerations, walking in a straight line towards a predefined goal, following a trajectory imposed by a parametric policy, and eventually coming to a stop at the target destination.

The experimental evaluations are conducted in both virtual and real environments to assess the performance and transferability of the developed methods. The virtual experiments begin in our custom simulator, which provides a flexible and efficient platform for most of the research process. This simulator allows for the training and evaluation of various methods, facilitating rapid iterations and exploring different parameter settings. To validate the virtual results and ensure their accuracy, a more robust simulator in Matlab's Simscape Multibody is utilized, which provides a high-fidelity representation of the robot's dynamics.

It is important to note that the training of all methods is performed in the simulated environment. This allows for extensive exploration and optimization of policies without the

constraints and risks associated with physical experiments. The simulated training serves as a crucial foundation for developing effective locomotion strategies.
The structure of this section begins with an overview of the virtual experimental setups, where we present the configurations and simulation parameters used in the custom simulator. We then discuss the considerations for transferring the learned policies from the virtual environment to the real robot, including specific parameter identification and adjustment procedures. Finally, we detail the physical experimental setups, describing the hardware specifications, sensors, and actuators utilized for the real-world validation of the developed methods.

5.1.1. Virtual Setup

The virtual experiments were conducted using a desktop computer and involved two virtual robots, Darwin Mini and MAX-E2. We employed a custom simplified simulator that specifically captures the gait dynamics within the proposed model's scope.
Our simulator consists of four main components:

- **Continuous Kinematic Model:** The individual phases of the gait were evaluated using Euler integration with fixed steps. The support foot was considered a fixed root for the kinematic tree, enabling the calculation of precise kinematic relationships during each phase.

- **Torque Estimation:** To simulate the motion of the nominal robot, the corresponding torques were estimated based on the dynamic model. We utilized the Recursive Newton-Euler Algorithm [Siciliano and Khatib, 2016] to calculate the required torques for the simulated motion.

- **Discrete Transition Functions:** The simulator incorporates discrete transition functions that map the values of the robot's state variables at the end of each phase to the initial conditions of the next phase. This ensures a smooth and consistent transition between different phases of the gait.

- **Supervisor Functions:** Supervisor functions play a critical role in the simulator by determining when the phases should change, validating the model assumptions, and evaluating the reward function. If any of the model assumptions are violated, the experiment is terminated prematurely, with a penalty proportional to the remaining time of the current experiment.

Figure **5-2** showcases a sample frame of the walking process for both robots in the custom simulator. Additionally, demonstration videos of example gaits can be viewed at `https://youtu.be/O5SXMa4Wgoo` and `https://youtube.com/shorts/8LxjQaXd5zM`.

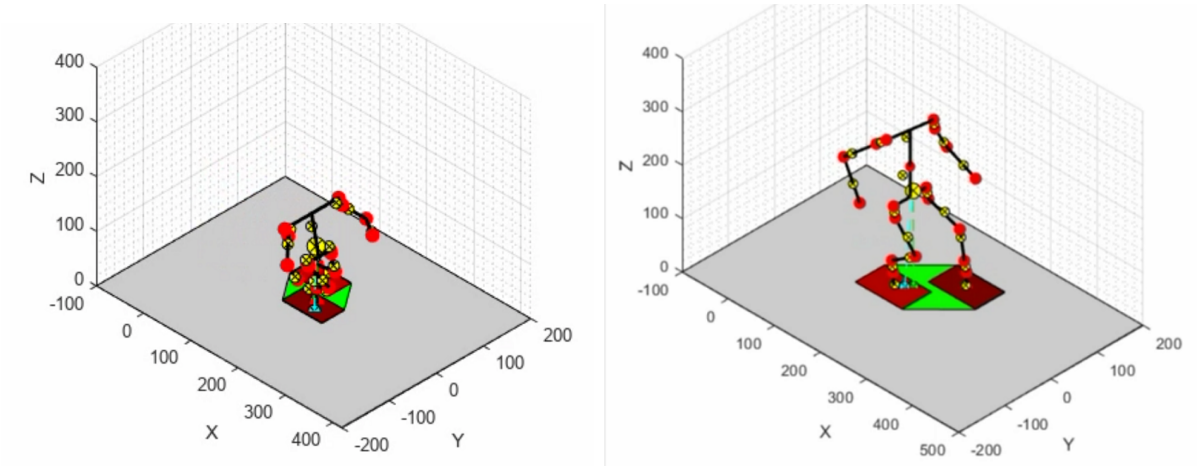


Figure 5-2.: Custom Simulator sample frames (Axis in milimeters) - (Left) Darwin Mini (Right) MAX-E2 - The red dots correspond to the position of the relevant frames of the robot, the small yellow dots correspond to the centers of mass of each link, and the larger yellow dot corresponds to the position of the entire robot CoM, the green dot corresponds to the projection of the CoM of the robot onto the ground, and the cyan dot corresponds to the ZMP.

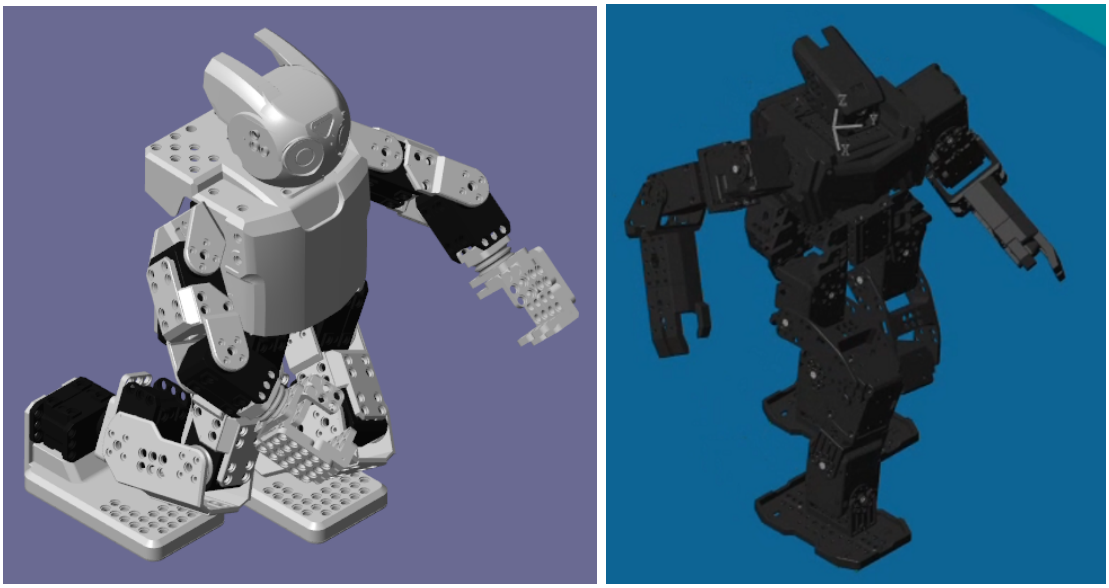


Figure 5-3.: Simscape Multibody Simulation sample frames - (a) Darwin Mini (b) MAX-E2

It is worth mentioning that the virtual experiments were initially performed using our custom simulator on a personal desktop computer. However, as mentioned in Section 2.3.2, we developed a detailed model of both robots in the MATLAB Simulink Simscape Multibody environment taking as references the CAD files of the links and servomotors available

online at [ROBOTIS, 2022a] and [ROBOTIS, 2022b]. Initial validation was carried out to ensure the accuracy and reliability of the virtual results before performing any test on the real robots. This alternative environment provides a more comprehensive and high-fidelity simulation environment, allowing us to assess the performance of the developed methods under more realistic dynamics and conditions at the cost of higher simulation times. Videos of those simulations can be found at `https://youtu.be/RvYAAQU-qEM` for Darwin Mini and `https://youtu.be/Faa3ncJLz_M` for MAX-E2.

5.1.2. Transferability Considerations

To ensure the transferability of our results, as discussed in Section 4.5.4, we implemented several strategies, including model refinement, characterization of communication delays, avoidance of hard-to-transfer behaviors, and using simple and robust controllers.
Regarding characterizing the communication channel, we conducted preliminary tests with both robots to determine the average delay time in the control loop. For Darwin Mini, the average delay time was approximately 2/33 seconds, while for MAX-E2, the average delay time was approximately 1/18 seconds. These delay times were considered in both the simulations and the physical control protocol to compensate for the delays in the references.
Regarding the sampling frequency, we identified a sampling frequency for reading and writing all the relevant registers of approximately 33 Hz seconds for Darwin Mini and approximately 180 Hz for MAX-E2. It is important to note that although the movement references for high-level control may have delays and comparatively slow sampling frequencies, the low-level controller directly influences the servomotors, providing a nearly instantaneous response time.
Furthermore, it is crucial to highlight that the communication parameters we obtained are specific to our setup and depend on various factors, such as the effective baud rate of the servomotor network, the specifications and configuration of the PC's USB port, and the processing time required to define a new action (reference) based on the current state.
By considering these factors and incorporating them into our experiments, we can use simplified models with minimal errors, resulting in a small gap that needs to be overcome between the simulated and real-world environments. This approach ensures that our results are transferable and applicable to real robotic systems.

5.1.3. Physical Setup

This section presents the experimental setups for the physical experiments conducted on two robots, Darwin Mini and MAX-E2. The detailed models of these robots are explained in Sections 2.3 and 2.4, respectively. Here, we will focus on the specific details of the experimental setup, including the network configurations, general characteristics of the environments, and considerations regarding friction and cables that were not explicitly considered in the

models.

Darwin-Mini

Figure 5-4.: Experiments setup for the physical experiments. The walking surface corresponds to a glass table, and the data wire (connected to Darwin's arm) is held next to the robot to avoid interfering with the gait.

For the processing of Darwin Mini, the robot is equipped with an internal OpenCM 9.04 microcontroller that acts as an intermediary between the main PC and the servomotor network. The kinematic control loop, state estimation, and trajectory planning are performed on a desktop computer running MATLAB, as the computational capabilities of the microcontroller are insufficient for the required computations. The communication between the desktop computer and Darwin Mini is established through a wired connection using the USB2Dynamixel communication device and the Dynamixel SDK for MATLAB. It is important to note that Darwin Mini is powered by onboard batteries, requiring only a data wire connected to its left arm free connector. (Note that the experiments performed required the battery charge to be high as otherwise, the control performance heavily deteriorates).

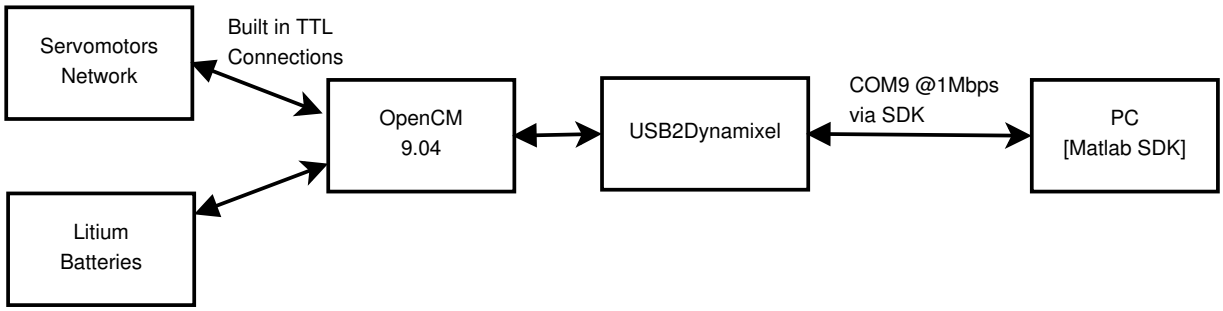


Figure 5-5.: Connections diagram for Darwin-Mini

With this configuration, we achieved an effective sample frequency of approximately 62 Hz for the entire control loop. The average delay in the control loop was measured as about four samples, corresponding to approximately 62.5 milliseconds. This delay accounts for the processing time required for the desktop computer to communicate with Darwin Mini and provide the necessary control signals. The communication diagram illustrating the required connections between the components is depicted in Figure **5-5**.

MAX-E2

For the processing of MAX-E2, the robot is equipped with an internal CM550 onboard computer and a Raspberry Pi Zero W. However, due to the limitations of the factory software, we opted to perform the kinematic control loop, state estimation, and trajectory planning processing on a desktop computer running MATLAB. The communication between the desktop computer and MAX-E2 was established through a wired connection using the U2D2 communication device and the Dynamixel SDK for MATLAB. It is important to note that MAX-E2 can be operated with batteries. Still, we utilized a wired power source for a more reliable and practical power supply, which required an additional wire in addition to the data wire.

With this configuration, we achieved an effective sample frequency of approximately 180 Hz for the entire control loop. The average delay in the control loop was measured as ten samples, corresponding to approximately 1/18 second. This delay accounts for the processing time required for the desktop computer to communicate with MAX-E2 and provide the necessary control signals. The communication diagram illustrating the required connections between the components is depicted in Figure **5-7**.

Figure 5-6.: Experiments setup for the physical experiments. The walking surface corresponds to a wood office desktop, and the power wire is held over the robot, avoiding interfering with the gait.

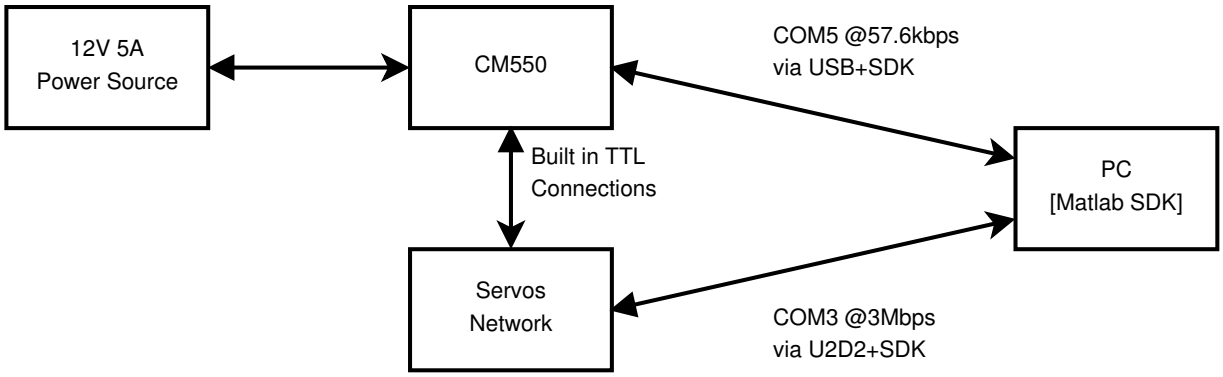


Figure 5-7.: Connections diagramMAXE

5.2. Results

To evaluate the performance of our proposed workflow, we conducted a series of experiments in both simulation and reality. These experiments aimed to verify the effectiveness of each module and the integration of all the modules together. We tested different approaches:

- Manually tunned Bézier trajectories (Section 5.2.1).
- Bézier trajectories obtained through imitation learning (Section 5.2.2).
- Bézier trajectories obtained through reinforcement learning from random initial conditions (Section 5.2.3).
- Bézier trajectories obtained through reinforcement learning from imitation learning initial conditions (Section 5.2.4).

Throughout these experiments, we observed a remarkable similarity between the results obtained in the simulation and the real-world environment. This section presents the detailed results and analyses of each experimental setup, highlighting the performance and effectiveness of our proposed workflow.

5.2.1. Manually Tunned Results

To establish a starting reference point for comparison, we initially implemented manually tuned Bézier trajectories. These trajectories were designed to generate stable and conservative gaits, allowing us to assess the performance of more advanced approaches. In both cases, the reference trajectory starts from an arbitrary position (that does not present singularities close to it) and has the following base parameters:

- The step width corresponds approximately to the original foot separation of the robot at a neutral position.

- The step length corresponds to approximately the foot length.
- The step height guide point corresponds to the height of the robot's ankle (given the polynomial nature, the actual height is way lower).
- The step duration corresponds to approximately 1 second, with 0.5 seconds for the simple support phase and 0.5 for the double support phase.
- The transition velocities and accelerations are set to 0.

