Universidade de Aveiro
2022

Seyed
Mohammadreza
Mohades Kasaei

Locomoção de Humanoides Robusta e Versátil baseada em Controlo Analítico e Física Residual

A Robust, Agile and Versatile Humanoid Locomotion based on Analytical Control and Residual Physics

Universidade de Aveiro
2022

Seyed
Mohammadreza
Mohades Kasaei

Locomoção de Humanoides Robusta e Versátil baseada em Controlo Analítico e Física Residual

A Robust, Agile and Versatile Humanoid Locomotion based on Analytical Control and Residual Physics

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Informática, realizada sob a orientação científica de José Nuno Panelas Nunes Lau e Artur José Carneiro Pereira, professores do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

The jury

presidente / president	**Doutor João Filipe Colardelle da Luz Mano** Professor Catedrático da Universidade de Aveiro
vogais / examiners committee	**Doutor Klaus Dorer** Professor Catedrático, Hochschule Offenburg, Alemanha
	Jorge Miguel Matos Sousa Pinto Professor Associado com Agregação, Universidade do Minho
	Doutor José Luís Sousa de Magalhães Lima Professor Coordenador, Instituto Politécnico de Bragança
	Doutor José Nuno Panelas Nunes Lau (Orientador) Professor Associado, Universidade de Aveiro
	Doutor Filipe Miguel Teixeira Pereira da Silva Professor Auxiliar, Universidade de Aveiro

Acknowledgements

As I finish this thesis I have the pleasure of recounting the people that have shaped my experience. My most sincere thanks go to my supervisors Prof. Nuno Lau and Prof. Artur Pereira for their support, guidance, and encouragement during the work on this dissertation. Their efforts in shaping my Ph.D. were absolutely invaluable. In addition, I would like to give very special thanks to my twin brother, Hamidreza Kasaei, for his support. I am lucky to have you.
Miguel Abreu has contributed to the development of the learning residual physics module and helped me to carried out a bunch of simulations. I was fortunate to have Miguel as my colleague, in a new country and a new lab. Additionally, I thank my colleagues, Gi Hyun Lim, Eurico Pedrosa and David Simões for their friendship, guidance, stimulating conversations and help when I needed. Also my regards to my close friends Morteza Hosseini, Vahid Mokhtari, Abbas Abdolmaleki, Nima Shafii, Ehsan Shahri, Maryam Sajedin and Mohammadreza Khalaj. I want to thank my parents, my brothers and my sister for their constant encouragement, strength, and support. This thesis, in many ways, is a result of the dedication of my parents to the education of their children. Finally, I want to thank the Portuguese Science Foundation (FCT) for the Ph.D. scholarship grant (SFRH/BD/118438/2016). I am very thankful to Institute of Electronics and Informatics Engineering of Aveiro (IEETA), my research institution, for providing the financial support for attending several conference events. I would also like to thank the open-source community for sharing and spreading the knowledge, for all to use.

Palavras-chave

Robôs humanoides, locomoção bípede, controlo ótimo, aprendizagem de física residual, otimização de política proximal, aprendizado por reforço profundo.

Resumo

Os robôs humanoides são feitos para se parecerem com humanos, mas suas habilidades de locomoção estão longe das nossas em termos de agilidade e versatilidade. Quando os humanos caminham em terrenos complexos ou enfrentam distúrbios externos combinam diferentes estratégias, de forma inconsciente e eficiente, para recuperar a estabilidade. Esta tese aborda o problema de desenvolver um sistema robusto para andar de forma omnidirecional, capaz de gerar uma locomoção para robôs humanoides versátil e ágil em terrenos complexos. Projetámos e desenvolvemos motores de locomoção sem modelos e baseados em modelos. Formulámos os controladores usando diferentes abordagens, incluindo esquemas de controlo clássicos e ideais, e validámos o seu desempenho por meio de simulações e experiências reais. Estes *frameworks* têm estruturas hierárquicas compostas por várias camadas. Essas camadas são compostas por vários módulos que são conectados entre si para diminuir a complexidade e aumentar a flexibilidade dos *frameworks* propostos. Adicionalmente, o sistema pode ser implementado em diferentes plataformas de forma fácil.

Acreditamos que o uso de aprendizagem automática sobre abordagens analíticas é a chave para abrir as portas para robôs humanoides saírem dos laboratórios. Propusemos um forte acoplamento entre controlo analítico e aprendizagem profunda por reforço. Expandimos o nosso controlador analítico com módulos de aprendizagem por reforço para aprender como regular os parâmetros do motor de caminhada (planeadores e controladores) de forma adaptativa e gerar resíduos para ajustar as posições das juntas alvo do robô (física residual). A eficácia das estruturas propostas foi demonstrada e avaliada em um conjunto de cenários de simulação desafiadores. O robô foi capaz de generalizar o que aprendeu em um cenário, exibindo habilidades de locomoção humanas em circunstâncias imprevistas, mesmo na presença de ruído e impulsos externos.

Keywords

Humanoid robots, biped locomotion, optimal control, learning residual physics, Proximal Policy Optimization (PPO), Deep Reinforcement Learning (DRL).

Abstract

Humanoid robots are made to resemble humans but their locomotion abilities are far from ours in terms of agility and versatility. When humans walk on complex terrains or face external disturbances, they combine a set of strategies, unconsciously and efficiently, to regain stability. This thesis tackles the problem of developing a robust omnidirectional walking framework, which is able to generate versatile and agile locomotion on complex terrains. We designed and developed model-based and model-free walk engines and formulated the controllers using different approaches including classical and optimal control schemes and validated their performance through simulations and experiments. These frameworks have hierarchical structures that are composed of several layers. These layers are composed of several modules that are connected together to fade the complexity and increase the flexibility of the proposed frameworks. Additionally, they can be easily and quickly deployed on different platforms.
Besides, we believe that using machine learning on top of analytical approaches is a key to open doors for humanoid robots to step out of laboratories. We proposed a tight coupling between analytical control and deep reinforcement learning. We augmented our analytical controller with reinforcement learning modules to learn how to regulate the walk engine parameters (planners and controllers) adaptively and generate residuals to adjust the robot's target joint positions (residual physics). The effectiveness of the proposed frameworks was demonstrated and evaluated across a set of challenging simulation scenarios. The robot was able to generalize what it learned in one scenario, by displaying human-like locomotion skills in unforeseen circumstances, even in the presence of noise and external pushes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans expect robots to be able to operate in human environments using skills identical to humans. For a long time, wheeled robots were used for research in the fields of Artificial Intelligence (AI) and robotics. Humanoid robots opened the doors to new research topics since wheeled robot locomotion is not adapted to many human environments. Unlike wheeled robots, bipedal robots can adapt to human environments and overcome challenges of these environments, such as stairs, gaps, and uneven terrain. Whilst humanoid robots overcome these limitations, the generation of stable and human-like walking is a complex subject that involves several disciplines. This complexity comes from their inherent unstable nature due to their complex kinematics as well as dynamics. Despite many research efforts in developing biped locomotion, the performance of biped locomotion is still far from the expectations.

Over the past decades, several approaches have been presented to construct stable walking frameworks for biped robots, which can be categorized into four major categories: *model-based*, *model-free*, *Central Pattern Generator (CPG)-based* and *hybrid* frameworks. *Model-based* frameworks are designed by considering the physical dynamics model of the systems, but uncertainties often affect their performance and prevent them from achieving optimal performance. *Model-free* frameworks are based on machine learning (ML) algorithms like reinforcement learning (RL). These approaches are trial intensive and need a considerable amount of samples that may cause many falls. Consequently, they cannot be used directly on real robots. *CPG-based* frameworks are inspired by nature and designed based on neurophysiological studies on animals. In these approaches, a set of oscillators are coupled together in a specific manner to generate rhythmic locomotion. Their greatest strengths are their flexibility, adaptability, and computational efficiency. On the other side, the difficulties in adjusting the parameters of the oscillators and adapting sensory information are their worst weaknesses. *Hybrid* frameworks combine the aforementioned approaches to leverage their different capabilities.

Nowadays, humanoid robots can perform only some simple tasks. To use humanoid robots in a real human environment, we believe they should be capable of generating omnidirectional human-like walking and be able to react appropriately to disturbances (e.g., external pushes, inclined surfaces, uneven terrain, etc.).

1.1 Motivation

Versatility is a coveted feature when designing humanoid robots. Their shape allows them to be extremely resourceful in our daily-life environments without the necessity of adjusting the surroundings. According to this distinctive property, humanoid robots have a wide range of applications from working in factories to helping elderly people. Despite a significant effort from the research community, their capabilities are still far from ours, particularly, in terms of speed, stability, and safety. People expect humanoid robots to walk robustly over any type of terrain and be able to recover from external perturbations.

In most of the proposed walking frameworks, the knowledge of robots is static and does not evolve from past experiences. Therefore, they need to at least re-tune the parameters to be able to adapt to new environments. To cope with these issues, several robotics groups have started to explore how to learn from previous experiences to achieve robustness and stability [1, 2, 3, 4]. This Ph.D. project aims to push forward the development of biped locomotion by coupling walking frameworks (model-based and model-free) with Deep Reinforcement Learning (DRL) algorithms to combine the potential of both approaches. This hybrid framework aims at generating robust, versatile, and agile omnidirectional walking gaits by exploring the full potential of the robot, taking advantage of the analytical solution's consistency and the flexibility of residual learning.