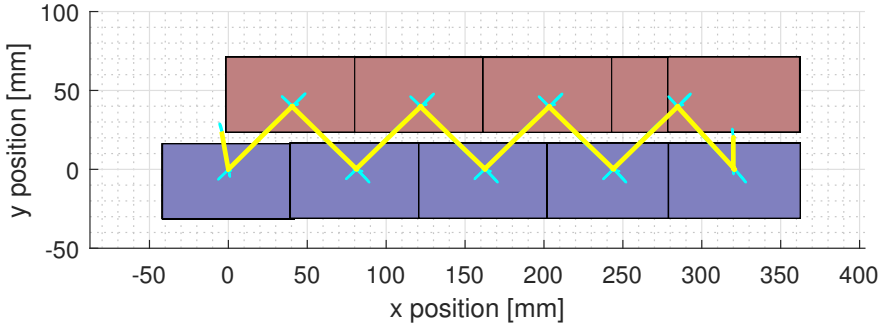
The corresponding parameters are summarized in Table **5-1**

Parameters	**Darwin Mini**	**MAX-E2**
Simple Support Time [s]	0.5	0.75
Double Support Time [s]	0.5	0.75
Step Size (X, Y, Z) [mm]	(40, 40, 20)	(45, 60, 40)
Double Support Target CM [mm]	(0, 0, 100)	(0, 0, 180)
Simple Support Target CM [mm]	(0, 0, 100)	(0, 0, 180)
Double Support Target VCM [mm/s]	(0, 0, 0)	(0, 0, 0)
Simple Support Target VCM [mm/s]	(0, 0, 0)	(0, 0, 0)
Double Support Target ACM [mm/s^2]	(0, 0, 0)	(0, 0, 0)
Simple Support Target ACM [mm/s^2]	(0, 0, 0)	(0, 0, 0)

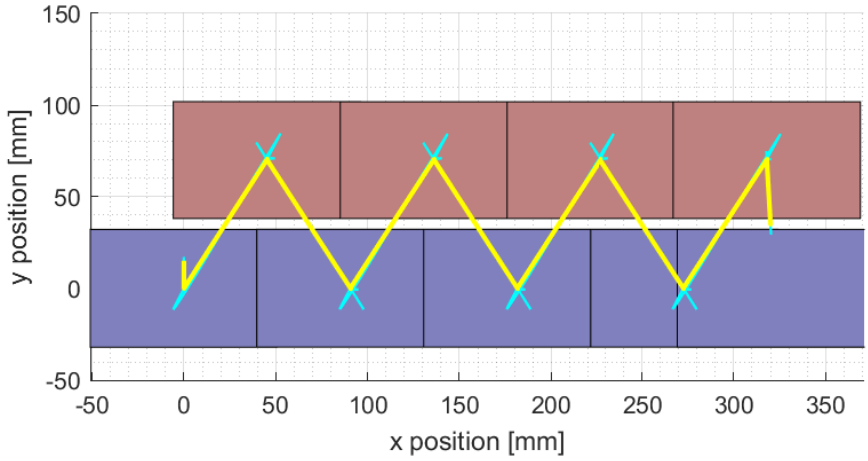
Table 5-1.: Parameters manually tuned for both robots

Figure **5-8** depicts the Zero Moment Point (ZMP) behavior relative to the support polygon for both robots. The resulting trajectory requires nine steps for both robots. Additionally, it is worth mentioning that the joint limits of Darwin restrict the step length as the sagittal hip servomotors cannot move freely behind the robot. Regarding the ZMP behavior, most of the time, it closely follows the CoM trajectory. Still, it deviates some millimeters when the robot accelerates without approaching the support polygon, evidencing the robust stability of the conservative gait.

Videos showcasing the manually tuned gait can be observed at `https://youtu.be/OcvwPjfmo2s` for Darwin-Mini and at `https://youtu.be/9lQOy1bWauU` for MAXE-2. The videos show the gait from 4 different perspectives: Our custom simulation, the Simulink Simscape Multibody simulation, the CoM/ZMP trajectory, and the test with the real robots.



(a)

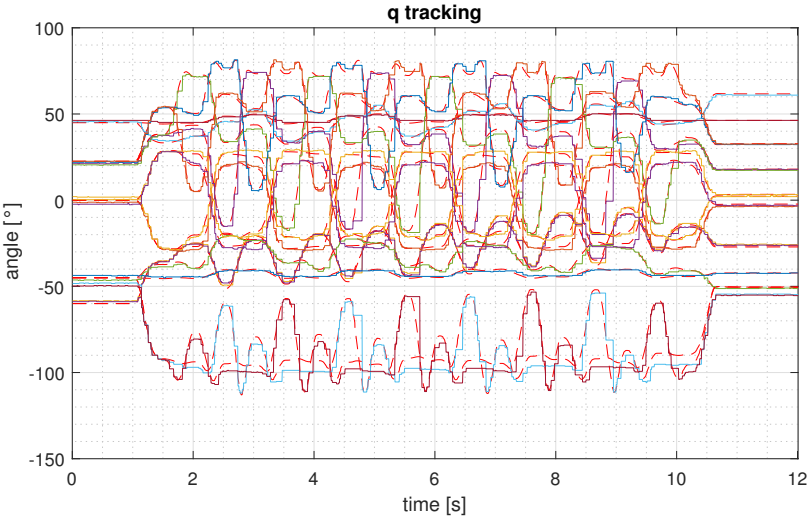


(b)

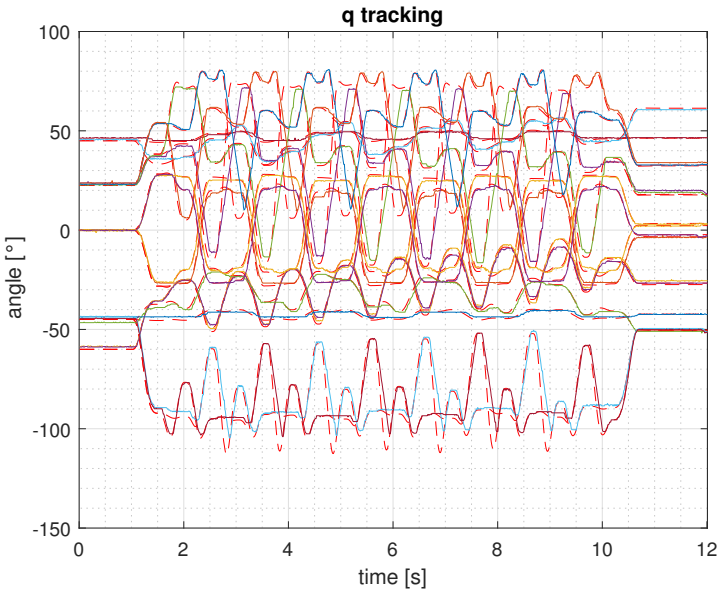
Figure 5-8.: ZMP behavior with manually tunned Bézier trajectories - (a) Darwin Mini (b) MAX-E2 - The yellow line corresponds to the CoM trajectory, the cyan line shows the ZMP trajectory (mostly overlapped with the CoM), the blue rectangle corresponds to the position of the right foot, and the red rectangle corresponds to the position of the left foot.

Figure **5-9** depicts the behavior of the robot in the joint space in reality compared to the desired behavior expected from the simulated environments. As expected, the behavior observed in the simulation in reality closely tracks the behavior expected from the simulation. Nonetheless, it is worth mentioning that in Darwin-Mini, some links' flexibility and the ser-

vomotors' backslash evidence qualitatively more significant errors than expected in some positions. Additionally, as the sampling frequency of Darwin Mini is slower, it is possible to note some edges corresponding to abrupt changes in its trajectories, even for the reference gait.



(a)



(b)

Figure 5-9.: Virtual and real joint behavior of the robots with manually tunned Bézier trajectories - (a) Darwin Mini (b) MAX-E2 - The color lines correspond to the real values of the joint trajectories, and the red dotted lines correspond to their corresponding references.

These manually tuned Bézier trajectories serve as a starting point for comparison with the results obtained through more advanced approaches, allowing us to assess the effectiveness and improvements achieved by the subsequent modules of our proposed workflow.

5.2.2. Imitation Learning Results

To illustrate the usability of our proposed approach for using the Motion Capture data for generating viable gaits for the robots. We considered a specific human recording as a reference. A video of the reference gait can be found at `https://youtu.be/EQn921aqoDc`, and a sample frame of this video is presented in Figure **5-10**. This specific gait was arbitrarily chosen as it presented walking in a straight line, similar to our interest case.

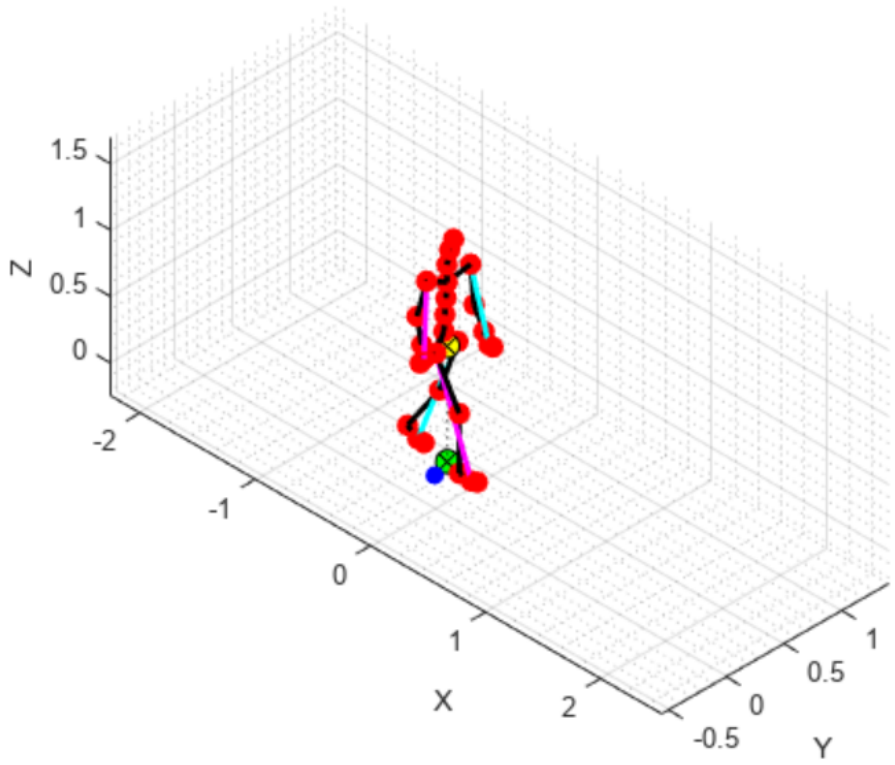


Figure 5-10.: Example frame of the reference MoCap gaits (Axis in meters).

The first step is to estimate the support modes from the data based on the velocity of the heel and metatarsus frames. Figure **5-11** shows the data and the support estimation for the interest gait.
With the estimation of the step phases, we identified 17 keyframes (12 phase changes and 5 balancing feet highest position frames). At these frames, we identified the position, velocity, and acceleration of the imitation vectors in the space of the human. However, we considered only the time and position information as the velocities and acceleration of the data presen-

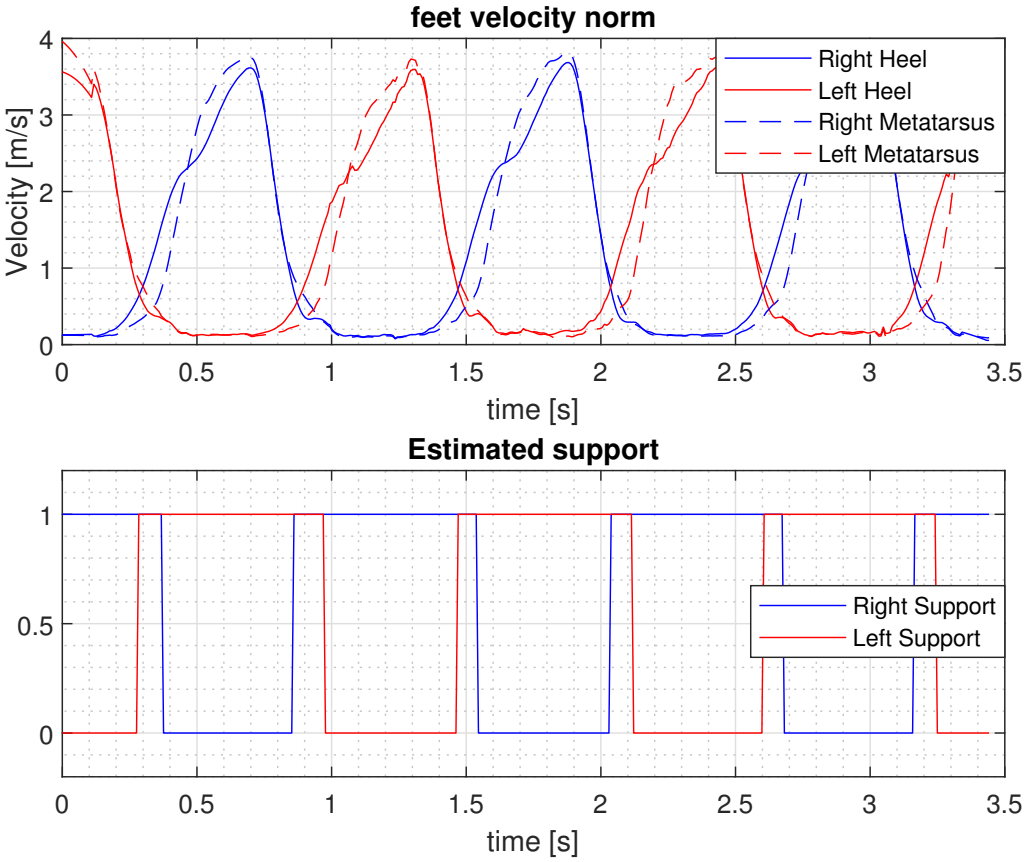
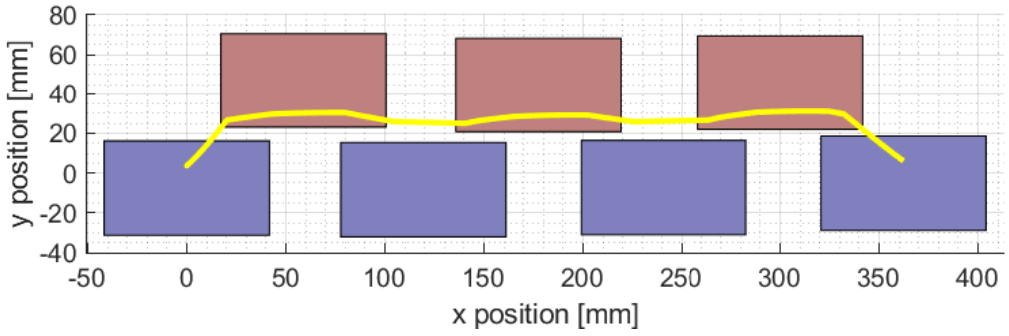
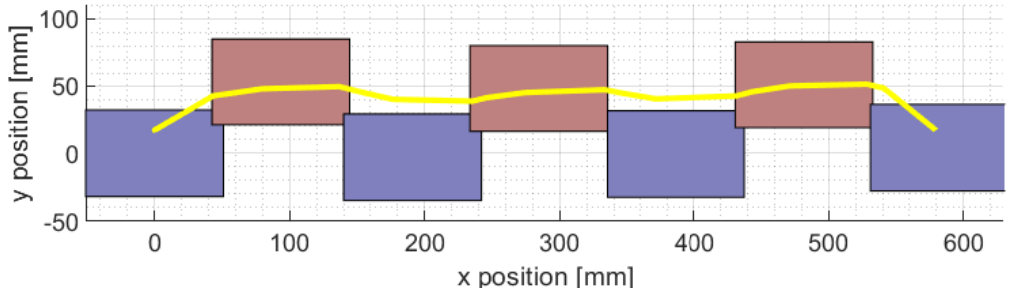


Figure 5-11.: Step phases estimation according to the heel and metatarsus frames velocities

ted very noisy behaviors that were not consistent when testing different filters for the data or slightly different choices for the keyframes. We mapped the positions from the human space towards the two robots' imitation spaces using equation 3-11 with their corresponding parameters. As a preliminary step, we performed a Bézier interpolation between the keyframes to observe the resulting behavior in a simulation with relaxed constraints allowing for feet superposition and escape of the ZMP from the support polygon. Figure **5-12** presents the two robots' footprints and CoM behavior during these simulations.



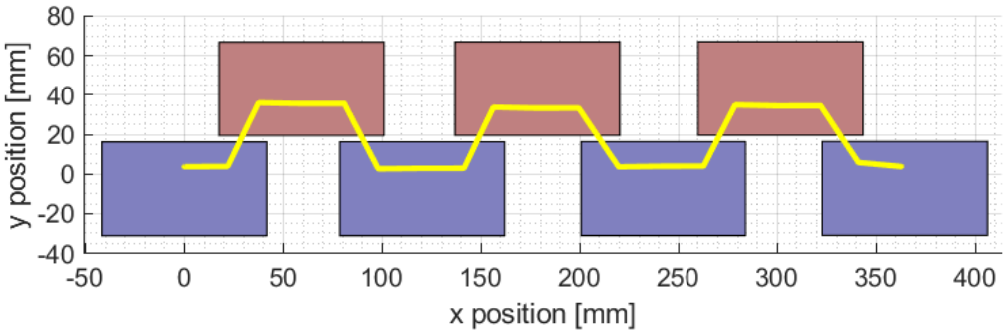
(a)



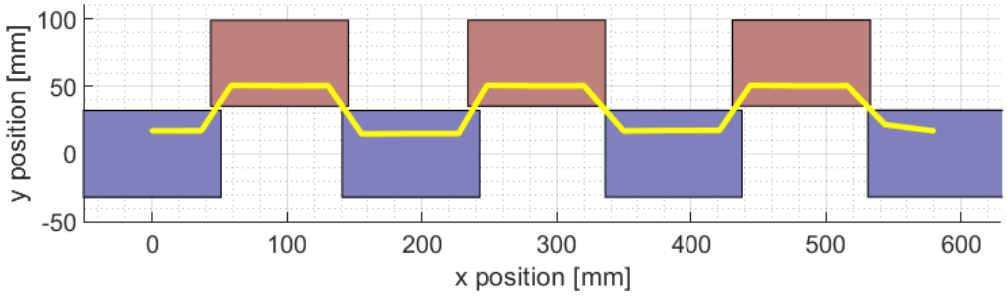
(b)

Figure 5-12.: Footprints and CoM behavior (before retargeting) - (a) Darwin-Mini (b) MAX-E2

For Darwin, the only apparent problem is the position of the CoM that lies towards the left causing the robot to fall when starting the simple support phases over the right foot. For MAX-E2, there are two evident problems. First, there is an overlap in the position of the feet, which would cause the robot to collide with itself and probably fall. And second, the trajectory of the Center of Mass (CoM) consistently falls over the left foot, which implies that when the robot gets into simple support phases over the right foot, it will fall. Additionally, for both robots, the Zero Moment Point (ZMP) behavior, though not depicted in the Figure due to its chaotic nature, frequently moves outside the support polygon, further highlighting the instability of the resulting gaits. We applied the proposed retargeting process to all keyframes to address the issues of the feet' superposition and the floor's perforation. This resulted in a new set of geometrically consistent keyframes, using an initial estimation of $d_{bound} = 20[mm]$ for Darwin-Mini and $d_{bound} = 15[mm]$ for MAX-E2 (We used a larger bound for Darwin-Mini because its low-level control performance is not as good as MAX-E2's control). The footprints and CoM behavior for the two robots during these simulations are shown in Figure **5-13**.



(a)

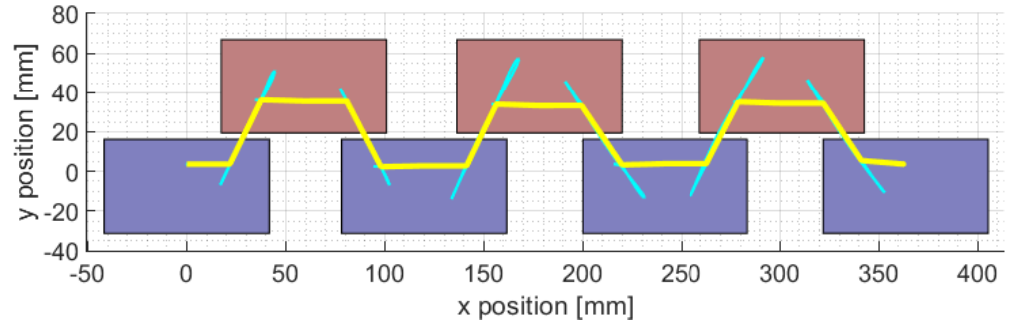


(b)

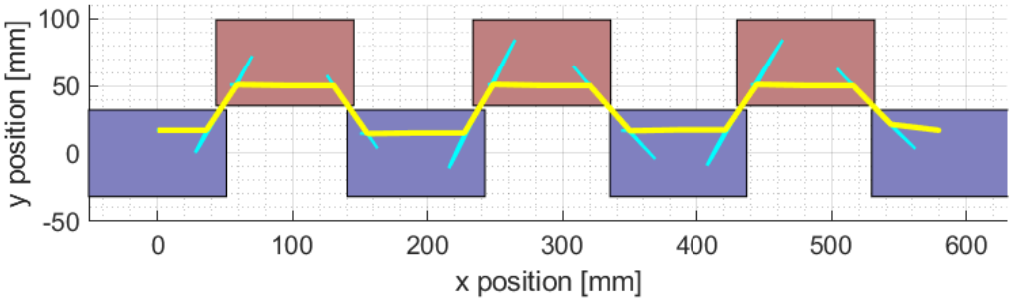
Figure 5-13.: Footprints and CoM behavior (after retargeting) - (a) Darwin-Mini (b) MAX-E2

In this case, the resulting gaits are geometrically consistent. However, the accelerations observed evidence that it is impossible to fix appropriate points within the support polygon for the CoMs as the maximum distances from the ZMP to the CoMs are way more extensive than the feet of both robots. We iteratively tested different slowdown factors until we found a value corresponding to a stable gait. The slowdown factor corresponded to 1/4 of the original velocity for Darwin-mini and 1/5 for MAX-E2. With these slowdown processes, we obtained feasible gaits that can be applied to the robots. The footprints, CoM, and ZMP behavior for the two robots during these gaits are shown in Figure **5-14**.

Videos showcasing the resulting gaits obtained through our proposed approach are available at `https://www.youtube.com/watch?v=5dr2heR-RZA` for Darwin-Mini and `https://www.youtube.com/watch?v=oNSWzcZsL_w` for MAX-E2. The videos show the gait from 4 different perspectives: Our custom simulation, the Simulink Simscape Multibody simulation, the CoM/ZMP trajectory, and the test with the real robots.



(a)



(b)

Figure 5-14.: Footprints, CoM and ZMP behavior (after retargeting and slowdown) - (a) Darwin-Mini (b) MAX-E2

Although it was required to slow down the gait to make it feasible for the robot by just using our IL approach, we summarize the corresponding Bézier gait parameters in Table **5-2** for using them as initialization parameters in Section 5.2.4.

Parameters	**Darwin Mini**	**MAX-E2**
t^{SS} [s]	0.57	0.57
t^{DS} [s]	0.22	0.22
p_{BF}^{SS} [mm]	(60.15, 35.58, 12.61)	(96.20, 97.12, 20.18)
p_{CoM}^{DS} [mm]	(-21.7, 3.7, 102.69)	(-36, 17, 173.45)
p_{CoM}^{SS} [mm]	(21.7, 3.7, 105.52)	(36, 17, 177.97)

Table 5-2.: Gait parameters obtained from Imitation Learning (without slow down)

5.2.3. Reinforcement Learning (from random initial conditions) Results

To evaluate the performance of our proposed approach based on Bézier trajectories (see Section 4.3.5), we conducted extensive experiments using reinforcement learning. The experiments consisted of 80 repetitions with the Darwin Mini robot and 80 repetitions of the algorithm with the MAX-E2 robot. In each repetition, the algorithm started from random initial conditions, initialized in a random neighborhood around the manually tunned gait (See Section 5.2.1). The algorithm hyperparameters were set as follows:

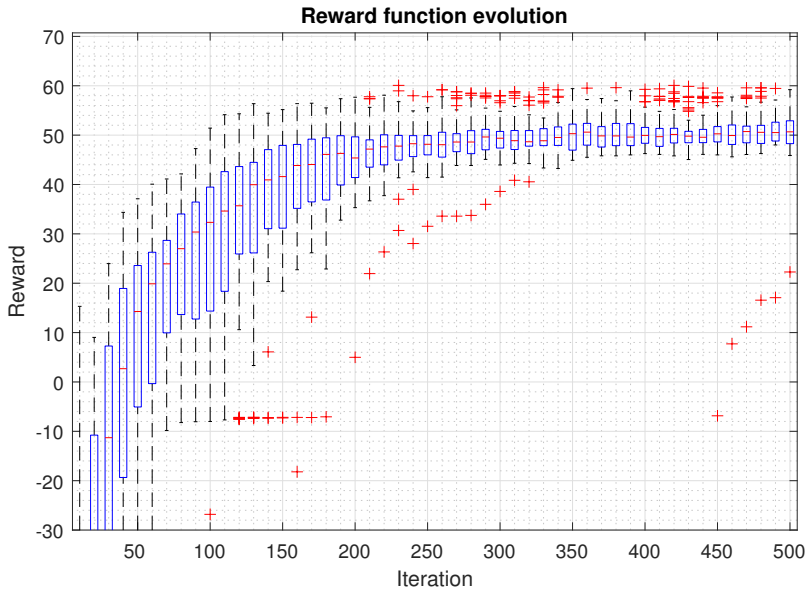
- Darwin Mini: $i_{max} = 500$, $\nu = 0{,}05$, $\alpha = 0{,}25$, $N_d = 9$, $b = 6$, and $M = 120$.

- MAX-E2: $i_{max} = 750$, $\nu = 0{,}05$, $\alpha = 0{,}5$, $N_d = 9$, $b = 6$, and $M = 180$.

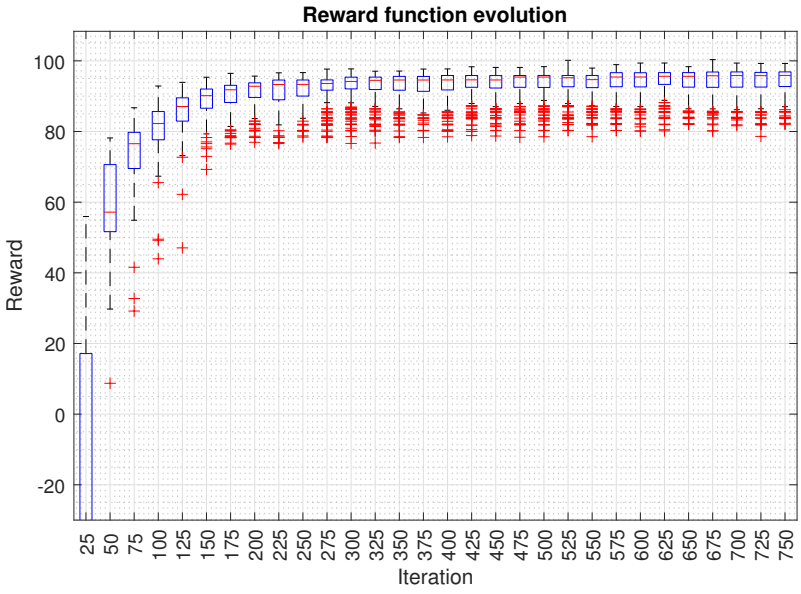
These hyperparameters were selected based on preliminary experiments, and although the algorithm is adaptive, hyperparameter optimization can improve the overall behavior. Nonetheless, further hyperparameter optimization was not formally applied beyond minor preliminary tests.

Figure **5-15** shows the progression of the reward function throughout the iterations of the algorithm for both robots. Notably, despite the initial variability in the rewards, a significant portion of the repetitions successfully converged toward the vicinity of the best solution found. However, some repetitions encountered challenges and got stuck in lower-performing local optima, underscoring the difficulty of optimizing the gait parameters effectively.

The experiments showed that the algorithm successfully discovered stable and efficient walking patterns for the Darwin Mini and MAX-E2 robots. The resulting Bézier trajectories generated by the reinforcement learning process exhibited smooth motion, and the ZMP remained consistently within the support polygon, ensuring the robots' stability during walking.



(a)



(b)

Figure 5-15.: Learning curves evolution - (a) Darwin-Mini (b) MAX-E2 - Concatenation of the boxplot diagrams for the reward at each iteration.

Videos showcasing the Reinforcement Learning optimized gait can be observed at `https://youtu.be/_bS11whLJG8` for Darwin-Mini and at `https://youtu.be/iORdiBx8xIg` for MAXE-2. The videos show the gait from 4 different perspectives: Our custom simulation, the Simulink Simscape Multibody simulation, the CoM/ZMP trajectory, and the test with the real robots.

Parameters	Darwin Mini	MAX-E2
t^{SS} [s]	0.47	0.57
t^{DS} [s]	0.32	0.24
p^{SS}_{BF} [mm]	(53.81, 60.71, 4.61)	(82.89, 71.64, 84.82)
p^{DS}_{CoM} [mm]	(-18.66, 8.74, 100.53)	(-29.87, 27.15, 178.62)
p^{SS}_{CoM} [mm]	(21.53, 5.25, 99.03)	(24.27, 20.10, 171.94)
v^{DS}_{CoM} [mm/s]	(6.00, -16.39, -3.49)	(3.55, -47.53, 29.67)
v^{SS}_{CoM} [mm/s]	(-7.29, -3.08, -8.24)	(28.17, 40.23, -1.14)
a^{DS}_{CoM} [mm/s^2]	(-4.29, 7.41, 14.53)	(-121.37, 93.80, 312.68)
a^{SS}_{CoM} [mm/s^2]	(7.30, 0.38, -15.17)	(138.69, -47.03, 247.20)

Table 5-3.: Best parameters found by the RL optimization (from random initial conditions) for both robots

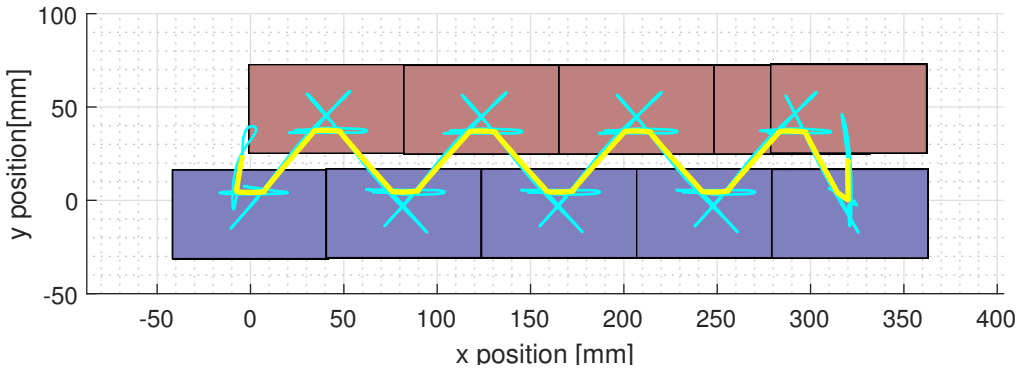
The best set of parameters found are summarized in Table 5.2.3. With these parameters, the robots achieve a maximum linear velocity of about 18 cm/s for Darwin Mini and about 20 cm/s for MAX-E2. From a qualitative perspective, the best policies obtained present some similar characteristics for both robots and between different repetitions:

- Steps were as narrow as allowed.
- CM trajectories with the tendency to be close to the internal edge of the feet with a certain margin.
- Double support phases are shorter than simple support phases.

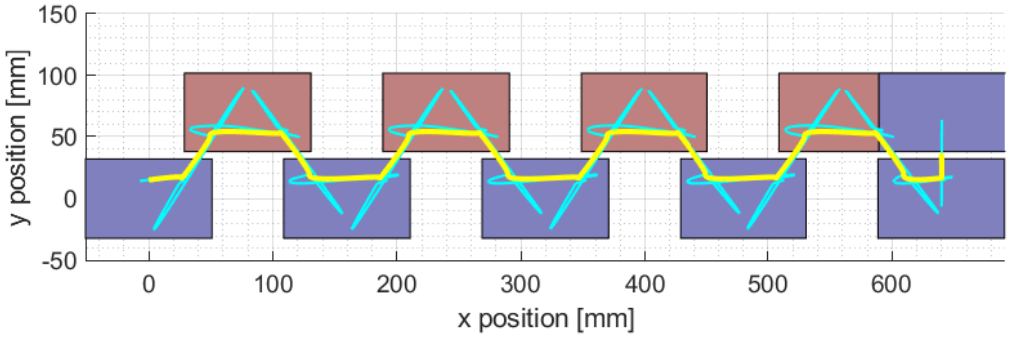
On the other hand, there are also some differences between the behaviors observed in Darwin-Mini and MAX-E2:

- In Darwin-Mini's trajectories, the balancing foot tends to be raised as minimum as possible, while MAX-E2 Trajectories show a higher step height.
- The main limitation for MAX-E2 to speed up is its stability. Under the current policy structure, the ZMP goes very close to the safety boundary (See Figure **5-17**). At the same time, Darwin-Mini still has a more significant margin of safety and is limited mainly for its joint limits, especially the hip sagittal joints that cannot take negative values.
- When performing tests on the real robots, the structure of Darwin presented more significant deformations that appeared ¨invisible¨ to its sensors, causing more instability than expected from the models.

- While MAX-E2 was capable of consistently showing its maximum velocity from simulation to reality thanks partially to its feedforward controller, Darwin's controller struggled to follow the references at high speeds limiting the maximum speed consistently achievable (See Figure **5-18**).

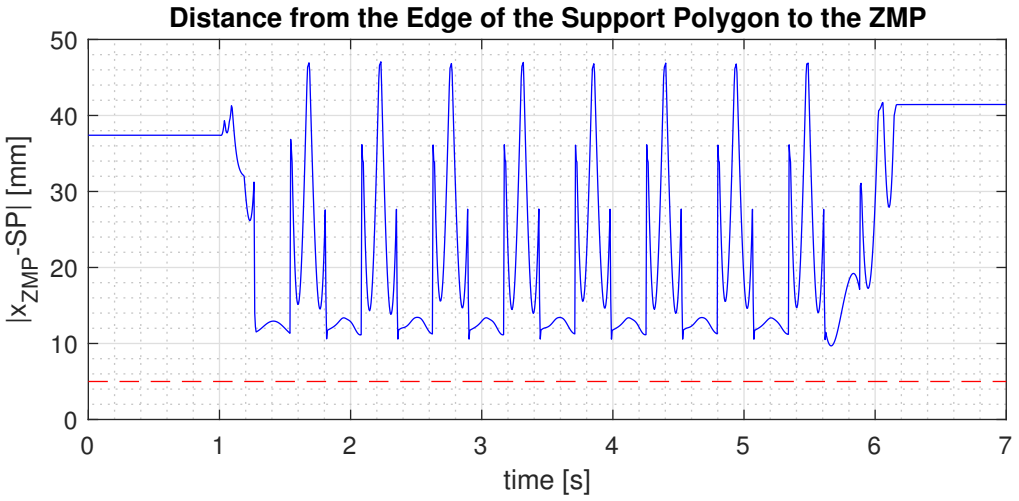


(a)

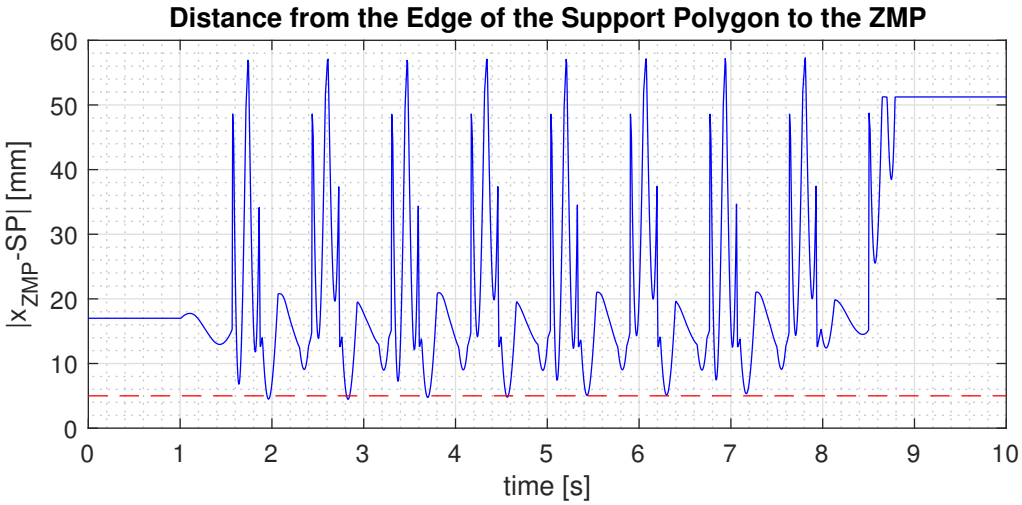


(b)

Figure 5-16.: ZMP behavior with Reinforcement Learning tunned Bézier trajectories - (a) Darwin Mini (b) MAX-E2 - The yellow line corresponds to the CoM trajectory, the cyan line shows the ZMP trajectory (mostly overlapped with the CoM), the blue rectangle corresponds to the position of the right foot, and the red rectangle corresponds to the position of the left foot.