1.2 Research objectives

The main objective of this research is to study the topic of walking and push recovery in the humanoid robotic domain. To migrate a humanoid robot to new environments, one must often completely redesign and remodel the knowledge-base that it is running with. To cope with this limitation, we intend to propose a tight coupling between the analytical control approach and machine learning approaches in autonomous humanoid robots. This coupling provides automatic capabilities that will allow robots to (i) incrementally learn suitable actions/parameters from the set of accumulated experiences and (ii) reason about how to behave in response to external pushes. The proposed system is composed of two major components — an analytical planner and controller; and a fully connected neural network. The former is responsible for optimally controlling the overall state of the robot based on an abstract dynamics model. It is also responsible for generating reference trajectories using dynamic planners with genetically optimized parameters and overcome uncertainties up to a certain degree. The latter component — a fully connected network — is optimized with reinforcement learning to control the arms residuals, and the Center of Mass (COM) height of the robot, thus improving the upper body efficiency, adaptively updating the planners' parameters and controllers' gains as well as the constraints which impact the overall stability and speed of the robot. We believe that using machine learning on top of analytical approaches is the key to open doors for humanoid robots to step out of laboratories. This project mainly focuses on the following state-of-the-art questions:

- Despite the humanoid robots' versatility, why are they not as capable as us?
- How is it that humans can constantly change their direction when running while keeping their stability, but humanoids are not good as they are?
- How accurate should the (whole-body / abstract) dynamics model be?

To address these questions, the following specific objectives will be pursued:

- Acquire a deep understanding of a set of well-known abstract dynamics models based on Inverted Pendulum (IP).
- Design and develop a set of planners and controllers including classical and optimal control approaches to generate and track the walking reference trajectories.

- Design and develop a hierarchical framework to increase the flexibility and decrease the complexity.
- Develop of an abstract control interface for humanoid robots that enables the high-level artificial intelligence of these robots with transparent handling of lower-level issues.
- Investigate and use machine learning techniques such as optimization algorithms and deep reinforcement learning to learn how to adaptively update the parameters in unforeseen circumstances.
- Develop robust methods for transforming commands defined in a high-level language into real robot movements.
- Design and develop a set of scenarios including simulations and real robots to assess system performance comprehensively.

1.3 Outline

This thesis is structured in eight chapters and it would be mention that, to avoid distracting the reader and to provide better representation, some chapters contains repetitious figures and equations. The structure of this thesis is as follows: the first being this introduction. In chapter 2, we focus on legged robot locomotion approaches including model-based and CPG-based. Particularly, we design and develop a model-based and a CPG-based walk engine and perform a set of simulations and real robot experiments to show their performances. Chapter 3 is dedicated to investigating push recovery strategies in humanoid robots. In this chapter, we will study a set of well-known dynamics models and perform a set of numerical simulations to find the ablest model to keep the stability of robots. Chapter 4 is devoted to design and develop optimal control schemes to provide optimal and robust locomotion for humanoid robots. These controllers deal with finding control signals over a period of time such that an objective function is optimized. Besides, we will augment the controllers with an online footstep adjustment controller based on the prediction of the Divergent Component of Motion (DCM) at the end of each step, and we will show how this online modification increases the withstanding level of the robot against external pushes. Chapter 5 proposes a modular framework to generate robust biped locomotion using a tight coupling between an analytical walking approach and deep reinforcement learning. This framework is composed of six main modules which are hierarchically connected to reduce the overall complexity and increase its flexibility. In chapter 6, we approach Model Predictive Control (MPC) to develop a robust locomotion framework. The core of this framework is an abstract dynamics model which is composed of three masses to consider the dynamics of stance leg, torso, and swing leg for minimizing the tracking problems. Chapter 7 deals with learning residual physics. In this chapter, we will propose a robust hybrid stabilizer system and a CPG-ZMP based walk engine by pairing an analytical controller with a neural network with symmetric partial data augmentation to learn residuals. Finally, in Chapter 8, the conclusions are presented and future research directions are discussed.

1.4 Publications

The work presented in this thesis spawned a series of publications presented at conferences and journals in the field. Below is the list of manuscripts that have been published as part of this thesis:

- **Journals**

  [1] **Kasaei, Mohammadreza**, et al. "Robust Biped Locomotion Using Deep Reinforcement Learning on Top of an Analytical Control Approach." Robotics and Autonomous Systems (2021).

  [2] **Kasaei, Mohammadreza**, et al. "A modular framework to generate robust biped locomotion: from planning to control." SN Applied Sciences 3.9 (2021): 1-18.

- **Under-review**

  [1] **Kasaei, Mohammadreza**, et al. "A Hybrid Biped Stabilizer System Based on Analytical Control and Learning of Symmetrical Residual Physics." arXiv preprint arXiv:2011.13798 (2020).

  [2] **Kasaei, Mohammadreza**, et al. "A CPG-Based Agile and Versatile Locomotion Framework Using Proximal Symmetry Loss." arXiv preprint arXiv:2103.00928 (2021).

  [3] **Kasaei, Mohammadreza**, Nuno Lau, and Artur Pereira. "A Hierarchical Framework to Generate Robust Biped Locomotion Based on Divergent Component of Motion." arXiv preprint arXiv:1911.07505 (2019).

- **Conferences**

  [1] **Kasaei, Mohammadreza**, et al. "A Robust Model-Based Biped Locomotion Framework Based on Three-Mass Model: From Planning to Control." 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE, 2020.

  [2] **Kasaei, Mohammadreza**, Nuno Lau, and Artur Pereira. "A robust biped locomotion based on linear-quadratic-gaussian controller and divergent component of motion." 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 1429-1434, doi: 10.1109/IROS40897.2019.8967778.

  [3] **Kasaei, Mohammadreza**, Nuno Lau, and Artur Pereira. "A fast and stable omnidirectional walking engine for the NAO humanoid robot." Robot World Cup. Springer, Cham, 2019.

  [4] **Kasaei, Mohammadreza**, Nuno Lau, and Artur Pereira. "A model-based biped walking controller based on divergent component of motion." 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE, 2019.

  [5] **Kasaei, Mohammadreza**, Nuno Lau, and Artur Pereira. "Comparison Study of Well-Known Inverted Pendulum Models for Balance Recovery in Humanoid Robot." MAPiS 2019 - First MAP-i Seminar, (2019).

  [6] **Kasaei, Mohammadreza**, Nuno Lau, and Artur Pereira. "An optimal closed-loop framework to develop stable walking for a humanoid robot." 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE, 2018.

  [7] **Kasaei, Mohammadreza**, et al. "A reliable hierarchical omnidirectional walking engine for a bipedal robot by using the enhanced lip plus flywheel." Human-Centric Robotics: Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines. 2018.

[8] **Kasaei, Mohammadreza**, et al. "A reliable model-based walking engine with push recovery capability." 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE, 2017.

[9] **Kasaei, Mohammadreza**, et al. "A hybrid ZMP-CPG based walk engine for biped robots." Iberian Robotics conference. Springer, Cham, 2017.

[10] **Kasaei, Mohammadreza**, et al. "How to select a suitable action against strong pushes in adult-size humanoid robot: Learning from past experiences." 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE, 2016.

- **Workshops**

[1] **Kasaei, Mohammadreza**, Nuno Lau, and Artur Pereira. "A Model-Based Balance Stabilization System for Biped Robot."2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Chapter 2

Legged Robot Locomotion

Developing a fast and reliable locomotion framework for humanoid robots is a complicated problem due to dealing with a naturally unstable system. In particular, humanoid robots are known as hyper Degree of Freedom (DOF) systems which generally have more than 20 DOFs. According to the similarity in kinematic architecture, bipedal robots are the most appropriate type of robot to operate in humanoid environments. However, developing reliable locomotion is a complex subject that absorbs the attention of researchers. Unlike wheeled robots, humanoid robots can handle real environment limitations like gaps, stairs, uneven terrain, etc. Thus, they can be utilized in a wide range of applications from helping elderly people to performing dangerous tasks like fire fighting.

Over the past decades, several stable walking engines have been presented and tested on real and simulated robots. Although the performances of robots in simulators are not perfectly equal to the real-world experiences, the significant advantage of using realistic simulators instead of real robot experiments is that researchers can perform much experimentation without worrying about damage on the devices of the robot (e.g., wear and tear). The number of researches in the field of developing stable walking framework during recent years shows an increasing interest in investigating humanoid locomotion. In this chapter, we divided the studies in this field into two main groups: *(i) model-based approaches*: these approaches consider an abstract dynamics model of the robot to design and develop the walking framework; *(ii) model-free approaches*: research in this group are biologically inspired and typically focus on generating some rhythmic patterns for the limbs of robots without considering any physical model of the robot. In the remainder of this chapter, these approaches along with their related work will be discussed.

2.1 Related Work

Many model-based and model-free locomotion frameworks have been proposed to provide stable locomotion. A brief literature survey concerning them will be presented in the rest of this section.

2.1.1 Model-Based

Several successful approaches to biped locomotion have been proposed and the Linear Inverted Pendulum Model (LIPM) based method [5] is one of the most used approaches to develop dynamic walking in humanoid robots. In particular, this model considers human motion as a first-order linear dynamic system by linearizing about the vertical plane as well as defining height constraints to the horizontal plane. Indeed, this model tries to approximate the overall dynamics of a bipedal robot using a simple model. The model is composed of a mass that is connected to the ground via a mass-less link. This simplification not only reduces the computational cost for real-time implementation but also provides an appropriate framework to control.