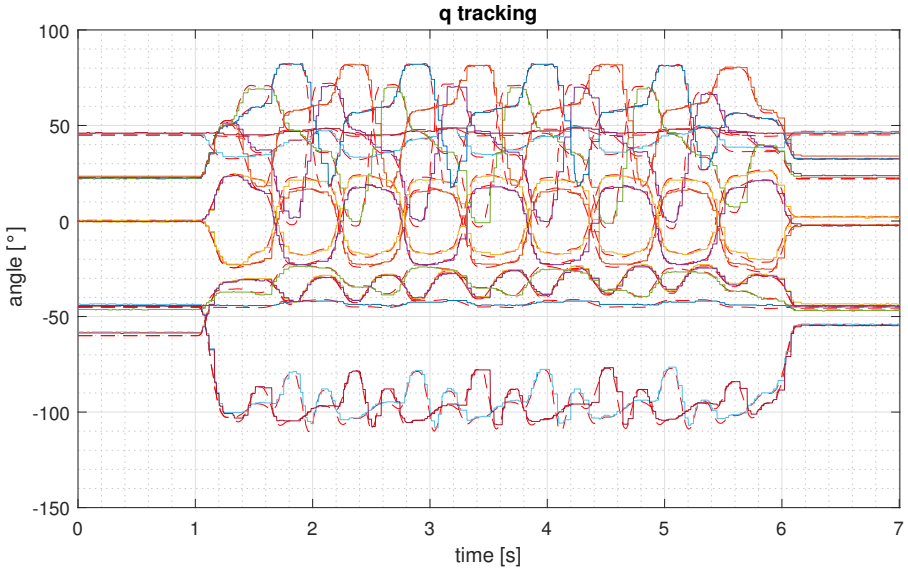


(a)

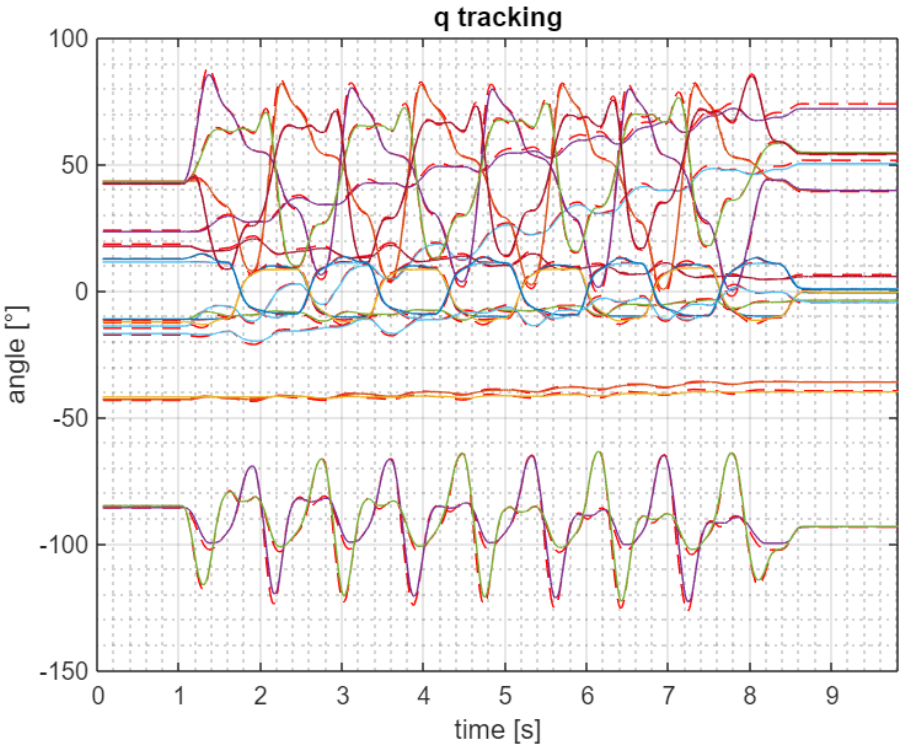


(b)

Figure 5-17.: Distance from the ZMP to the support polygon limit during gaits - (a) Darwin-Mini (b) MAX-E2 - the red dotted line corresponds to a safety boundary



(a)



(b)

Figure 5-18.: Virtual and real joint behavior of the robots with Reinforcement Learning tunned Bézier trajectories - (a) Darwin Mini (b) MAX-E2 - The color lines correspond to the real values of the joint trajectories and the red dotted lines correspond to their corresponding references.

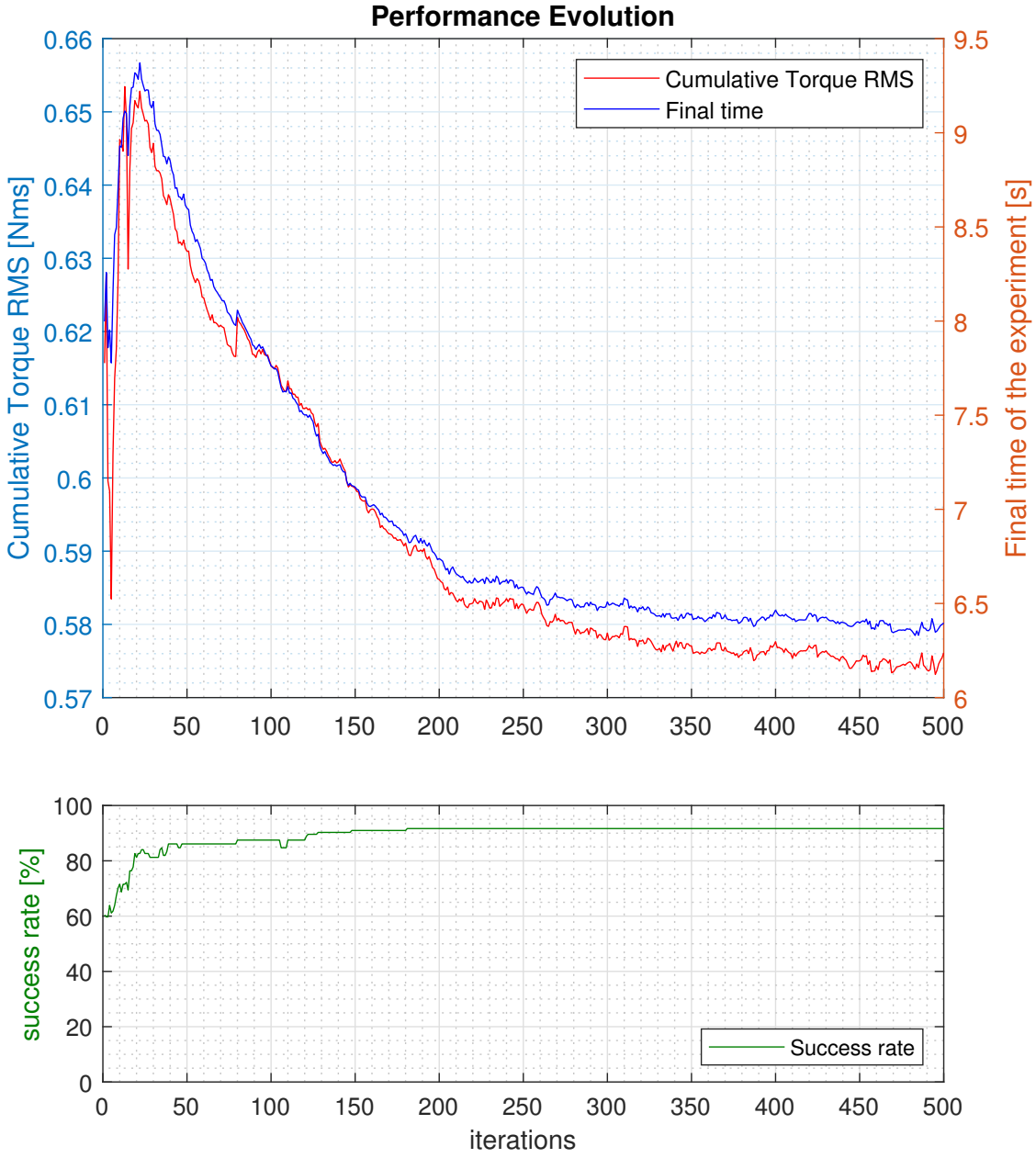


Figure 5-19.: Evolution of the mean cumulative RMS torque and experiment final time for Darwin Mini - At the start, the experiments finish as failure early. But, as the iterations pass, the success rate grows up to approximately 91 % and the final time and the cumulative torque get progressively reduced.

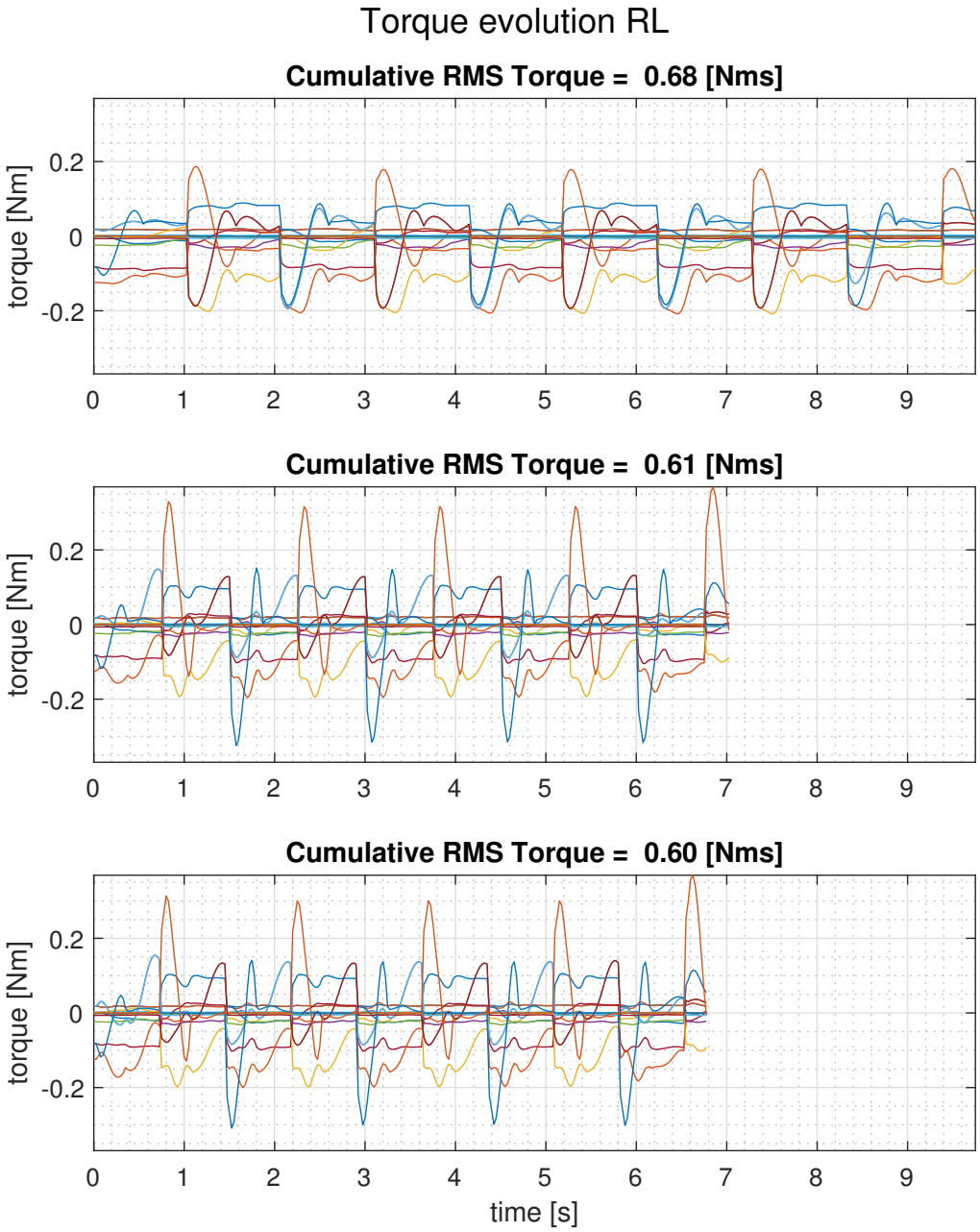


Figure 5-20.: Torque evolution through sampled iterations ($i = \{1, 50, 500\}$) for Darwin Mini - When starting near to the Handcrafted gait at the initial iterations, the cumulative RMS torque is comparatively large. As the iterations pass, the successful experiments get sharper and the load is distributed more evenly between different joints resulting in shorter successful gaits.

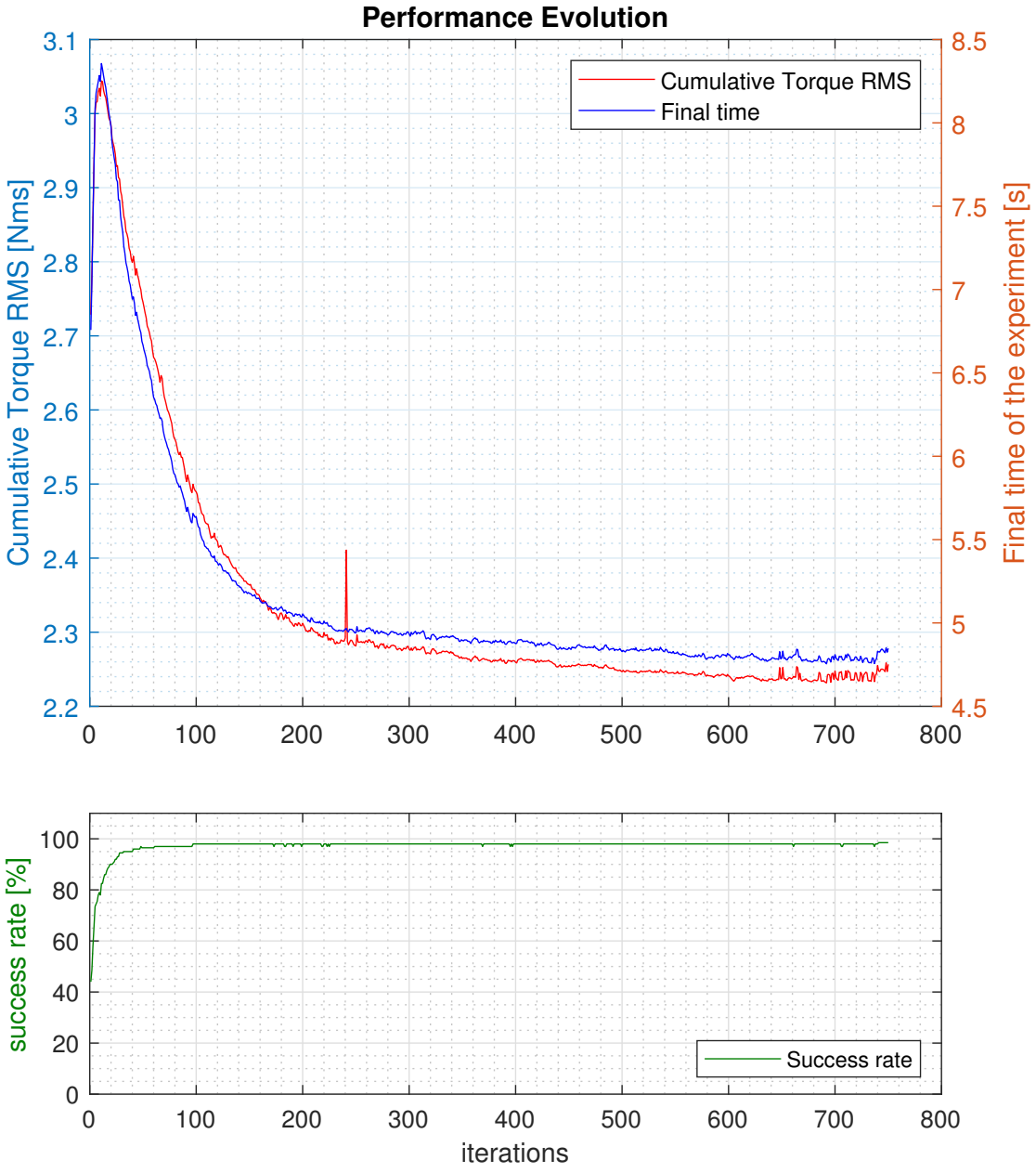


Figure 5-21.: Evolution of the mean cumulative RMS torque and experiment final time for MAX-E2 - At the start, the experiments finish as failure early. But, as the iterations pass, the success rate grows up to aproximately 98 % and the final time and the cumulative torque get progressively reduced.

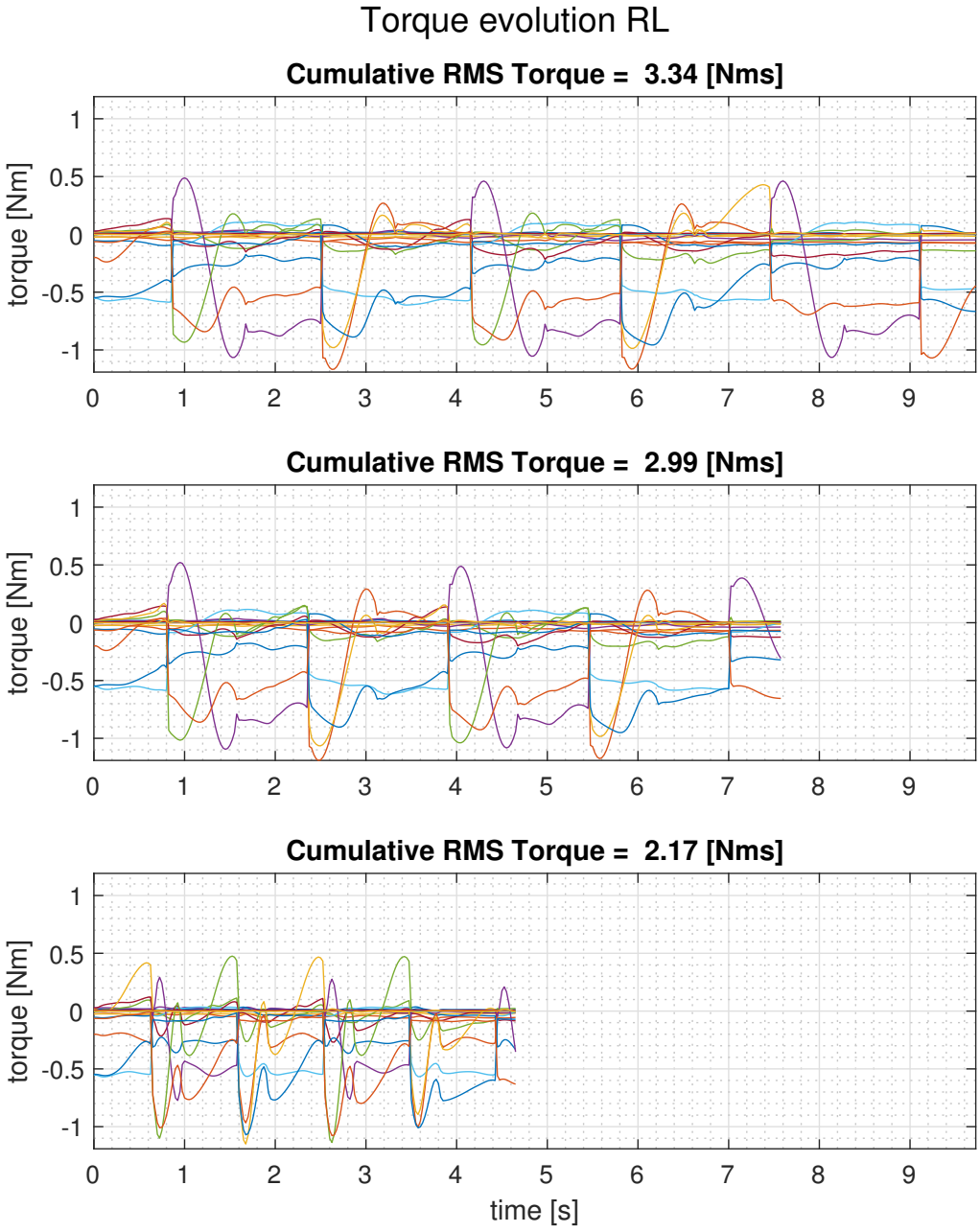


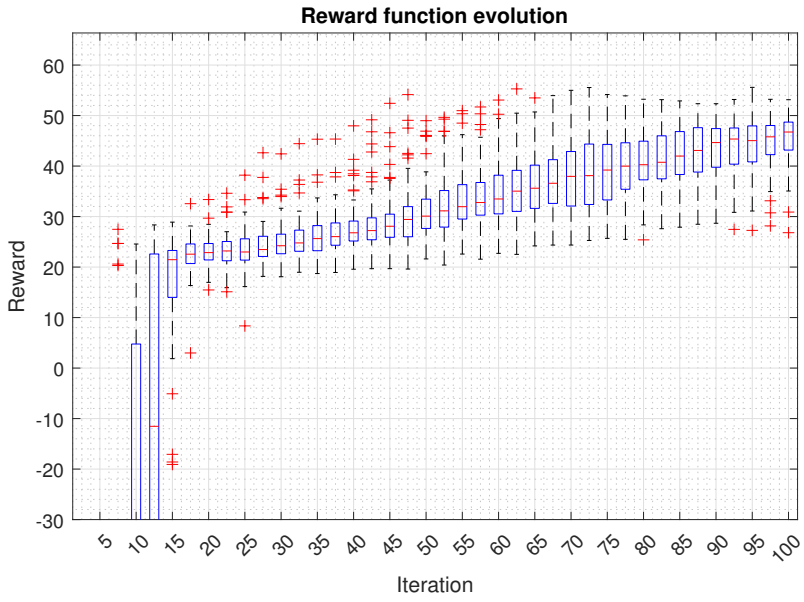
Figure 5-22.: Torque evolution through sampled iterations ($i = \{25, 50, 750\}$) for MAX-E2 - When starting near to the Handcrafted gait at the initial iterations, the cumulative RMS torque is comparatively large. As the iterations pass, the successful experiments get similar torques close to the performance obtained with RL from random conditions and present light improvements in velocity (faster finish times)

5.2.4. Reinforcement Learning (from Imitation Learning conditions) Results

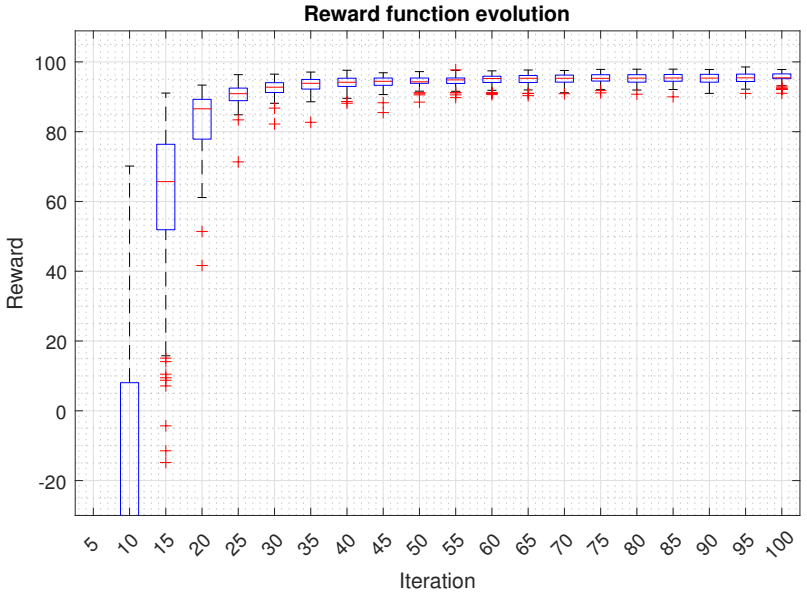
To further enhance the performance of our proposed approach, we explored the combination of Reinforcement Learning (RL) and Imitation Learning (IL). The goal was to leverage the strengths of both techniques to achieve more efficient convergence and better results for humanoid gait optimization. In the previous Section, the algorithm started from random initial conditions. However, this time, we initialized the RL process using the parameters obtained from the Imitation Learning phase (See Table **5-2**). These parameters served as a promising starting point, as they represented qualitatively similar walking patterns learned from human demonstrations, although they were not feasible by themselves.

Like the previous section, we conducted extensive experiments with almost the same hyper-parameters except for the iterations per repetition set to $i_{max} = 100$. For each robot (Darwin Mini and MAX-E2), we executed 80 repetitions using the combined Reinforcement Learning and Imitation Learning approaches. Figure **5-23** shows the progression of the reward function throughout the iterations of the algorithm for both robots. By initializing the RL process with parameters from IL, Darwin Mini presented a slight reduction in the number of iterations required to converge to stable and efficient walking gaits, while MAX-E2 showed a more significant improvement. While starting from random initial conditions (around the handcrafted gait) took several hundreds of iterations for convergence, starting from the IL parameters required only about hundred for Darwin-Mini and a few tens for MAX-E2. This substantial improvement in convergence speed highlights the complementary nature of RL and IL, where the imitation-guided initialization jumpstarts the RL optimization process.

In addition to the enhanced convergence speed and improved stability, the combination of Reinforcement Learning (RL) and Imitation Learning (IL) also exhibited a notable reduction in the variability of the results. In the pure RL experiments with randomly initialized gaits, we observed occasional failures where the optimization process struggled to reach the vicinity of the best solutions or sometimes failed to find feasible gaits altogether. This inherent variability in the outcomes highlighted the sensitivity of the RL approach to the initial conditions and the challenges associated with finding suitable solutions in a high-dimensional parameter space.



(a)



(b)

Figure 5-23.: Learning curves evolution - (a) Darwin-Mini (b) MAX-E2 - Concatenation of the boxplot diagrams for the reward at each iteration .

In contrast, the Imitation Learning phase provided a substantial advantage regarding result consistency. The RL process, initialized with parameters obtained from the IL phase, consistently reached excellent solutions, residing in the neighborhood of the best solutions with remarkably lower variability. Darwin-Mini presents a higher variability in its results than MAX-E2 in both scenarios (starting from random and from IL conditions) partially

because the joint limits interfere with the process more strongly than the stability constraints resulting in MAX-E2 being more adaptable to the different gaits. The imitation-guided initialization helped the RL algorithm overcome the pitfalls of random initialization and leverage the knowledge learned from human demonstrations. As a result, the combined approach demonstrated a more reliable and predictable performance, significantly reducing the risk of obtaining suboptimal or unstable gaits.

Videos showcasing the optimized gaits obtained can be observed at `https://www.youtube.com/watch?v=5dr2heR-RZA` for Darwin-Mini and at `https://www.youtube.com/watch?v=2wStOQBUX9w` for MAXE-2. The videos show the gait from 4 different perspectives: Our custom simulation, the Simulink Simscape Multibody simulation, the CoM/ZMP trajectory, and the test with the real robots. These gaits are very similar to the ones obtained with only RL, with the following main differences:

- The step height for MAX-E2 resulted in lower values when initializing from IL.

- The step width for Darwin Mini resulted in wider values when initializing from IL.

These differences may arise for reaching different local optima into the search space. However, the overall performances of both approaches were very similar. Note that the improvement for MAX-E2 was significantly better than for Darwin-Mini. On the other hand, it is worth to mention that initializing from IL improved the consistency of the result as the random initialization lead to approximately 9 % of failures for Darwin Mini and approximately 2 % of failures for MAX-E2 compared to the flawless success observed with IL initialization.

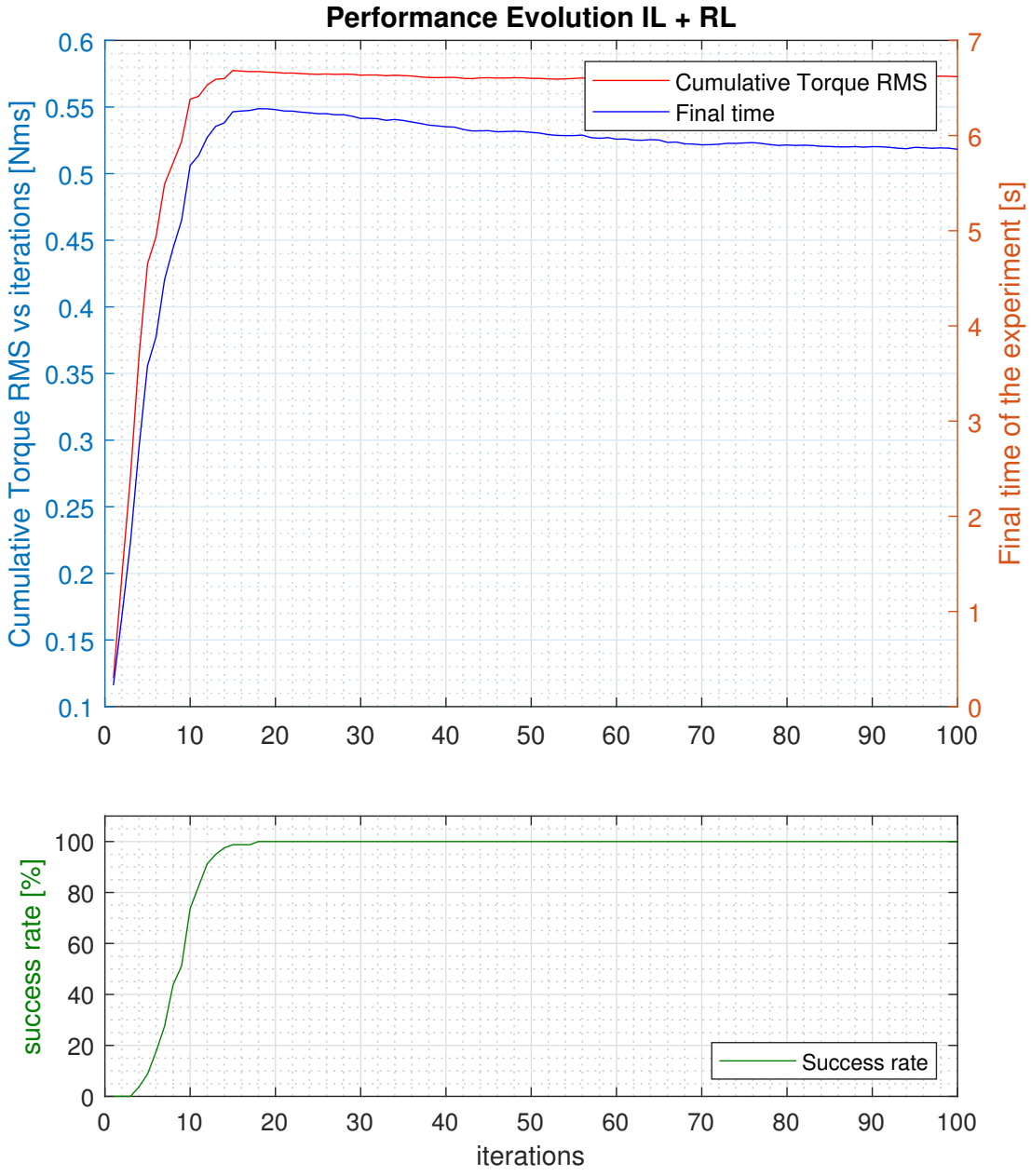


Figure 5-24.: Evolution of the torque and experiment final time through sampled iterations for Darwin Mini - At the start, the experiments finish as failure early. But, as the iterations pass, the success rate grows up to 100 %, the final time maintains stable, and the cumulative torque gets slightly reduced.

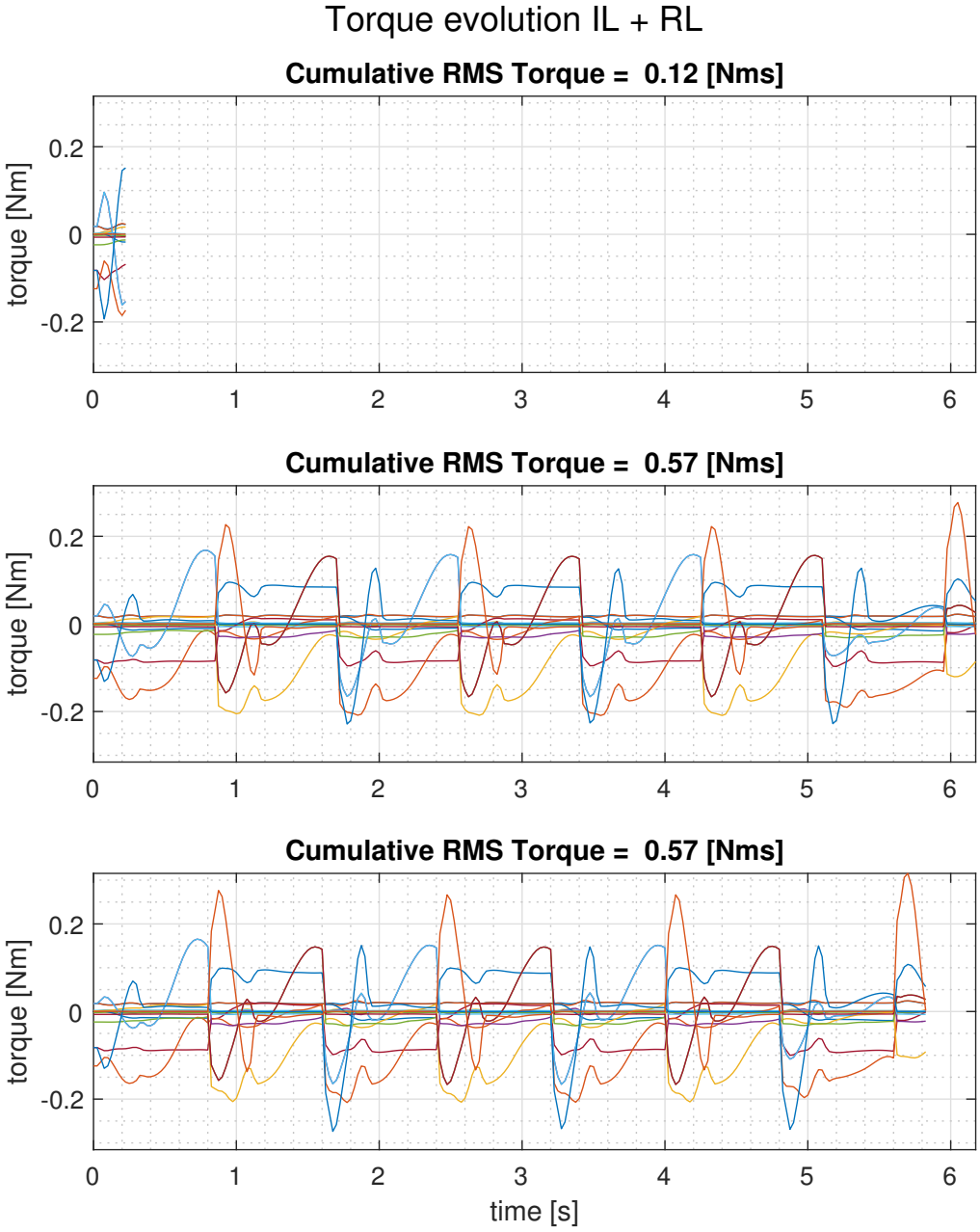


Figure 5-25.: Sample of torque evolution through sampled iterations for Darwin Mini - At the start, the cumulative RMS torque is low because the direct imitation policy is not feasible and results in early falls. As the iterations pass, the successful experiments get similar torques close to the performance obtained with RL from random conditions and present light improvements in velocity (faster finish times)