The zero-moment point (ZMP) concept has become one of the most important criteria for analyzing dynamic balance. Conceptually, ZMP is a point on the ground plane where the ground reaction force acts to compensate gravity and inertia. Vukobratovic et al. [6] were the first ones that used ZMP as the main criterion to develop a stable walking for a humanoid robot. Kajita et al. [7] introduced ZMP preview controller for developing stable humanoid walking. In this method, a predefined ZMP trajectory has been used to generate the trajectory of the COM, and a preview controller was used to minimize ZMP tracking error using the jerk of COM. Later, several enhanced versions of LIPM have been proposed [1, 8, 9]. Some of these approaches require considerably more computation, therefore, they can only be applied to robots that have enough computation resources [10].

Shimmyo et al. [11] proposed a model-based locomotion framework by designing a specific dynamics model. Their model was composed of three masses that were on the base and the legs. They used this model to generate the ZMP reference trajectories and developed a preview controller to track the reference trajectories. Their approach has been tested on a real robot and the experiment results showed that it was able to generate stable walking patterns.

Faraji and Ijspeert [1] proposed a dynamics model for a humanoid robot that is composed of three linear pendulums. They showed that by using this model, robots are able to produce a wide range of speeds without requiring off-line optimization and tuning of parameters. As they argued, their approach is fast enough and computationally similar to LIPM.

Young-Jae [12] proposed stable locomotion for humanoid robots. His framework was composed of a controller as well as a gait generator. In his method, LIPM was used to generate the gait trajectories and a feedback controller based on a gyro sensor was adopted for stabilization. His approach has been tested on a real 110cm humanoid robot. The results of the experiments showed the efficacy of the proposed walking engine.

Tedrake et. al. [13] formulated ZMP gait generation and feedback stabilization as a continuous time-varying Linear Quadratic Regulator (LQR) problem. Furthermore, they have derived a closed-form solution for this controller which could be computed in a reasonable amount of time by real-time recomputing the optimal controller. Their approach has been tested on Atlas humanoid robot and the experiment results showed that their approach was able to generate stable walking patterns.

Kuindersma et al. [14] implemented a convex quadratic program (QP) to describe a whole-body dynamic walking controller. They have used an unconstrained model of the walking system and designed a time-varying LQR to compute the optimal cost-to-go for this model and used it as a part of an objective function to compute inputs for the robot. To show the performance of their controller a set of experiments with a simulated Atlas has been carried out and the results showed their controller was able to compute control inputs efficiently.

Feng et al. [15] described a walking and manipulation controller system which was based on a cascade of online optimizations. They estimated a virtual force acting at the COM of a robot and used this force to compensate for the modeling errors of the COM. The performance of their controller has been tested on an Atlas humanoid robot during DARPA Robotics Challenge Finals and they were the only competitive walking humanoid team whose robot always kept its stability and did not fall.

Koch et al. [16] proposed a mathematical trajectory optimization to generate stable walking for humanoid robots. They used a dynamics model of the robot to generate the trajectories of joint as well as actuator torques at the same time using optimal control techniques. They applied their method on a simulated humanoid robot HRP-2 to demonstrate the performance of their method. The simulation results showed that their approach was able to generate walking motion.

Mason et al. [17] proposed a whole-body controller for humanoid robots based on a contact-consistent LQR design. This controller was computationally lightweight and it was able to provide optimal feedback controllers for whole-body coordination by explicitly considering the coupling between the different joints. To show the performance of this controller, several experiments have been carried out using a hydraulically actuated torque-controlled Sarcos humanoid. The experimental results showed this controller was not only able to reject strong impulses (e.g. up to 11.7Ns) but also was able to track squatting trajectories up to 1Hz without re-linearization.

Hong et al. [18] introduced a new walking pattern generator using LIPM and ZMP concepts. Their walking engine consisted of a pole-zero cancellation by series approximation (PZCSA) as a feed-forward controller and also an LQR as a state feedback controller. They illustrated the efficiency of this walking engine using several experiments on a simulated as well as a real humanoid robot. Results showed that their method was able to provide stable and smooth walking.

Andrei et al. [19] proposed a walking motion generator based on Model Predictive Control (MPC). In this work, they have introduced a fully automatic footstep placement with a minimal modification to an LQR controller which was presented in [7]. Simulation results using HRP-2 showed that their approach was able to track the reference velocity while keeping the stability of the robot under control even in the case of strong perturbations.

Although all of the aforementioned approaches can generate walking for humanoid robots, their performances are not the same and depend to several elements. For instance, some of them use more accurate [1, 11] or whole-body dynamics models [17] while the others [12, 20, 21] focus on simplicity and use abstract models. The former approaches are suitable to deploy on robot with powerful computation resources but the latter are good to deploy on robots with limitation resources.

2.1.2 Model-Free Locomotion

Several model-free approaches as a possible alternative to generate humanoid locomotion have been proposed. The CPG-based approaches are one of the important categories in this group.

Picado et al. [22] used some Partial Fourier Series (PFS) oscillators to generate a forward walk for a simulated humanoid robot and used Genetic Algorithm (GA) to optimize the oscillators' parameters. They performed a set of simulation using a simulated Nao robot in the SimSpark[1] simulator and showed the optimized version of their approach was able to achieve a fast forward walking (51cm/s). Later, Shahriar et al. [23] extended their approach and developed a walking engine which was able to generate faster forward walking but also able to generate a stable side walk. It should be noted that several hybrid approaches have been proposed [21, 24, 25] which used the ZMP concept to modify the outputs of the oscillators, consequently providing more stable walking. Some of these approaches try to find an appropriate way to adapt sensory feedback based on classical control approaches. In some other approaches, using learning algorithms, robots learn how to modify the output of each oscillator to track the reference trajectories and compensate errors. For instance, in [26], a framework based on neural networks has been developed to learn a model-free feedback controller to stabilize a quadruped robot while walking on rough terrain.

Missura et al. [27] developed an approach for humanoid robots based on Capture Step Framework (CSF) [28] that complements a CPG gait with balance control. The proposed CSF used an analytical balance controller based on CPG to prevent robot falling and track a commanded step size. They showed a stable walk with disturbance rejection capabilities on a real robot.

Yi et al. [29] performed a reinforcement learning method as online learning on a real humanoid

[1]`http://simspark.sourceforge.net/`

robot. Their approach was on top of an open-loop gait trajectory generator. Their approach learned to optimize the input parameters of three disturbance rejection strategies. They simplified the reinforcement learning algorithm by a discretization of the input space and assumed simple assumptions about the control parameters.

Song et al. [30] developed a CPG-based forward walk generator with a balance controller for a humanoid robot. In their approach, onboard gyro and accelerometer sensors are used to measure the angles of the upper body. Using these angles, they have developed a controller to adjust the tilt angle of the upper body. To verify the effectiveness of their controller, ZMP data has been recorded during walking on different slopes between -7 to 7 degrees. Experimental results on a real NAO humanoid robot showed that their approach can generate stable forward walking on unknown slopes.

Liu et al. [31] designed an adaptive walking engine by combining biological concepts and the traditional control strategies. Their walking engine was composed of a center-of-gravity (COG) trajectory generator and a workspace trajectory modulation process. In their method, an accelerometer along with a low pass filter were used to estimate the body attitude angle. This estimation was then used to adjust the amplitude and the period of the output waveforms. Capabilities of the proposed method have been shown by performing several tests on both real and simulated NAO robots. The experimental results demonstrated an NAO robot able to walk on sloped terrain.

Cristiano et al. [32] implemented a CPG-based walking engine with sensory feedback that enabled biped robots to walk on uneven terrain. To develop a closed-loop walking engine, inertial and force sensors were used. Their system consisted of two stages including trajectory generators (trunk, legs, and arms) and a stability controller to automatically adjust these trajectories using sensor data. To evaluate the capabilities of their walk engine, simulation experiments with an NAO humanoid robot have been carried out. Simulation results showed that the robot was able to walk on stairs with a step height of 0.01m and was also able to walk on sloped terrain.

Nassour et al. [33] presented an extended mathematical model of CPG which they named multilayered multipattern CPG. The proposed walk engine was composed of three layers to generate rhythmic and non-rhythmic motions. In their approach, phase resetting and phase-shifting were used to apply feedback signals to the generated patterns. Several simulations and experiments using an NAO humanoid robot have been performed to validate their CPG model. The results showed that their model was able to produce a 3D walking gait and regain balance after applying small disturbances.

Jing Liu and Oliver Urbann [34] presented and implemented a CPG-based walking engine that was able to model the walking pattern using continuous and differentiable mathematical functions. In addition, they used ZMP criterion and a body posture controller to stabilize the robot during walking. The proposed controller used sensory feedback and tried to modify the walking pattern in real-time. They have performed several experiments on a NAO humanoid robot to verify their method. The results showed their method could provide energy-efficient and stable dynamic walking.

In the rest of this chapter, we will develop a model-based approach followed by a model-free walking framework and perform a set of experiments to assess their performances and to provide a better insight into these approaches.