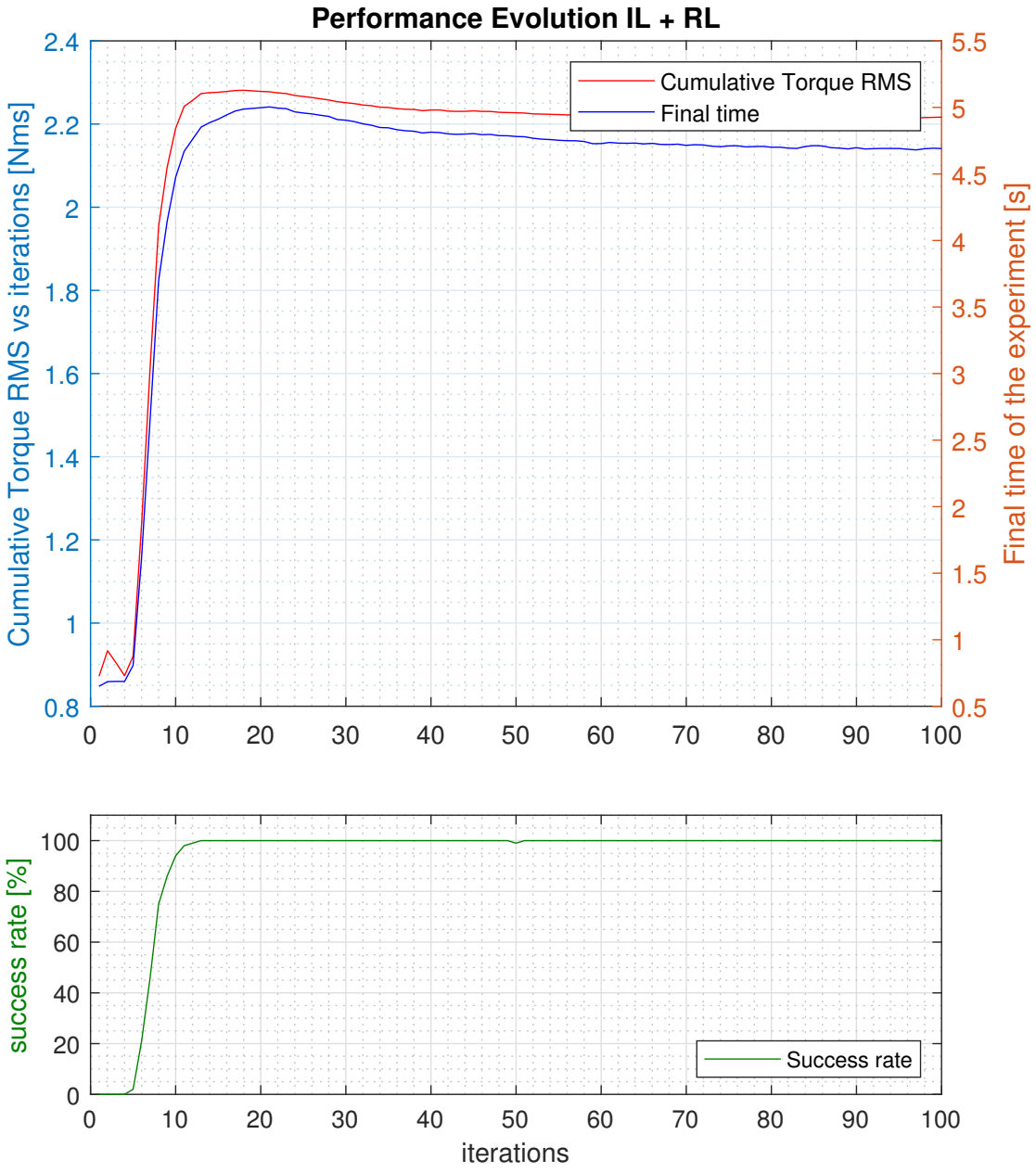


Figure 5-26.: Evolution of the torque and experiment final time through sampled iterations for MAX-E2 - At the start, the experiments finish as failure early. But, as the iterations pass, the success rate grows up to 100 %, the final time maintains stable, and the cumulative torque gets slightly reduced.

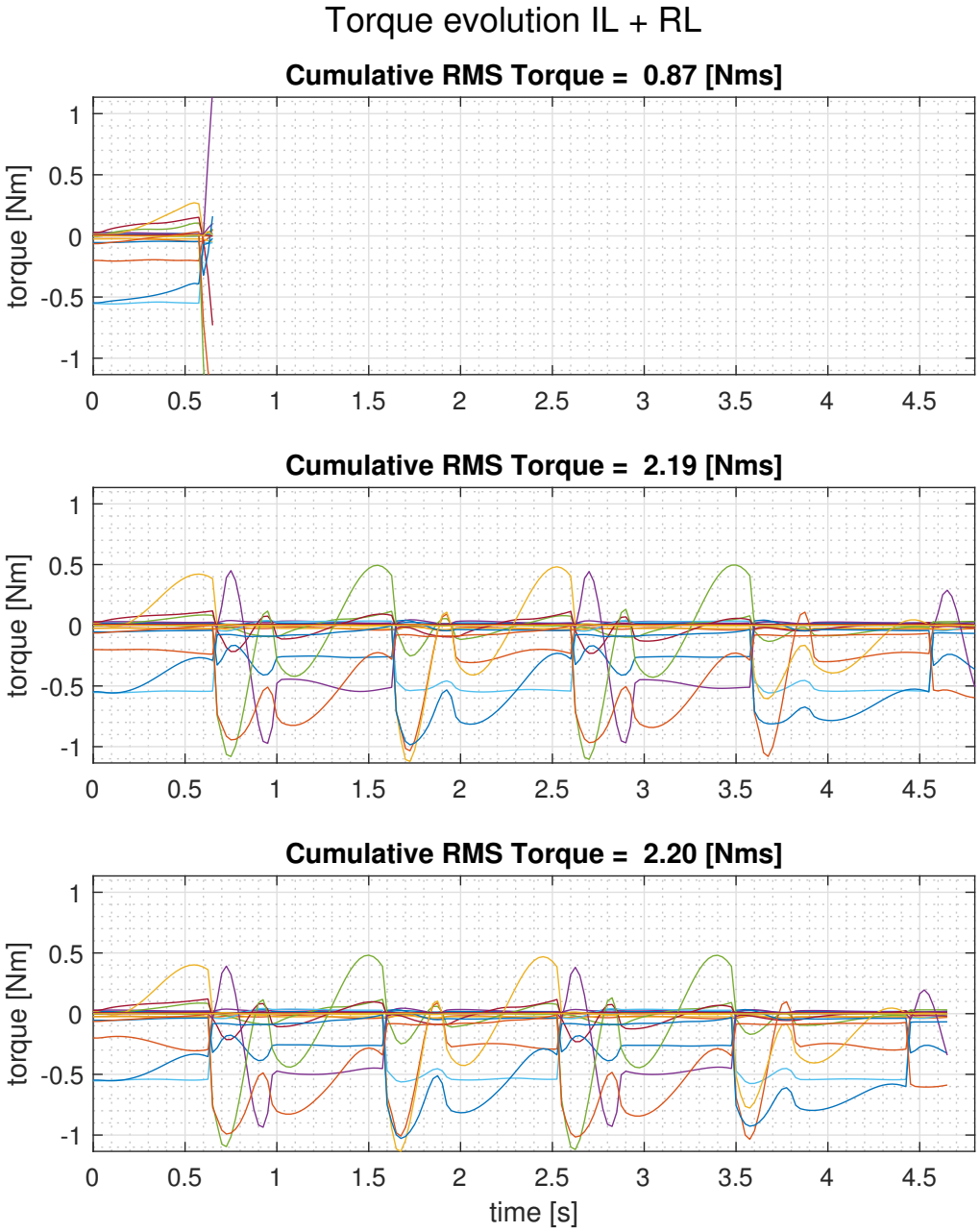


Figure 5-27.: Sample of torque evolution through sampled iterations for MAX-E2 - At the start, the cumulative RMS torque is low because the direct imitation policy is not feasible and results in early falls. As the iterations pass, the successful experiments get similar torques close to the performance obtained with RL from random conditions and present light improvements in velocity (faster finish times)

6. Conclusions, Future Work and Recommendations

This thesis addressed the challenge of using artificial intelligence techniques to train robotic systems for bipedal locomotion tasks. We aimed to achieve optimized performances in terms of energy consumption and execution speed despite the limitations imposed by the reality gap. We proposed a modular training scheme integrating Bézier based trajectories, reinforcement learning, and imitation learning techniques to achieve this. The framework allowed us to integrate various approaches to improve the performance of bipedal robots in locomotion tasks.

Chapter 1 motivated the study, highlighting the remarkable achievements of artificial intelligence in various domains, including board games and video games. The chapter also identified the challenges associated with implementing artificial learning techniques for robots, emphasizing the significance of addressing the reality gap, result variability, and the high costs involved in robot experiments. Chapter 2 presented an overview of the fundamental concepts required for modeling and analyzing humanoid robots. Non-linear control techniques applicable to this type of robot were discussed, along with specific details of the robots used in our project. Chapter 3 focused on the concept of imitation learning. We explored the use of human teachers to demonstrate tasks to robots and emphasized the importance of refining learned behaviors. Challenges in imitation were discussed, particularly the differences between robots (especially smaller ones) and humans. We developed a geometric approach using Motion Capture data for learning trajectories, which provided stable walking patterns and served as an informed initial guess for optimization processes. In Chapter 4, the concept of reinforcement learning was presented. We provided an overview of the most relevant methods for humanoid robot tasks and introduced our Bézier trajectories tuning policy-based approach. The use of reward functions and exploration improvement methods were discussed. Additionally, the reality gap problem was addressed, and strategies to mitigate this gap, such as randomization and robust control models, were explored. Chapter 5 presented the results of the individual proposed modules and their viable combinations: manually tuned Bézier trajectories, trajectories obtained through our proposed imitation learning method, and trajectories optimized through reinforcement learning from random and imitation-based initial conditions. The efficacy of our proposed methods was demonstrated, and the best approach for the humanoid locomotion task was determined considering the limitations imposed by the reality gap.

6.1. Final Training Strategy

As shown in Section 5.2.4, the best strategy explored for optimizing the performance of the humanoid robots in bipedal gait problems employed all of the tools developed. The Bézier trajectories and non-linear control strategies served as the base model for the policy, the initialization of the optimization process from imitation learning-based initial conditions sped up the convergence in most situations. It reduced the variability of the results, and the optimization through reinforcement learning improved the performance of the proposed policies until the achievement of stable, efficient, and transferable gaits in the simulation and in reality.

6.1.1. The Imitation Process

Efficiently transferring information from human gait patterns to humanoid robots using Reinforcement Learning necessitates a thoughtful approach to account for the differences between the specific robot and the subject of imitation. Directly comparing the two entities is generally not feasible due to these inherent disparities. We employed a process to map the relevant information to a shared space that the robot can comprehend and follow to address this challenge.

However, the journey does not end with mapping the data to a shared space. Noise in the data and variations in the dynamics of the robot and humans can lead to discrepancies between the mapped trajectories and the robot's capabilities. Simply attempting to track the data, even in the shared space ideally, may result in failed behaviors characterized by collisions or instabilities.

To tackle this issue, we utilized a geometric retargeting process, as presented in Section 3.5, to adjust the resulting trajectories and ensure their feasibility in the robot's motion. This retargeting process enabled the robots to achieve stable walking patterns both in simulation and in reality, mirroring the behavior of the humans during the imitation learning process.

By integrating Reinforcement Learning with the geometric retargeting process, we successfully facilitated the efficient transfer of human gait patterns to humanoid robots. The combination of these techniques contributed to enhanced robot locomotion capabilities, showcasing a significant step forward in closing the reality gap and achieving more robust and human-like behaviors in robotic systems. However, further research and refinement in this area hold the potential for even more remarkable advancements in robotics and artificial intelligence.

6.1.2. The Reinforcement Learning Optimization

The second specific objective of this study aimed to design a reinforcement learning algorithm that could iteratively enhance the performance of the bipedal locomotion gait in terms of energy consumption and moving velocity. To address this objective, we employed a policy-based approach to reinforcement learning. This allowed us to optimize and refine

the gait patterns generated by randomly initialized policies and policies initialized through imitation learning. Through the reinforcement learning process, we achieved significant improvements in gait efficiency. Initially, we considered that optimizing for both velocity and torque consumption might lead to contradicting objectives, as faster movements would typically result in higher instantaneous energy consumption. However, the optimization process revealed an exciting finding. While slow movements exhibited lower instantaneous energy consumption, they led to more significant overall energy consumption over time. On the other hand, the policies that achieved higher moving velocities corresponded with the most efficient ones, as they allowed the robot to complete tasks more swiftly, thereby reducing the overall energy expenditure. The iterative nature of the reinforcement learning process enabled us to continuously explore the search space of policies, refining them over time to strike an optimal balance between moving velocity and energy consumption. By leveraging this approach, we improved gait efficiency, resulting in more efficient locomotion patterns for the bipedal robots.

It is worth noting that the policy structure and additional considerations effectively narrowed the reality gap so that experiments conducted in simulation and with real robots presented very close behaviors. This process was crucial in mitigating the reality gap problem, allowing us to develop policies that performed well in simulation and transferred effectively to real-world environments. This ability to adapt and optimize policies for real robots ensures the practical applicability of our approach, making it a valuable tool for training robots to perform complex locomotion tasks efficiently. In conclusion, based on policy optimization, our reinforcement learning algorithm proved to be highly effective in iteratively enhancing bipedal gait performance concerning energy consumption and moving velocity. By finding policies that strike an optimal balance between these objectives, we demonstrated improved efficiency in robot locomotion, making significant strides toward practically implementing artificial learning techniques in robotic systems.

6.1.3. The Policies Integration

Our proposed method offered a flexible and efficient integration of manual tuning, imitation learning, and reinforcement learning approaches to obtain optimized gaits for humanoid locomotion tasks. This integration was made possible through the parametrization considered, which was transversal to all the approaches. By employing this parametric framework, we could apply the different techniques independently or sequentially, taking advantage of the best characteristics of each method. The parameterization allowed us to easily incorporate manually tuned Bézier trajectories, trajectories obtained through imitation learning from human demonstrations, and trajectories optimized through reinforcement learning with random initial conditions or based on imitation learning results. Each approach contributed unique insights and advantages to the gait optimization process. The manual tuning approach provided a foundational understanding of the robot's dynamics and control, enabling

us to establish preliminary walking patterns. Imitation learning allowed us to refine these patterns, achieving stable and human-like gaits. Additionally, reinforcement learning further improved the gaits, optimizing them for energy consumption and movement speed, resulting in more efficient locomotion. This integrated approach offered a holistic strategy for achieving high-performance gaits that balanced stability and efficiency. The ability to apply these techniques independently or in combination provides a powerful toolkit for addressing various challenges and fine-tuning the walking behaviors of humanoid robots.
In conclusion, our parametric integration framework allowed us to leverage the strengths of manual tuning, imitation learning, and reinforcement learning to obtain optimized gaits, thereby advancing the development of robust and efficient bipedal locomotion for humanoid robots.

6.1.4. Comparison Between Approaches

Integrating the modules, including Non-Linear Control and Bézier Trajectories, Imitation Learning, and Reinforcement Learning, played a crucial role in achieving robust and optimized performances for bipedal locomotion tasks. Each approach demonstrated transferable policies that could enhance the robot's locomotion abilities. The Reinforcement Learning approach stood out as the most promising in achieving high-performance locomotion among the integrated modules. The Reinforcement Learning approach, with its iterative optimization process, allowed the robot to adapt and refine its behavior through interactions with the environment. The results from Reinforcement Learning showed a considerable improvement in energy efficiency and execution speed, surpassing the performances achieved through Non-Linear Control, Bézier Trajectories, or Imitation by themselves.
As expected, initializing the Reinforcement Learning approach from imitation learning conditions proved highly effective. Using the imitation learning results as a starting point, the Reinforcement Learning algorithm demonstrated comparable performances with lower variability and faster convergence than random initial conditions. This initialization technique played a vital role in reducing the exploration effort required during Reinforcement Learning training, resulting in a higher sample efficiency. In summary, integrating the individual modules allowed for the development an efficient and adaptive robot training strategy for bipedal locomotion tasks. The Reinforcement Learning approach, particularly when initialized from imitation learning conditions, showed the most promising modular results, achieving superior performances with reduced variability and faster convergence.

6.2. Future Work

This research's successful results and insights open up several avenues for future exploration and improvement in humanoid robotics and artificial intelligence. The following are some promising directions for further investigations:

- **Extension to More Complex Locomotion Patterns:** The current work focused on achieving stable and efficient walking gaits. Future research could extend the proposed methods to tackle more complex locomotion patterns, such as jogging or running. This extension would require developing policies that can handle higher speeds and more dynamic movements, potentially involving new stability and energy efficiency challenges.

- **Utilization of Higher-Class Robots:** Expanding the research to higher-class robots with better mechanical properties and actuators could lead to using faster and more efficient control algorithms. These robots would offer increased capabilities for executing complex and dynamic movements, enabling the development of more sophisticated locomotion behaviors.