2.2 Model-Based Locomotion

According to the similarity in kinematic architecture, bipedal robots are the most appropriate type of robot to operate in humanoid environments. Most humanoid robots have more than 20 DoFs, therefore they have complex kinematics and dynamics. Although developing a full-body dynamics model is not impossible, it is computationally too expensive. Therefore, to reduce the computation cost and

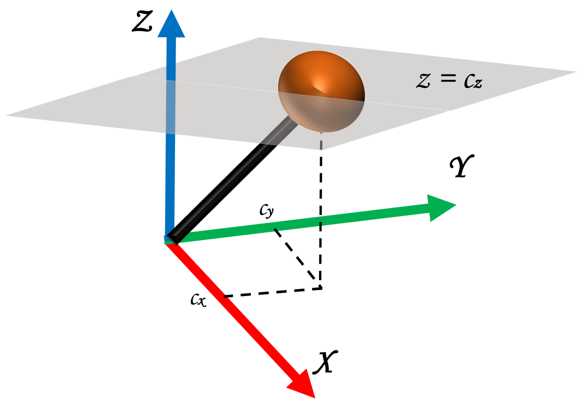


Figure 2.1: 3D-Linear Inverted Pendulum.

complexity, the overall dynamics of biped humanoids are approximated by abstract models. Many dynamic models have been proposed to generate stable gait locomotions, and Three-Dimensional Linear Inverted Pendulum (3D-LIPM) [7] is one of the common dynamics models which is used to analyze the behavior of COM during walking. In this model, the overall dynamics of a humanoid robot is approximated by a single mass that is connected to a certain point on the ground via a massless rod. To achieve simplicity and linearity in dynamics, the single mass is restricted to move along a horizontally defined plane and the momentum about COM is considered to be zero (see Figure 2.1). According to this assumptions, the motion equations of COM are decoupled in the sagittal and frontal plane. The motion equation of this pendulum is as follows:

$$\ddot{c}_x(t) = \frac{g}{c_z} c_x(t) - \frac{\tau(t)}{m c_z}, \tag{2.1}$$

where t represents the time which is reset at the end of each step, m is the mass, g describes the gravity acceleration, c_z is the height of COM and τ is the actuator torque associated with c_x. ZMP is the most commonly used criteria for controlling a biped robot during walking. ZMP concept is used to specify a point on the ground which does not produce any moment (due to gravity and body inertia) in the horizontal direction. The ZMP equation for 3D-LIPM is obtained by the following equation:

$$p_x(t) = \frac{\tau(t)}{mg}, \tag{2.2}$$

where p_x is the location of the ZMP. By plugging (2.1) to (2.2) and reducing, we obtain:

$$p_x(t) = c_x(t) - \frac{c_z}{g} \ddot{c}_x(t), \tag{2.3}$$

where c_z represents the height of COM. Traditionally, using a preplanned ZMP trajectory and solving this differential equation, a static balanced walking can be obtained [7]. Although this approach has been successfully tested on several real robots, it has several drawbacks. This approach requires preplanned footsteps. In addition, to keep the COM at a constant height, knee joints have to be bent during walking, which is not only energy inefficient but also harmful to the knee actuators. Moreover, it does not generate very human-like motion [24, 35, 36].

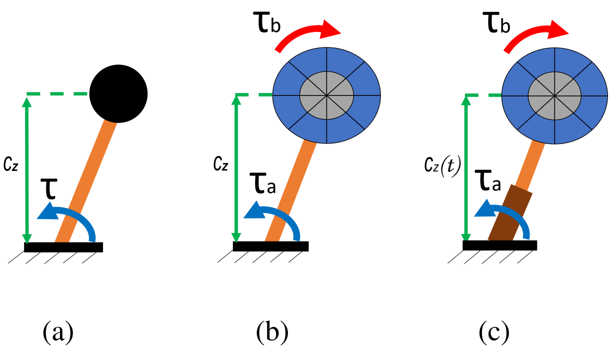


Figure 2.2: Dynamic models of a biped: (a) LIPM; (b) LIPM Plus Flywheel; (c) Enhanced LIPM Plus Flywheel.

2.2.1 Enhanced LIPM Plus Flywheel

The upper part of the body of a humanoid robot has several DOF (i.e., waist, arms, and neck) and their motions generate momentums around the COM [37, 38, 39]. If this effect is considered and a proper method to manage these momentums is not implemented, the robot could not keep its stability and may fall down. To cope with this issue, some extensions to the 3D-LIPM have been proposed that considered the angular momentum around COM [37, 39]. In [37], to model the centroidal angular momentum about COM, a flywheel is used instead of a point mass. The motion equation of the flywheel is as follows:

$$J\ddot{\alpha}(t) = \tau_b(t), \tag{2.4}$$

where J is the rotational inertia of the flywheel, τ_b is the motor torque on the flywheel and α is the flywheel angle. This model called LIPM Plus Flywheel (LIPPFM) which is shown in Figure 2.2 (b). As this figure shows, the dynamics of this model consists of two parts that are (2.1) and (2.4). This model also considers a constant height for the COM. However, to keep the COM at a constant height, knee joints have to be crouched and it is not only harmful to the knee joints but also it consumes more energy during walking. The Z-trajectory of the COM in a human can be modeled as a sinusoidal function that has a variable amplitude depending on the situation [40]. To achieve more human-like motion and more energy-efficient walking, we decided to release this constraint and assumed the height of COM as an arbitrary (but predefined) function of time ($z_l \leq c_z(t) \leq z_h$) (see Figure 2.2 (c)). Therefore, the dynamics of the enhanced model is given by:

$$\begin{cases} \ddot{c}_x(t) = \frac{g+\ddot{c}_z(t)}{c_z(t)} c_x(t) - \frac{\tau(t)}{mc_z(t)} \\ \\ \tau_b(t) = J\ddot{\alpha}(t) \end{cases}, \tag{2.5}$$

where $\tau(t) = \tau_a(t) - \tau_b(t)$ (see Figure 2.2 (c)) and J and α are the rotational inertia and the angle of the flywheel, respectively. The ZMP equation for this model is obtained by the following equation:

$$p_x(t) = \frac{\tau(t)}{m(g+\ddot{c}_z(t))}, \tag{2.6}$$

where p_x is the location of the ZMP. By substituting (2.5) to the (2.6) we obtain:

$$p_x(t) = c_x(t) - \frac{c_z(t)}{g+\ddot{c}_z(t)} \ddot{c}_x(t), \tag{2.7}$$

by solving this differential equation, the trajectory of the COM is obtained:

$$c_x(t) = p_x + \frac{(p_x - c_{x_f})\sinh\big(\omega(t-t_0)\big) + (c_{x_0} - p_x)\sinh\big(\omega(t-t_f)\big)}{\sinh(\omega(t_0-t_f))}, \tag{2.8}$$

where $\omega = \sqrt{\frac{g+\ddot{c}_z(t)}{c_z(t)}}$, c_{x_0} and c_{x_f} are the positions of the COM at the start (t_0) and the end (t_f) of a step, respectively. This model can generate a more natural motion for the COM and upper part dynamics is taken into account. In the next section, we will design a hierarchical walking engine using this model and explain each layer of the hierarchy in detail.

2.2.2 Hierarchical Walking Engine

Biped locomotion can be generated based on the trajectories of ZMP and COM and then calculating the kinematic parameters for the joint angles. To generate these trajectories and develop stable locomotion, we proposed a hierarchical framework that is shown in Figure 2.3. This framework consists of three layers including `Footstep Planner`, `COM Trajectory Generator` and the `Low-Level Controller`. This framework will generate the target joint position for the leg actuators based on the given step parameters. In the remainder of this section, each layer will be described separately.

Footstep Planner

`Footstep Planner` is the first layer and has three main tasks which are (i) planning the position of the next supporting feet according to the input parameters, (ii) generating the ZMP trajectory based on the planned support feet, (iii) generating the trajectory of the swing leg. In our target structure, walking consists of three types of stepping including *stance to walk*, *walking*, and *walking to stop*. Each step consists of two phases, single support, and double support. Each step is defined by Length (L), Width (W), Orientation (θ), Duration (T), and the Ratio of the single support time to the double support time (β). Thus a step is defined as follow:

$$\text{Step} \equiv \{L, W, \theta, T, \beta\}. \tag{2.9}$$

These parameters can be selected based on the size of the robot, capability of the robot, and the tasks that the robot should perform. After the generation of the support foot position, the ZMP

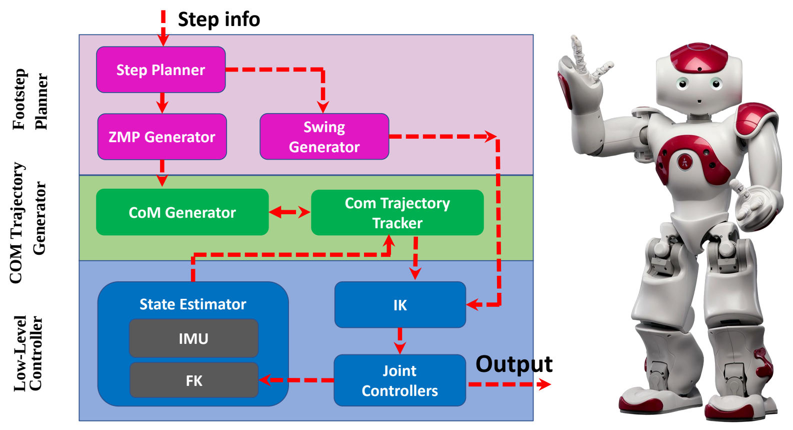


Figure 2.3: Proposed hierarchical architecture.

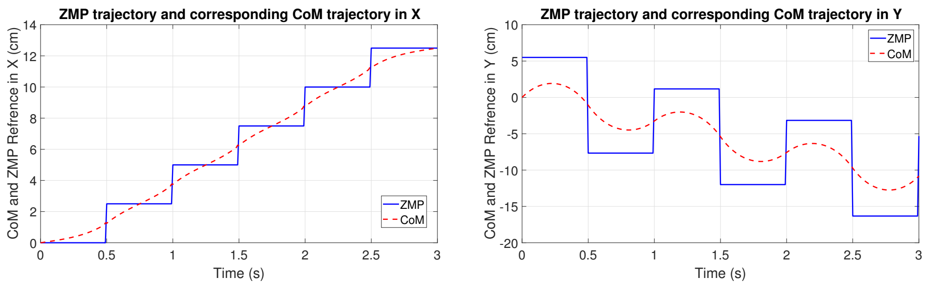


Figure 2.4: ZMP trajectory and corresponding COM trajectory in X and Y.

trajectory should be defined. The best intuitive choice for the ZMP trajectory during the single support phase is the middle of the supporting foot and it moves at constant speed between successive support feet position during the double support phase [41]. In the dynamics model, the swing leg is considered massless but, to reduce the inertia during the single support phase and reduce the effect of ground reaction force, two specific functions are used to control the trajectory of the swing leg for lifting (z_l) and landing (z_d):

$$\begin{cases} z_l(t) = z_s \frac{1-(\cos(\pi t/T))}{2} \\ z_d(t) = z_s \frac{(2\arcsin(t/T))}{\pi} \end{cases}, \tag{2.10}$$

where z_s is the maximum height of lifting. These trajectories cause the swing leg to move fast near to the ground [42].

COM Trajectory Generator

After generating footsteps and related ZMP trajectories, the trajectories for the COM should be generated. The enhanced version of 3D-LIPM plus Flywheel is used to generate them. By substituting the generated ZMP trajectory into 2.8, the trajectory of the COM will be obtained. These trajectories are shown in Figure 2.4. The reference trajectories of the COM, ZMP, and swing leg have already been generated (see Figure 2.5), and using an inverse kinematic method a feedforward loop walking engine can be developed.

Low-Level Controller

This controller is the lowest layer of the proposed structure and consists of four main modules, including state estimator, the forward and the inverse kinematics solvers, and joint position controllers. Position, velocity, and acceleration of the COM are estimated using a state estimator which utilizes the IMU's output and the joint encoder values as inputs. To increase the portability of the proposed walking engine, these modules are kept at the lowest layer. Actually, most humanoid robots have a standard 6 DOF leg with different configurations (3 DOF in the hip, 1 DOF in the knee, and 2 in the ankle). To apply the proposed walking engine to a new humanoid robot, only this layer should be configured.

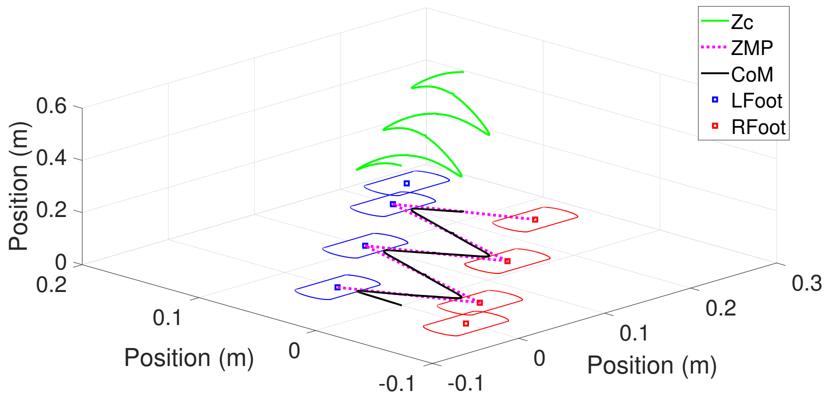


Figure 2.5: Walking reference trajectories.

2.2.3 Experimental Results

To show the performance of the proposed walking engine, an experiment has been set up. In this experiment, the walking engine has been deployed on a humanoid robot named KiKo to perform walking on a standard RoboCup soccer field. Figure 2.6 shows KiKo teen size humanoid robot. It has been built with aluminium brackets. The kinematics chains are powered by servo motors (i.e. Dynamixel EX-106+ and RX-64). The motion mechanism consists of 21 degrees of freedom distributed in six per leg, three per arm, one in the waist, and two degrees of freedom as a pan-tilt system holding the head. KiKo is 1.10m tall and its weight is about 8.5kg. KiKo is equipped with a single-board computer (LP170c) for processing data, an ARM-based (i.e. LPC2368) motion control board, an inertial measurement unit (x-IO) sensor that is installed on the hip, and a webcam (Logitech C905).

In this experiment, KiKo initially stays behind a line. After the start signal is generated, the robot should walk towards the opposite line which is 4m far from the initial line, make a turn there, and walk back. The sequences of this experiment are shown in Figure 2.7. In these experiments, to analyze the stability, IMU data were recorded every 20ms. Figure 2.8 shows the recorded data during walking. It was observed that the proposed walking engine is capable of providing stable walking for the robot. A video of this demonstration is available online at `https://goo.gl/KLRTxd`.

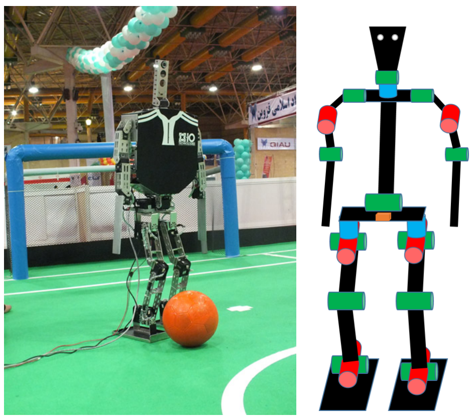


Figure 2.6: The humanoid robot KiKo.

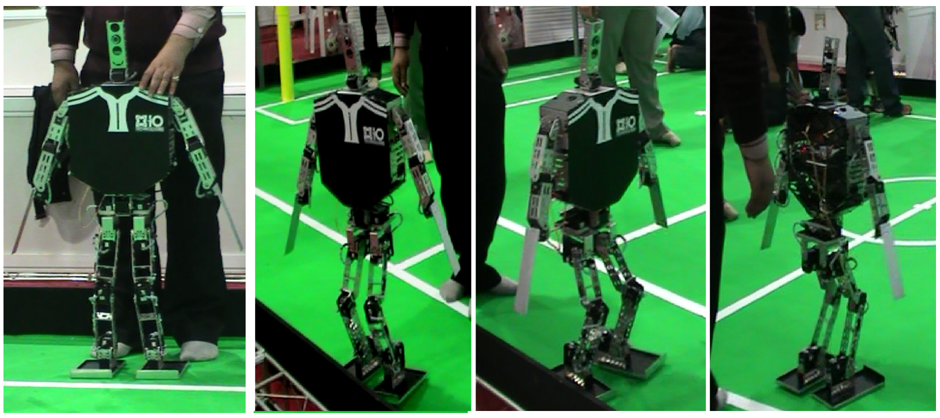


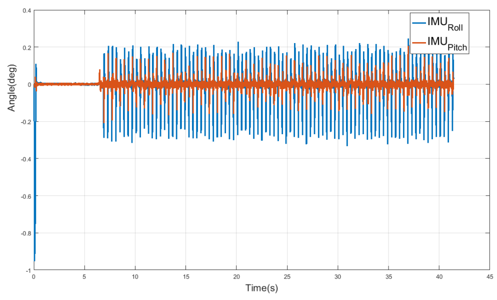
Figure 2.7: Robot snapshots of an experimental test showing the performance of the proposed walking engine.

2.3 Model-Free Locomotion

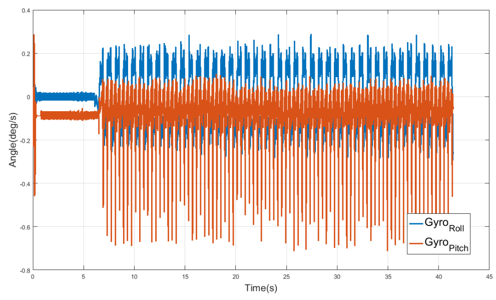
The latest neurophysiological studies on invertebrate and vertebrate animals showed that rhythmic locomotion patterns (e.g. walking, running, etc) are generated by CPGs in the spinal cord [43]. Furthermore, they proved the basic building block of a CPG is an oscillator and a CPG can be modeled by coupling some oscillators in a particular arrangement. Hence, CPGs can generate several endogenously rhythmic signals without rhythmic sensory input. In this section, we intend to develop a fast and stable omnidirectional CPG-based walk engine for a simulated soccer humanoid robot. Towards this goal, Partial Fourier Series (PFS) oscillators are used to generate smooth output trajectories. In order to achieve a more stable walking, sensory feedback signals are used to modify the outputs of the oscillators while interacting with the environment. Additionally, an optimization technique based on Contextual Relative Entropy Policy Search with Covariance Matrix Adaptation (CREPS-CMA) [44] is used to appropriately tune the parameters.

2.3.1 CPG Control Architecture

During the last decades, several CPG-based walk engines have been proposed which were based on different oscillator models such as Partial Fourier Series (PFS), Hopf, Matsuoka, etc. [22, 28, 30]. In most of them, an oscillator is allocated at each limb and its output is directly used as a set point



(a)



(b)

Figure 2.8: Recorded IMU data during walking: (a) represents the hip's orientation; (b) represents the hip's angular velocities.

command (torque, position, etc.). Humanoid robots have several Degrees of Freedom (DOF), therefore adjusting the parameters of each oscillator is not only difficult but also trial-intensive. In addition, adapting sensory information and obtaining appropriate feedback pathways for all the oscillators is a complex subject [30].