- **Experiments During the Optimization Process:** Inspired by [Koos et al., 2013], future work could explore the possibility of incorporating experiments at the end of the optimization process for validation and during the optimization itself. This approach would enable adaptive learning and faster convergence by using real-world data to guide the optimization process.

- **Flexible Bézier Trajectories for Reinforcement Learning:** Instead of treating Bézier trajectories as rigid impositions, an alternative approach could involve using them as a guiding framework for Reinforcement Learning approaches. The parameters of Bézier curves could become part of the policy's action space, allowing the robot to learn and adapt its walking patterns within the constraints defined by the Bézier trajectories.

- **Robustness Techniques:** Investigating techniques to improve the robustness of learned policies is crucial for practical deployment. Research could focus on perturbation rejection strategies, domain randomization methods, or other approaches to enhance the stability and adaptability of the policies to varying environments.

- **Model Predictive Control (MPC):** Integrating Model Predictive Control with the existing framework could further improve the performance of the gaits. MPC allows the robot to anticipate and adjust its actions based on predictions of future states, leading to more dynamic and responsive locomotion behaviors.

- **Multilevel Simulators:** Developing multilevel simulators could significantly accelerate the optimization process. A simple and less precise simulator would be used for most tests in this approach. In contrast, a middle-level simulator (or a set of them) would be employed for fewer simulations of promising policies. Finally, the best policies from the earlier stages would be tested in a more realistic simulator (or even in a real robot), providing a multi-fidelity optimization framework.

- **Adaptation to Varying Terrains:** Extending the proposed methods to enable humanoid robots to adapt to varying terrains is a critical aspect of real-world applications. Research could explore methods to make the policies more terrain-aware and capable of adjusting their walking patterns based on the ground conditions.

These future research directions hold the potential to enhance the capabilities of humanoid robots and contribute to the broader field of artificial intelligence and robotics. Exploring these avenues will pave the way for more sophisticated and efficient locomotion behaviors, enabling humanoid robots to perform complex tasks in real-world environments with greater adaptability and robustness.

6.3. Recommendations

The research presented in this thesis has made strides in improving humanoid robot locomotion by integrating artificial intelligence techniques. Based on the findings and insights obtained from this study, several key recommendations are proposed for future research and development in the field of humanoid robotics and artificial learning:

- **Validation and Real-World Testing:** It is recommended to conduct extensive real-world testing with physical humanoid robots to validate the proposed methods. Real-world experiments will provide crucial insights into the practical applicability and adaptability of the developed techniques across different environments and scenarios.
- **Exploration of Additional Locomotion Patterns:** Building upon the successful optimization of walking gaits, future research should investigate more complex locomotion patterns, such as jogging, running, turning, or jumping. Extending the research to encompass a broader range of movements will open up new possibilities for humanoid robots in various applications.
- **Comparison with State-of-the-Art Approaches:** To benchmark the proposed methods' effectiveness, researchers should thoroughly compare them with state-of-the-art approaches in humanoid locomotion. This analysis will help understand the advantages and limitations of the developed techniques compared to existing methodologies.
- **Parameter and Policy Structure Optimization:** Further exploration of parameterization and policy structures is recommended to fine-tune the performance of the proposed methods. Optimizing these aspects could lead to improved results and more efficient policies for humanoid locomotion.
- **Generalization to Different Robot Platforms:** It is essential to investigate the generalization capability of the developed methods to different robot platforms and morphologies. Ensuring the policies can adapt to diverse humanoid robots will enhance their applicability in real-world scenarios.

- **Real-Time Implementation:** Explore methods to implement the proposed policies in real-time on physical robots. Considering robot hardware's computational and processing constraints will be critical to achieving real-time feasibility.
- **Collaboration and Data Sharing:** Encourage collaboration and data sharing within the research community. Openly sharing datasets, simulation environments, and policy implementations can accelerate progress and facilitate reproducibility in the field.
- **Human-Robot Interaction Enhancement:** Investigate methods to improve human-robot interaction during learning. Developing techniques for human trainers to teach efficiently and correct robot behaviors during imitation learning can lead to more effective learning.
- **Integration of Perception:** Explore integrating perception and sensor data into learning. Leveraging sensory information from cameras, depth sensors, or other onboard sensors could enhance the robot's awareness and adaptability in dynamic environments.
- **Transfer Learning and Multi-Task Learning:** Investigate the potential of transfer and multi-task learning in humanoid locomotion. Leveraging knowledge gained from one task to improve performance in another could lead to more efficient and versatile locomotion behaviors.
- **Deployment in Real-World Scenarios:** Consider real-world applications and use cases for humanoid robots with optimized locomotion policies. This could include scenarios such as search and rescue operations, human-assisted tasks, or assisting people with mobility impairments.

By addressing these recommendations, the research community can advance the development of robust and efficient humanoid locomotion, ultimately contributing to the practical implementation of artificial learning techniques in real-world robotic systems.

A. Appendix A: Example Free Body Diagrams

Figures **A-1** to **A-6** depict example free body diagrams that have been utilized to evaluate the forces and reactions acting on humanoid robots, which are the focus of this thesis. These diagrams are essential tools for understanding and analyzing the mechanical interactions and dynamics involved in the robot's movements and interactions with its environment. Through these diagrams, we can gain insights into the distribution of forces, torques, and reaction forces at various points of interest on the robot's body, allowing us to study the stability, balance, and overall performance of the robot during different phases of its motion.

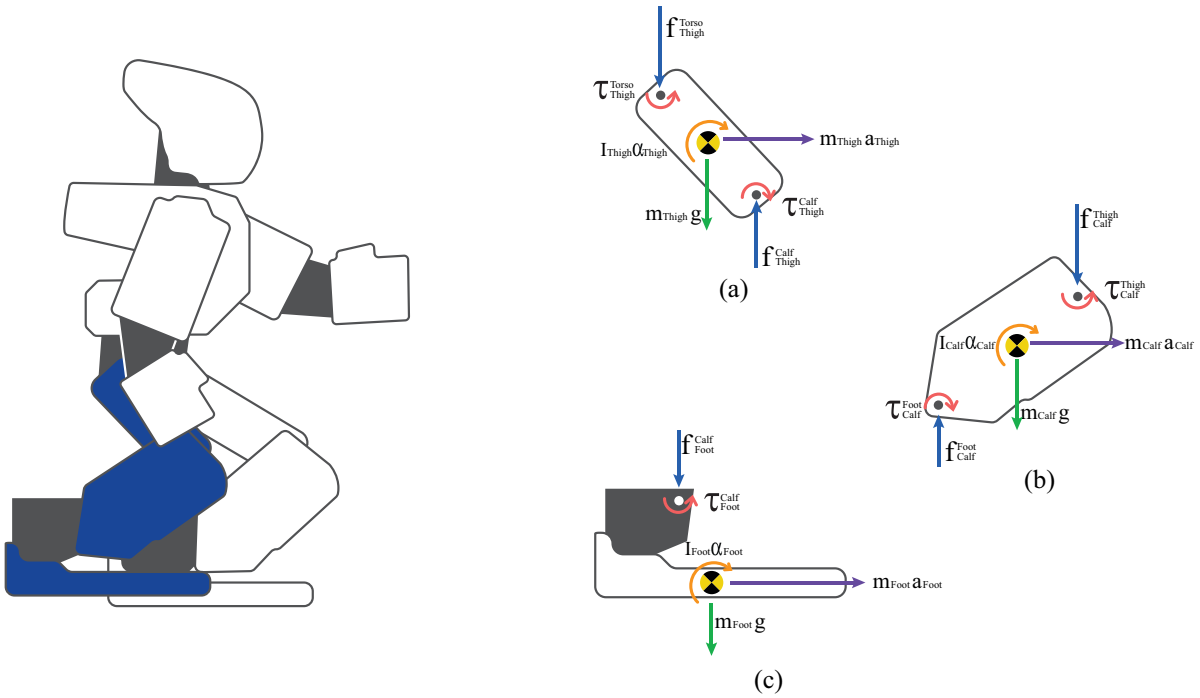


Figure A-1.: Humanoid Robot Free Body Diagrams Right View - (a) Balancing Thigh (b) Balancing Calf (c) Balancing foot

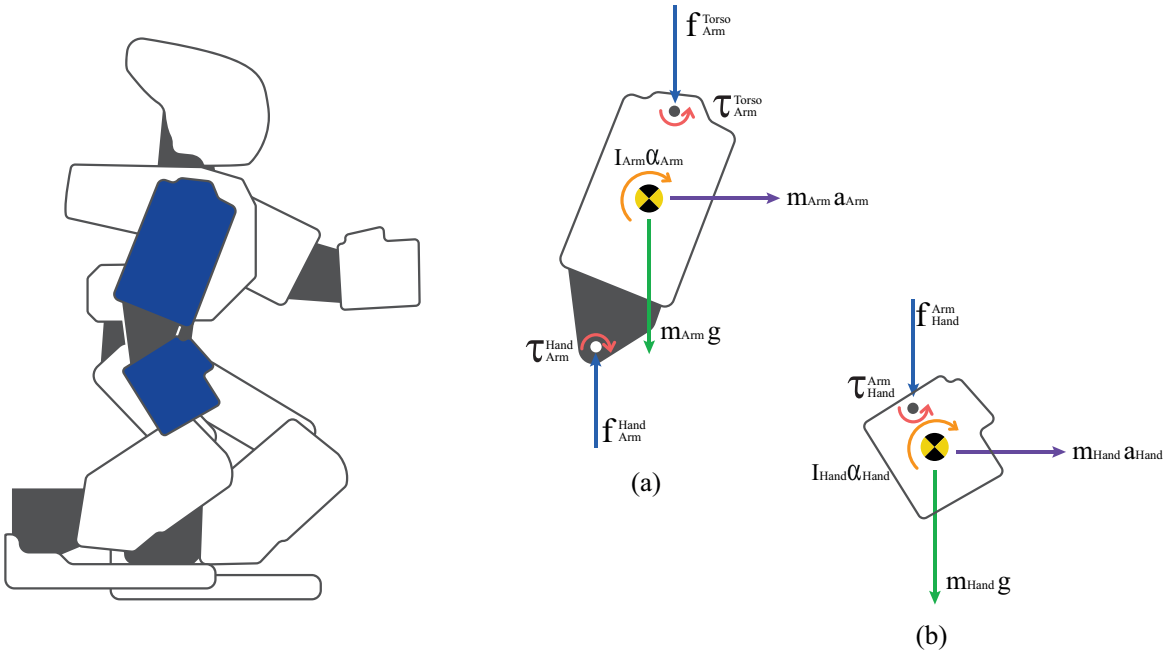


Figure A-2.: Humanoid Robot Free Body Diagrams Right View - (a) Balancing Arm (b) Balancing Forearm

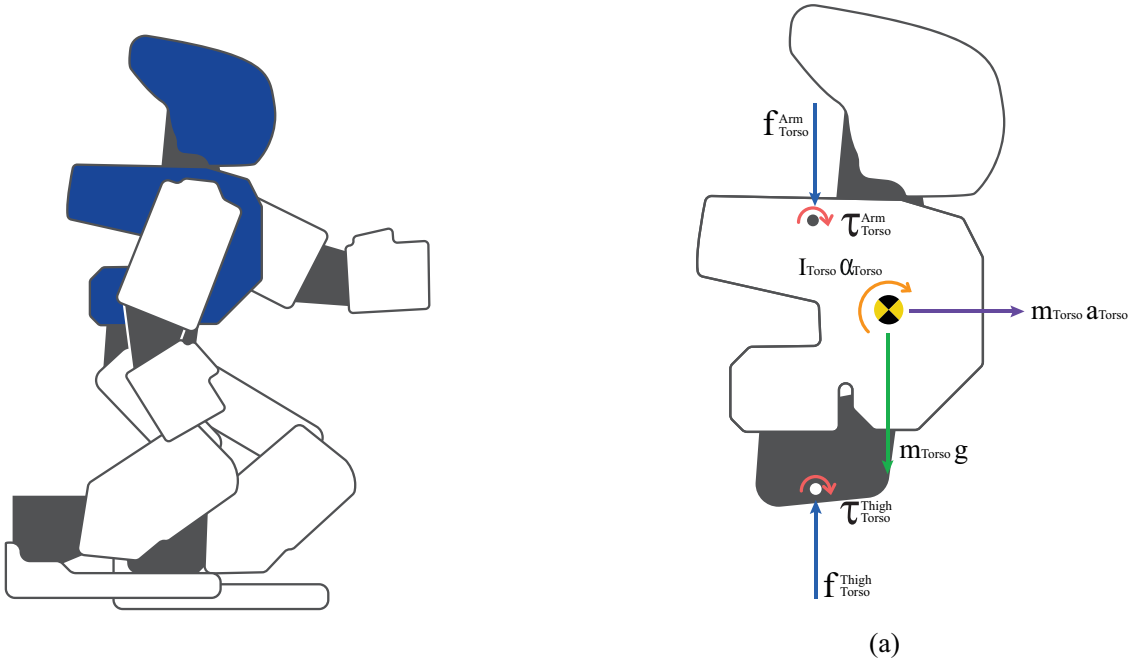


Figure A-3.: Humanoid Robot Free Body Diagrams Right View - Torso

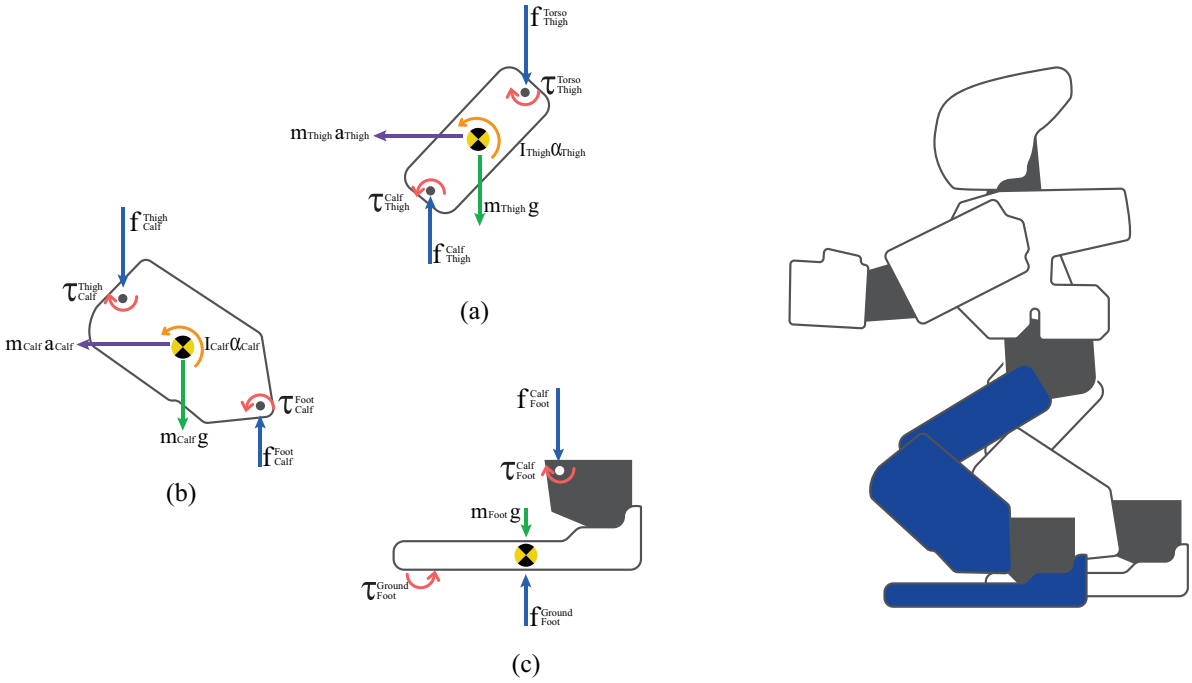


Figure A-4.: Humanoid Robot Free Body Diagrams Left View - (a) Balancing Thigh (b) Balancing Calf (c) Balancing foot

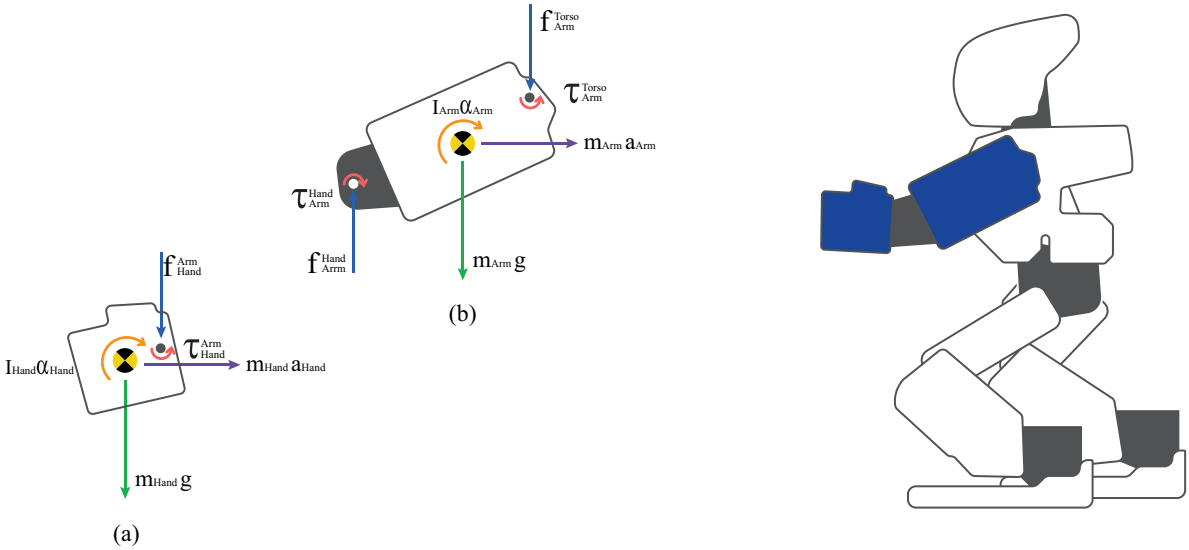


Figure A-5.: Humanoid Robot Free Body Diagrams Left View - (a) Balancing Arm (b) Balancing Forearm