In this section, to overcome these difficulties, a specific arrangement for the oscillators is designed which takes into account the dynamics and kinematics model of the robot. Our arrangement is composed of only seven oscillators by considering the symmetry motion pattern between the legs and also between the arms. These oscillators generate the Cartesian coordinate positions of feet and arms to produce overall stride trajectories. To develop a CPG trajectory generator, PFS oscillators are used as the main oscillators since they are able to generate multi-frequency shape signals. A PFS tries to decompose a periodic signal into a sum of simple oscillators (e.g. sines or cosines) and its model is defined as follows:

$$f(t) = A_0 + \sum_{n=1}^{N} A_n \sin(n\omega t + \phi_n), \qquad \omega = \frac{2\pi}{T} \qquad \forall t \in \mathbb{R}, \tag{2.11}$$

where N is the number of frequencies, and A_n , ω and ϕ_n are the amplitude, the angular velocity, and the phase of the n^{th} term, respectively. These parameters can be adjusted to obtain different shapes.

Walking is inherently periodic and according to (2.3), if the trajectory of the ZMP is periodic, then the trajectory of the COM is also periodic. Thus, if we consider trajectories of the COM, $x(t)$, and the ZMP, $p_x(t)$, are generated by PFS oscillators, then we have:

$$p_x(t) = A_0 + \sum_{n=1}^{N} A_n \sin(n\omega t + \phi_n), \qquad \omega = \frac{2\pi}{T} \qquad \forall t \in \mathbb{R}, \tag{2.12}$$

$$c_x(t) = B_0 + \sum_{n=1}^{N} B_n \sin(n\omega t + \phi_n), \qquad \omega = \frac{2\pi}{T} \qquad \forall t \in \mathbb{R}. \tag{2.13}$$

If we consider $\omega_0^2 = \frac{g}{c_z}$ and by plugging (2.12) and (2.13) into (2.3) we obtain:

$$A_0 + \sum_{n=1}^{N} A_n \sin(n\omega t + \phi_n) = B_0 + \sum_{n=1}^{N} (B_n + \frac{B_n n^2 \omega^2}{\omega_0^2}) \sin(n\omega t + \phi_n). \tag{2.14}$$

This equation shows the ZMP trajectories and the COM trajectories have the same form with different amplitudes. Now by comparing the left-hand coefficients with those on the right-hand of (2.14), the relationship between the amplitudes will be obtained:

$$\begin{cases} A_0 = B_0 \\ A_n = B_n(1 + (\frac{n\omega}{\omega_0})^2) \end{cases} . \tag{2.15}$$

The relationships between the amplitudes declare several features for biped walking which are very useful for adjusting the oscillators and walking parameters. For instance, by increasing the ZMP frequency, the amplitudes of the COM will be decreased.

Generally, each step consists of two phases, single support, and double support. To achieve fastest walking speed, we considered the double support period is very short. In our target structure, a step is defined by Length (L), Width (W), Orientation (θ) and Duration (T). According to these parameters, ZMP trajectory can be generated. The best intuitive choice for the ZMP trajectory is the middle of the supporting foot [41]. Hence, ZMP trajectories with respect to the torso location can be generated

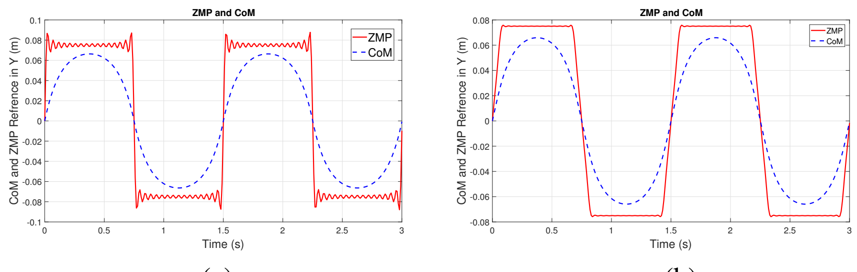


Figure 2.9: ZMP trajectory and corresponding COM trajectory in Y-direction: (a) the outputs of oscillators when the sum is truncated at the $N = 31$; (b) the outputs of oscillators after applying the Lanczos sigma factors.

using two distinct periodic symmetric square functions. Therefore, ZMP trajectories are defined as follow:

$$p_y(t) = \begin{cases} -(D+W) & \text{-T} \le \text{t} < 0 \\ D+W & 0 \le \text{t} < \text{T} \end{cases} \quad , \quad p_x(t) = \begin{cases} -L & \text{-T} \le \text{t} < 0 \\ L & 0 \le \text{t} < \text{T} \end{cases} \quad , \tag{2.16}$$

where D is the distance between the feet. Since p_y and p_x are odd functions, a straight forward method exists to approximate them by a PFS. Using this method, the coefficients of p_y and p_x can be calculated as follows:

$$c_y = \begin{cases} A_0 = 0 \\ A_n = \frac{4(D+W)}{n\pi} & \text{for n odd} \end{cases} \quad , \quad c_x = \begin{cases} A_0 = 0 \\ a_n = \frac{4L}{n\pi} & \text{for n odd} \end{cases} \quad . \tag{2.17}$$

By substituting these coefficients into (2.15), the trajectories of COM can be approximated. An approximation to ZMP and the corresponding COM trajectory in Y direction when the sum is truncated at the $N = 31$ are shown in Figure 2.9 (a). As shown in this figure, the output of n^{th} PFS has large oscillations near the target ZMP. To overcome this issue and to generate a smooth output, the Lanczos sigma factors [45] are employed which are defined as follows:

$$\operatorname{sinc}\left(\frac{n}{m}\right) = \frac{\sin\left(\frac{n\pi}{m}\right)}{\frac{n\pi}{m}}, \tag{2.18}$$

where $m = N + 1$. By multiplying these factors into each terms of PFS, the proper smooth outputs will be achieved. The obtained results are shown in Figure 2.9 (b). In the next section, we will design an omnidirectional walk engine by using this CPG step generator.

2.3.2 Trajectories Generator

As we mentioned in the previous chapter, biped locomotion can be generated based on the trajectories of ZMP and COM. ZMP trajectories can be defined by substituting the parameters of a step into (2.17) to obtain the coefficients for (2.12). Then, by substituting these coefficients into (2.15) and using (2.13) the trajectories of COM can be generated. Walking is a motion with bilateral symmetry, therefore, the same trajectories with a half-period phase shift are generated for the swing leg movements. Moreover, three separate oscillators are allocated to generate rotation (θ), swing leg vertical motion, and arms trajectories for both legs and both arms in a symmetrical manner. All of the reference trajectories of walking have already been generated and by using an inverse kinematic method a feed-forward walking engine can be achieved. Figure 2.10 shows these trajectories concerning the COM position.

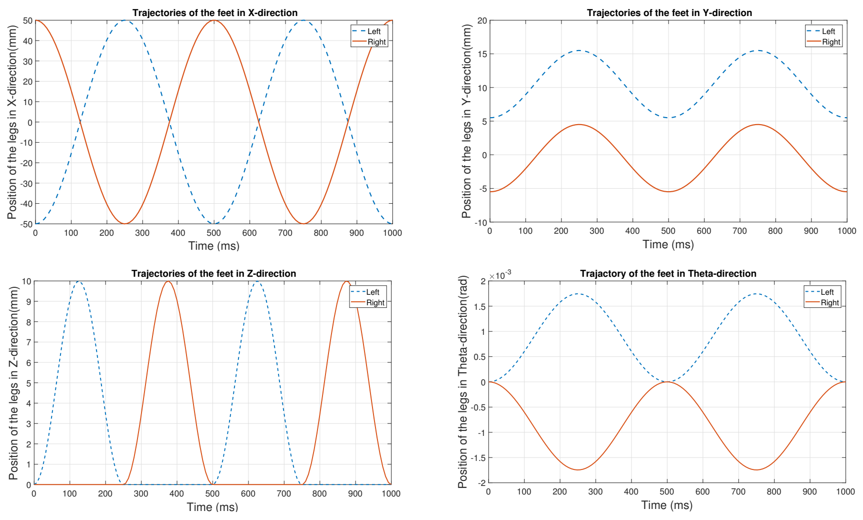


Figure 2.10: Walking reference trajectories with respect to the COM.

2.3.3 Optimization of Walking Engine Parameters

As described in previous sections, the walking engine is parameterized by 7 parameters, representing the expected step movement in the x, y, and z axes, along with the turning and inclination angles, the step duration, and the center of mass height. The parameters are filtered to avoid out-of-bounds values, by using their absolute values when parameters must be positive. This creates a smooth fitness landscape for the algorithm to explore, instead of simply limiting the parameters within a certain range, which creates sharp ridges and flat areas. These parameters have been tuned to obtain both an initial solution that was very stable and that could move forward, albeit slowly, as well as the fastest hand-tuned solution. CREPS-CMA [44] is then used to optimize the slow and stable solution and compared it against the best hand-tuned version.

CREPS-CMA is a state-of-the-art contextual policy search algorithm that iteratively generates sets of candidate solutions sampled from a parameter distribution based on a given context. Candidate solutions are evaluated against a given task and context, and the parameter distribution is updated based on the candidates' fitness, thus improving the quality of the samples in the next iteration. Due to CREPS-CMA's parallel nature, we could take advantage of a distributed optimization architecture to achieve a speed-up of 30x, thus enabling us to optimize the parameters in a practical amount of time.

Maximum Speed Scenario

This scenario focuses on optimizing a simulated NAO humanoid in the SimSpark simulator to achieve the highest speed, moving straight forward, without being unstable enough to fall. The agent is allowed to move for 15 seconds, and the cost function f with parameters θ is defined as

$$f(\theta) = -\delta x + \delta y + \varepsilon, \tag{2.19}$$

where δx represents the total distance covered in the x-axis, δy represents the total distance deviated in the y-axis, and ε is 0 if the agent did not fall and 30 otherwise. With this function, the agent is rewarded for moving along the x-axis, also it is penalized for deviating to the sides or falling. After 5000 epochs of training, CREPS-CMA improved upon our initial speed of 0.11m/s, reaching a

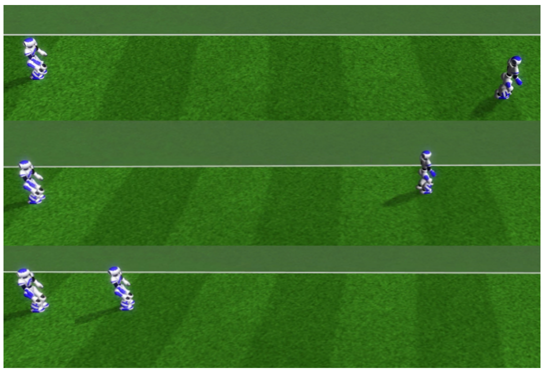


Figure 2.11: The top panel shows the distance covered by the agent after optimization. The middle panel shows the distance covered by our hand-tuned solution. The bottom panel shows the stable initial hand-tuned solution given to CREPS-CMA.

maximum stable velocity of 0.59m/s. In comparison, the hand-tuned best solution achieved only a top speed of 0.50m/s, resulting in an improvement of 18% faster speed. In addition, our optimized agent is faster than both agents in [22] and [23]. The distances covered by the optimized, hand-tuned, and initial solutions are shown in Figure 2.11.

Omnidirectional Scenario

In our second scenario, we focused on optimizing the humanoid agent to achieve high mobility and turning speed, without being unstable enough to fall. We allow the agent to start moving with maximum speed forwards for a random amount of time (from 1 to 3 seconds), this way assuring our parameters allow the agent to turn when the agent is already moving. The agent is then allowed to turn to a specific angle for 10 seconds, or less if he starts moving away from the target. We defined the angle as a random value in the interval $[0,45]$deg, and the agent is given a target 1 unit of distance away in that direction. Using a positive interval instead of a symmetrical one is a simple trick to speed up optimization, which we exploit by negating the y and θ parameters when, in actual play, the target stands in a direction with a negative angle. Cost function f with parameters θ is defined as

$$f(\theta) = \delta d \frac{20}{3} + \delta\theta \frac{10}{180} + \delta t + \varepsilon, \tag{2.20}$$

where δd represents the total distance to the intended target, $\delta\theta$ represents the difference between the final and target orientations of the agent, δt represents the time taken, and ε is 0 if the agent did not fall and 30 otherwise. With this function, we reward the agent for moving as fast as possible and with as much accuracy as possible to the target pose, without falling. We use some ratios to scale each segment of the reward, such that the distance, angle, and time have a similar importance penalty-wise. After 2000 epochs of training, we achieve a fast omnidirectional walk, where the agent can turn up to 45deg in a single unit of distance, as can be seen in Figure 2.12.

2.4 Summary

In this chapter, we first discussed the characteristics of humanoid robots and explained why they are the most appropriate type of robots to operate in our daily-life environments and what are the main difficulties in developing legged robot locomotion. We then investigated the literature to figure out existing approaches. Furthermore, we divided the studies in the literature into two main categories:

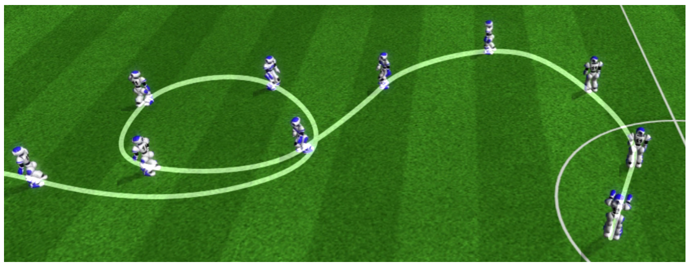


Figure 2.12: An exemplary omnidirectional walk after the optimization process has finished. The agent is given several turning targets, starting with left turns and finishing with right ones, all of these with increasing angle. The agent walks forward for 2 seconds after every turn.

(i) model-based and *(ii)* model-free. Afterwards, we developed model-based as well as model-free walk engines and performed a set of experiments to validate their performances and to provide a better insight into the presented approaches. Besides, to achieve more human-like motion and more energy-efficient walking, we enhanced LIPPFM by releasing its restriction in the vertical motion of COM. According to this dynamics model, we developed a hierarchical framework to generate omnidirectional walking. This framework decouples the walking motion into three hierarchy layers to increase flexibility and portability. To achieve a highly portable walk engine, all modules which depend on the robot's platform are developed at the lowest layer. We then deployed the framework on a real humanoid robot and performed an experiment to validate the performance of the walking engine. Moreover, we developed a CPG-based walk engine based on LIPM and PFS oscillators. The parametric nature of this walk engine allows using a policy search algorithm to find the optimal parameters for the walking. This walk engine has been deployed and completely tested on a simulated NAO humanoid robot. In the next chapter, we will focus on push recovery in humanoid robots.

Chapter 3

Push Recovery

Nowadays, the number of researches in the field of humanoid robots is increasing and one of the common targets of these researches is realizing a humanoid robot that can operate in our dynamic daily-life environments with the same skill as humans. Due to their similarity in kinematics as well as dynamics to a human, they can be utilized in a wide range of applications and not limited to our daily life tasks, also they can be used in many other applications such as rescue missions and helping incapable people. One of the essential requirements for utilizing humanoid robots in such environments is the ability to perform tasks safely. A reliable stabilization method for humanoid robots' motion is a key to let them safely arrive in social environments and collaborate with humans in performing physical tasks. Generally, humanoid robots have more than 20 DOFs and have complex dynamics as well as kinematics. In particular, they have high dimensional state space systems, hence motion planning and control for this type of robot are complex subjects that are still one of the active topics in the robotic community.

In the previous chapter, we presented a feed-forward walking engine capable of generating walking only on flat surfaces or predefined inclined slopes. That walking engine generates rhythmic gait for given inputs regardless of the environment or of the robot state. Therefore, due to unexpected errors which can be raised from several sources such as external disturbances, it is not a reliable walking. For instance, during walking on rough terrain or once the robot collides with an obstacle, several unpredicted nonlinear forces will be applied to the robots. Hence, to operate in these environments, robots should be able to do some recovery strategies to keep their stability. In similar situations, the human uses three distinctive actions including ankle, hip, and step recovery strategies for providing chances to regain balance [36].

In model-based approaches, a dynamics model of a robot is employed to generate reference trajectories and develop controllers. To model the dynamics of robots, two perspectives exist. In the first point of view, the whole body dynamics (true model) of a robot is considered, while in the second point of view, the overall dynamics of the robot is approximated by a simplified model along with a set of assumptions. Although many successful approaches have been proposed based on both perspectives, selecting the appropriate perspective needs to consider a trade-off. For instance, a true dynamics model can (not always) provide more accurate results, but these methods are not only computationally expensive but also their results are platform-dependent.

In the previous chapter, we proposed an enhanced version of LIPPFM by releasing the height constraint of the COM, and we developed a feed-forward walking engine and showed how this enhancement allows generating a more human-like and more stable walking. In this chapter, we will improve the enhanced model by considering the mass of the stance leg to have a more accurate dy-

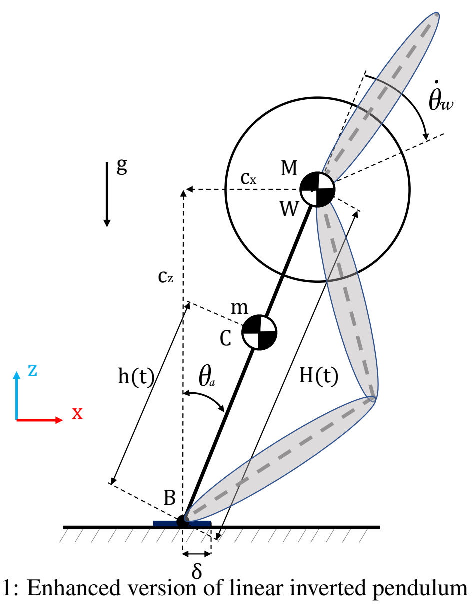


Figure 3.1: Enhanced version of linear inverted pendulum plus flywheel model.

namics model. Then, we will update our walking engine presented in the previous chapter to provide a closed-loop walking based on this dynamics model. Moreover, we will review five well-known dynamics models based on inverted pendulum (IP) that are commonly used in the literature. Afterwards, we will design and perform a set of numerical simulations using MATLAB to compare and evaluate the performance of the proposed dynamics model with the presented well-known dynamics models. Besides, we will deploy our walking engine on a real humanoid robot and perform a set of experiments to show its performance in walking and push recovery scenarios.