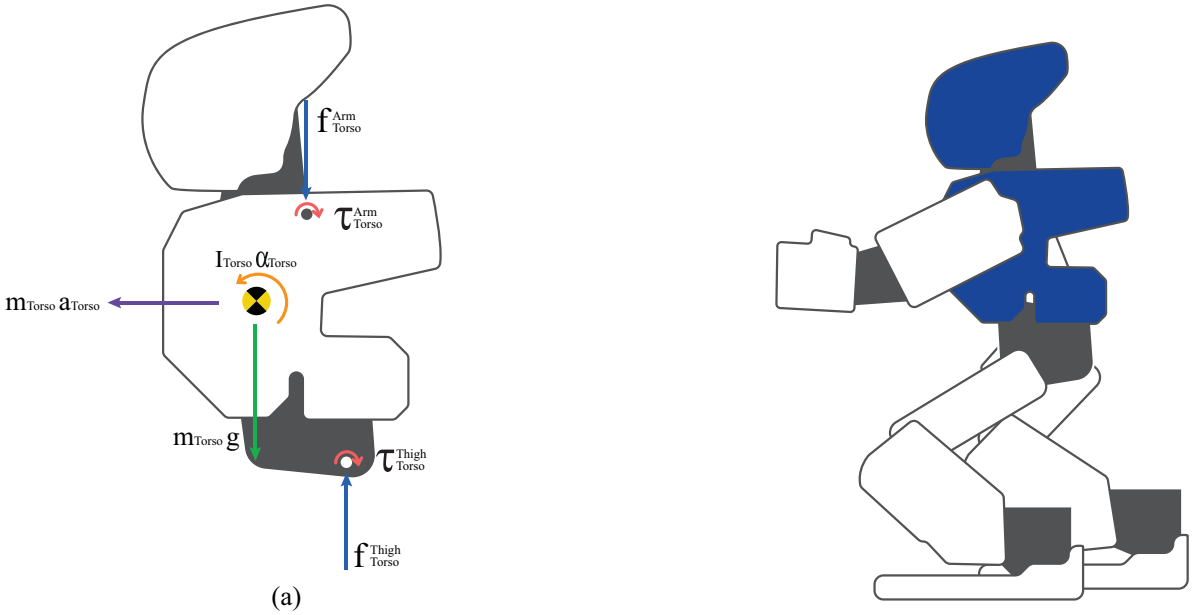


Figure A-6.: Humanoid Robot Free Body Diagrams Left View - Torso

B. Appendix B: Darwin-Mini Kinetic Parameters Tables

Tables B-1 to B-11 present the inertia matrices of each one of Darwin's links concerning its CoM.

Link	Mass [g]
Foot	37.6
Ankle	19.67
Calf	9.432
Thigh	19.94
Hip	23.27
Torso	241.17
Shoulder	0.446
Elbow	23.27
Forearm	26.59

Table B-1.: Darwin Mini - Link masses in grams [g]

12470.0	362.9	6653.0
362.9	27700.0	-454.6
6653.0	-454.6	26690.0

Table B-2.: Inertia matrix of the Darwin Mini Foot [gmm^2]

2757	-1.631	-118.8
-1.631	2810	10.83
-118.8	10.83	2157

Table B-3.: Inertia matrix of the Darwin Mini Ankle [gmm^2]

4958	0	473.4
0	3647	0
473.4	0	3117

Table B-4.: Inertia matrix of the Darwin Mini Calf [gmm^2]

9750	-166.862	-166.862
-166.862	6281	-740.398
-166.862	-740.398	5127

Table B-5.: Inertia matrix of the Darwin Mini Thigh [gmm^2]

6477	0	-1.52
0	5428	16.19
-1.52	16.19	2997

Table B-6.: Inertia matrix of the Darwin Mini Hip [gmm^2]

285500	66.01	-45520
66.01	255600	69.22
-45520	69.22	195300

Table B-7.: Inertia matrix of the Darwin Mini Torso [gmm^2]

0.4989	0	0
1	1.158	-0.1329
0	-0.1329	1.161

Table B-8.: Inertia matrix of the Darwin Mini Shoulder [gmm^2]

9750	-166.862	-166.862
-166.862	6281	-740.398
-166.862	-740.398	5127

Table B-9.: Inertia matrix of the Darwin Mini Elbow [gmm^2]

15420	195.9	377.6
195.9	15800	-1307
377.6	-1307	2796

Table B-10.: Inertia matrix of the Darwin Mini Forearm [gmm^2]

C. Appendix C: MAX-E2 Kinetic Parameters Tables

Tables C-1 to C-10 present the inertia matrices of each one of the MAXE's links concerning their corresponding CoMs.

9319	-146.034	-562.945
-146.034	22038	-471.863
-562.945	-471.863	28341

Table C-1.: Inertia matrix of MAXE's Foot [gmm^2]

27900	-631.782	-282.779
-631.782	27900	-282.9823
-282.779	-282.9823	20073

Table C-2.: Inertia matrix of MAXE's Ankle [gmm^2]

11857	-7.195	-323.024
-7.195	6281	-677.988
-323.024	-677.988	11857

Table C-3.: Inertia matrix of MAXE's Calf [gmm^2]

9750	-166.862	-166.862
-166.862	6281	-740.398
-166.862	-740.398	5127

Table C-4.: Inertia matrix of MAXE's Thigh [gmm^2]

27900	-631.782	-282.779
-631.782	27900	-282.9823
-282.779	-282.9823	20073

Table C-5.: Inertia matrix of MAXE's Hip [gmm^2]

203036	-44.242	25204
-44.242	223988	-4.169
25204	-4.169	320987

Table C-6.: Inertia matrix of MAXE's Waist [gmm^2]

126984	9200	19871
9200	143417	-5694
19871	-5694	188171

Table C-7.: Inertia matrix of MAXE's Chest [gmm^2]

27900	-631.782	-282.779
-631.782	27900	-282.9823
-282.779	-282.9823	20073

Table C-8.: Inertia matrix of MAXE's Shoulder [gmm^2]

26636	25.786	1192
25.786	26272	-508.958
1192	-508.958	11592

Table C-9.: Inertia matrix of MAXE's Elbow [gmm^2]

13426	418.394	7251
418.394	20157	-662.211
7251	-662.211	12793

Table C-10.: Inertia matrix of MAXE's Forearm [gmm^2]

References

[Arcos-Legarda et al., 2019] Arcos-Legarda, J., Cortes-Romero, J., Beltran-Pulido, A., and Tovar, A. (2019). Hybrid disturbance rejection control of dynamic bipedal robots. *Multibody System Dynamics*, 46:281–306.

[Belongie, 2023] Belongie, S. (2023). Rodrigues' rotation formula. Created by Eric W. Weisstein.

[Burda et al., 2018] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018). Large-Scale Study of Curiosity-Driven Learning. *ArXiv*.

[Carnegie Mellon University Graphics Lab, 2004] Carnegie Mellon University Graphics Lab (2004). CMU Motion Capture Database. The database was created with funding from NSF EIA-0196217.

[Chebotar et al., 2018] Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. (2018). Closing the sim-to-real loop: Adapting simulation randomization with real world experience.

[Chernova and Thomaz, 2014] Chernova, S. and Thomaz, A. L. (2014). Robot Learning from Human Teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121.

[Chernova and Veloso, 2010] Chernova, S. and Veloso, M. (2010). Confidence-based multi-robot learning from demonstration. *International Journal of Social Robotics*, 2(2):195–215.

[Chevallereau et al., 2014] Chevallereau, C., Sinnet, R. W., Ames, A. D., and Université, L. (2014). Models feedback control and open problems of 3d bipedal robotic walking. *Automatica*.

[Collins et al., 2005] Collins, S., Ruina, A., Tedrake, R., and Wisse, M. (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307:1082–1085.

[Ding et al., 2019] Ding, J., Zhou, C., and Xiao, X. (2019). Energy-Efficient Bipedal Gait Pattern Generation via CoM Acceleration Optimization. *IEEE-RAS International Conference on Humanoid Robots*, 2018-Novem:238–244.

[Duan et al., 2017] Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. (2017). One-Shot Imitation Learning. *ArXiv*.

[Duan et al., 2016] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking Deep Reinforcement Learning for Continuous Control. *ArXiv*.

[Erez et al., 2015] Erez, T., Lowrey, K., Tassa, Y., Kumar, V., Kolev, S., and Todorov, E. (2015). An integrated system for real-time model predictive control of humanoid robots. *Proceedings on IEEE-RAS International Conference on Humanoid Robots*, 2015-Febru(February):292–299.

[García et al., 1989] García, C. E., Prett, D. M., and Morari, M. (1989). Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348.

[Google, 2015] Google (2015). AlphaGo — DeepMind. `https://deepmind.com/research/alphago/`. accessed 20-May-2019.

[Heess et al., 2017] Heess, N., TB, D., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M., and Silver, D. (2017). Emergence of locomotion behaviours in rich environments.

[Ijspeert, 2008] Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653.

[Jumper et al., 2021] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583 – 589. Cited by: 8028; All Open Access, Green Open Access, Hybrid Gold Open Access.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization.

[Kobayashi et al., 2018] Kobayashi, T., Sekiyama, K., Hasegawa, Y., Aoyama, T., and Fukuda, T. (2018). Virtual-dynamics-based reference gait speed generator for limit-cycle-based bipedal gait. *ROBOMECH Journal*, 5(1).

[Koenemann et al., 2014] Koenemann, J., Burget, F., and Bennewitz, M. (2014). Real-time imitation of human whole-body motions by humanoids. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2806–2812.

[Koos et al., 2013] Koos, S., Mouret, J. B., and Doncieux, S. (2013). The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145.

[Kuindersma et al., 2016] Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R. (2016). Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, 40:429–455.

[Lab, 2003] Lab, C. M. U. C. G. (2003). Motion capture library.

[Lee et al., 2019] Lee, K., Kim, S., Lim, S., Choi, S., and Oh, S. (2019). Tsallis Reinforcement Learning: A Unified Framework for Maximum Entropy Reinforcement Learning. *ArXiv*.

[Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *ArXiv*.

[Lillicrap et al., 2019] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning.

[Liu et al., 2022] Liu, S., Lever, G., Wang, Z., Merel, J., Eslami, S., Hennes, D., Czarnecki, W., Tassa, Y., Omidshafiei, S., Abdolmaleki, A., Graepel, T., and Heess, N. (2022). From motor control to team play in simulated humanoid football. *Science Robotics*, 7.

[Loudon et al., 2008] Loudon, J. K. J. K., Swift, M., and Bell, S. (2008). *The clinical orthopedic assessment guide*. Human Kinetics.

[Luenberger and Ye, 2021] Luenberger, D. G. and Ye, Y. (2021). Penalty and barrier methods. *International Series in Operations Research and Management Science*, 228.

[Ma et al., 2021] Ma, L.-K., Yang, Z., Xin, T., Guo, B., and Yin, K. (2021). Learning and exploring motor skills with spacetime bounds. *Computer Graphics Forum*, 40(2):251–263.

[Ma et al., 2018] Ma, W.-L., Or, Y., and Ames, A. D. (2018). Dynamic Walking on Slippery Surfaces: Demonstrating Stable Bipedal Gaits with Planned Ground Slippage. *Proceedings on 2019 International Conference on Robotics and Automation (ICRA)*.

[Mania et al., 2018] Mania, H., Guy, A., and Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*.

[Mark W. Spong, 2020] Mark W. Spong, Seth Hutchinson, M. V. (2020). *Robot modeling and control*. JOHN WILEY and SONS, INC.

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

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529 – 533. Cited by: 15639.

[Müller et al., 2022] Müller, M., Jazdi, N., and Weyrich, M. (2022). Self-improving models for the intelligent digital twin: Towards closing the reality-to-simulation gap. *IFAC-PapersOnLine*, 55:126–131.

[Nehaniv and Dautenhahn, 2002] Nehaniv, C. L. and Dautenhahn, K. (2002). *The Correspondence Problem*, page 41–61. MIT Press, Cambridge, MA, USA.

[Nguyen and La, 2019] Nguyen, H. and La, H. (2019). Review of Deep Reinforcement Learning for Robot Manipulation. *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, pages 590–595.

[Niu et al., 2022] Niu, J., Hu, Y., Li, W., Huang, G., Han, Y., and Li, X. (2022). Closing the dynamics gap via adversarial and reinforcement learning for high-speed racing. *Proceedings of the International Joint Conference on Neural Networks*, 2022-July.

[OpenAI, 2018] OpenAI (2018). Openai five. `https://blog.openai.com/openai-five/`. accessed 20-May-2019.

[Recht, 2018] Recht, B. (2018). A Tour of Reinforcement Learning: The View from Continuous Control. *eprint arXiv:1806.09460*.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407.

[ROBOTIS, 2022a] ROBOTIS (2022a). Robotis engineer kit 1 e-manual (visited on 2022/04/01).

[ROBOTIS, 2022b] ROBOTIS (2022b). *Robotis Engineer Kit 2 E-Manual, (visited on 2022/04/01)*.

[ROBOTIS, 2022c] ROBOTIS (2022c). Robotis mini e-manual (visited on 2022/04/01).

[ROBOTIS, 2022d] ROBOTIS (2022d). Robotis xl-320 e-manual (visited on 2022/04/01).

[Rodriguez et al., 2018] Rodriguez, D., Brandenburger, A., and Behnke, S. (2018). Combining Simulations and Real-robot Experiments for Bayesian Optimization of Bipedal Gait Stabilization. *Lecture Notes in Computer Science book series*.

[Rosolia and Borrelli, 2016] Rosolia, U. and Borrelli, F. (2016). Learning Model Predictive Control for Iterative Tasks. *eprint arXiv:1609.01387*, 4.

[Salvato et al., 2021] Salvato, E., Fenu, G., Medvet, E., and Pellegrino, F. A. (2021). Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187.

[Schulman et al., 2015] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust Region Policy Optimization. *Proceedings of the 32nd International Conference on Machine Learning*.

[Siciliano and Khatib, 2016] Siciliano, B. and Khatib, O. (2016). *Springer handbook of robotics*. Springer eBooks.

[Siegwart et al., 2011] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots.* Intelligent robotics and autonomous agents. The MIT Press/Massachusetts Institute of Technology.

[Simba et al., 2016] Simba, K. R., Uchiyama, N., and Sano, S. (2016). Real-time smooth trajectory generation for nonholonomic mobile robots using bézier curves. *Robotics and Computer-Integrated Manufacturing*, 41.

[Singh et al., 2019] Singh, A., Yang, L., Hartikainen, K., Finn, C., and Levine, S. (2019). End-to-End Robotic Reinforcement Learning without Reward Engineering. *Robotics: Science and Systems*.

[Sutton and Barto, 2008] Sutton, R. S. and Barto, A. G. (2008). *Reinforcement Learning*. The MIT Press, London, England, second edi edition.

[Świechowski et al., 2022] Świechowski, M., Godlewski, K., Sawicki, B., and Mańdziuk, J. (2022). Monte carlo tree search: a review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

[Tan et al., 2018] Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. (2018). Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *eprint arXiv:1804.10332*.

[Tedrake, 2022] Tedrake, R. (2022). *Underactuated Robotics*. MIT.

[Tevatia and Schaal, 2000] Tevatia, G. and Schaal, S. (2000). Inverse kinematics for humanoid robots. *Proceedings-IEEE International Conference on Robotics and Automation*, 1(April):294–299.

[Thorp, 2023] Thorp, H. H. (2023). Chatgpt is fun, but not an author. *Science*, 379(6630):313. Cited by: 157; All Open Access, Bronze Open Access.

[Tobin et al., 2017] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:23–30.

[Todorov, 2019] Todorov, E. (2019). Mujoco: Modeling, simulation and visualization of multi-joint dynamics with contact. `http://www.mujoco.org/book/index.html`. accessed 20-May-2019.

[Uchibe, 2018] Uchibe, E. (2018). Cooperative and competitive reinforcement and imitation learning for a mixture of heterogeneous learning modules. *Frontiers in Neurorobotics*, 12(SEP).

[Vinyals et al., 2019] Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., and Silver, D. (2019). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/{}`. accessed 20-May-2019.

[Vukobratovic and Borovac, 2004] Vukobratovic, M. and Borovac, B. (2004). Zero-moment point — thirty five years of its life. *International Journal of Humanoid Robotics*, 01:157–173.

[Vukobratovic and Juricic, 1969] Vukobratovic, M. and Juricic, D. (1969). Contribution to the synthesis of biped gait. *IEEE Transactions on Biomedical Engineering*, BME-16(1):1–6.

[Wampler, 1986] Wampler, C. W. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man and Cybernetics*, 16.

[Watkins and Dayan, 1992] Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[Westervelt et al., 2003] Westervelt, E. R., Grizzle, J. W., and Koditschek, D. E. (2003). Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48:42–56.

[Xie et al., 2019] Xie, Z., Clary, P., Dao, J., Morais, P., Hurst, J., and van de Panne, M. (2019). Iterative reinforcement learning based design of dynamic locomotion skills for cassie. *3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.*

[Xie et al., 2020] Xie, Z., Ling, H. Y., Kim, N. H., and Panne, M. V. D. (2020). Allsteps: Curriculum-driven learning of stepping stone skills. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2020*, pages 213–224.

[Zou et al., 2019] Zou, F., Shen, L., Jie, Z., Zhang, W., and Liu, W. (2019). A sufficient condition for convergences of adam and rmsprop. volume 2019-June.