3.1 Enhanced LIPM Plus Flywheel

To have a more accurate model, we enhance the LIPM plus flywheel dynamics model by considering the mass of the pendulum. The enhanced version of this dynamics model is depicted in Figure 3.1 and we called it ELIPPFM. In the remainder of this section, we derive Euler-Lagrange equations for this model and represent this model as a state-space system. The Euler-Lagrange equation of this dynamics model is defined as follows:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = F_\theta, \tag{3.1}$$

where $L = T - P$ is a Lagrangian which is equal to the difference between kinetic (T) and potential energies (P) of the model, $\theta = [\theta_a \quad \theta_w]^\top$ is a vector of pendulum and flywheel angles respecting to the vertical axis, $F_\theta = [\tau_a \quad \tau_w]^\top$ represents a vector of corresponding forces acting in the direction of generalized coordinates θ. Thus, by deriving the Lagrangian for the model and differentiating it by $\dot{\theta}$ and θ, we will obtain:

$$\begin{gathered}\begin{bmatrix}\tau_a\\ \tau_w\end{bmatrix} = \begin{bmatrix}\gamma + I_w & I_w\\ I_w & I_w\end{bmatrix}\begin{bmatrix}\ddot{\theta}_a\\ \ddot{\theta}_w\end{bmatrix} - \begin{bmatrix}\mu(g+\ddot{c}_z)\sin\theta_a\\ 0\end{bmatrix},\\ \begin{cases}\gamma = ML^2 + I_p\\ \mu = ml + ML\end{cases},\end{gathered} \tag{3.2}$$

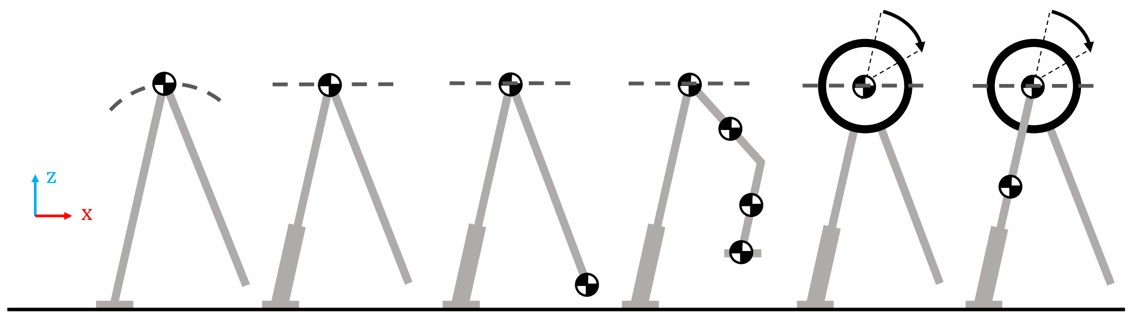


Figure 3.2: Schematics of the simplified dynamics models presented in related work. In these schematics, gray links show the massless links which do not have any effects in the models. Black dashed lines indicate the trajectories of the COM. Name of the models from left to right are IP [46], LIPM [5], TMIPM [47], MMIPM [47], LIPPFM [37], Enhance IPPFM [8].

where L and l are the lengths from base of the pendulum to flywheel center of mass and to pendulum center of mass respectively, g describes the gravity acceleration, I_p is rotational inertia of pendulum about the base of pendulum, I_w represents rotational inertia of flywheel around flywheel center of mass and c_z is the vertical position of the flywheel. By linearizing the model about the vertically upward equilibrium, $\theta_a = 0$, as well as by defining a new state vector, $x = [\theta_a \quad \dot{\theta}_a \quad \dot{\theta}_w]^\top$, the order of model will be reduced. Thus, by substituting x into (3.2) and rearranging in terms of $\dot{x}$, the system can be expressed as follows:

$$\frac{d}{dt}\begin{bmatrix}\theta_a\\ \dot{\theta}_a\\ \dot{\theta}_w\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ \frac{\mu(g+\ddot{c}_z)}{\gamma} & 0 & 0\\ \frac{-\mu(g+\ddot{c}_z)}{\gamma} & 0 & 0\end{bmatrix}\begin{bmatrix}\theta_a\\ \dot{\theta}_a\\ \dot{\theta}_w\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{1}{\gamma} & \frac{-1}{\gamma}\\ \frac{-1}{\gamma} & \frac{\gamma+I_w}{\gamma I_w}\end{bmatrix}\begin{bmatrix}\tau_a\\ \tau_w\end{bmatrix}. \tag{3.3}$$

In the next sections, a set of well-known dynamics models will be presented and briefly reviewed.

3.2 Common Dynamics Models

The basic idea behind using a simplified model is organizing a complex system as a hierarchy. Generally, in hierarchical control approaches, a simplified model is used to determine the overall behaviors of the system abstractly, and then, using a detailed full-body inverse dynamics controller, these behaviors can be converted to individual actuator inputs [1]. It should be noted that the performance of the system depends on the ratio of matching between the template and the exact model.

Several simplified models have been proposed and Inverted Pendulum Model (IPM) [46] is one of those, which is computationally efficient and straightforward to understand. This model describes human dynamics in the single support and provides a low-dimensional and physically accurate model. Kajita and Tani [5] restricted this model by defining a height constraint to the horizontal plane of the system. This simplification not only reduces the computational cost but also provides an appropriate framework to control. Indeed, this constraint causes the dynamics of the system to become a linear dynamic system and hence the model is known as Linear Inverted Pendulum Model (LIPM).

Albert et al. [47] proposed Two Masses Inverted Pendulum Model (TMIPM), which is an extended version of LIPM that considers the mass of the swing leg to increase the gait stability. In their method, the trajectories of COM have been generated using a linear differential equation according to a predefined ZMP and swing leg trajectories. They extended their model [47] by considering the influence of the dynamics of the thigh, the shank, and the foot of the swinging leg. The extended model is composed of four masses and it has been named Multiple Masses Inverted Pendulum Model (MMIPM). Unlike LIPM, TMIPM and MMIPM do not have a direct solution because of the dependency of motions of the masses to each other through the kinematic linkage. Therefore, they proposed an iterative algorithm to define the trajectory of the torso. It should be noted that, in their model, the vertical motion of COM has been restricted to move along a horizontal plane like LIPM.

In all presented models, the upper body has been considered as a single mass, however, the body of a humanoid robot has several DOF (i.e. waist, arms, and neck) and their motions can generate momentum around the COM. As a consequence, if a proper method to manage these momentums is not considered, the robot could not keep its stability and may fall down [8]. To cope with this issue, some extensions to the LIPM have been proposed to consider the angular momentum around COM [37, 39]. In [37], the legs of the robot have been considered massless and extensible. Besides, to model centroidal angular momentum around COM, a flywheel (also called a reaction wheel) has been used instead of a point mass (LIPPFM). According to this model, Pratt et al. [37] proposed the capture point as well as capture region concepts which can be used to answer this question: *when and where to take a step while robot faces a massive magnitude push*? Later, Stephens [38] used this model to determine decision surfaces that could describe when a particular recovery strategy (e.g., ankle, hip, or step) should be used to regain balance.

3.2.1 Gait Stability

Several criteria for analyzing the balance of a humanoid have been proposed and Zero-Momentum Point (ZMP) is one of the well-known approaches. Conceptually, ZMP is a point on the ground plane where the horizontal inertia and gravity forces negate each other [6]. The ZMP concept has been applied in many research on biped locomotion including planning, control, and analysis. In case of no external forces or torques, the ZMP can be defined using the following equation:

$$p_x = \frac{\sum_{i=1}^{k} m_i x_i (\ddot{z}_i + g) - \sum_{i=1}^{k} m_i z_i \ddot{x}_i}{\sum_{i=1}^{k} m_i (\ddot{z}_i + g)}, \tag{3.4}$$

where k represents the number of body parts which is considered in the dynamics model, m_i, x_i, z_i represent the masses and positions of the i_{th} body part. Generally, the trajectories of the masses are generated based on this equation along with a set of predefined trajectories like ZMP and swing leg subject to some constraints. For some of the dynamics models presented in this section, an analytical solution exists to generate these trajectories (e.g., LIPM) which is an important property of a dynamics model because it is not only straightforward but also computationally efficient. In other cases, where a direct solution is not feasible, these trajectories are generated based on some assumptions like considering the COM height trajectory to be a known function of time [14], otherwise, the problem can be formulated as an optimization problem that is generally expensive regarding the computation cost. In the rest of this section, we will explain how (3.4) can be used to develop static and dynamics gait for humanoid robots and obtain the equation of motion.

3.2.2 Static and Dynamic Gait

Humanoid gait can be generated in two different ways — static and dynamic. The static methods correspond to low-speed gaits, while the dynamic methods generate fast and more human-like gaits. Static methods try to keep the COM always inside the support polygon (mostly close to the center) when the robot is taking steps so that the COM and ZMP have almost the same trajectories. Unlike the static methods, in dynamic gaits, the COM is not required to be inside the support polygon and can leave this polygon, but the ZMP must be inside the support polygon. In the dynamic gait, when the robot is walking faster, the COM goes close or even outside the support polygon's boundary. In this approach, the acceleration of the COM is always controlled so that the ZMP stays inside the support polygon. To this end, many dynamic gaits have been developed based on simplified dynamics models capable of generating smooth trajectories for the COM by controlling the jerk. In the rest of this section, the motion equations of some well-known dynamics models will be presented.

Inverted Pendulum Model (IPM)

IPM is one of the widely used dynamics model that approximately describes human dynamics in the single support phase. In this model, all the masses are concentrated at a single mass which rolls over a contact point, established by a massless leg. According to (3.1), the motion equation of IPM can be represented as follows:

$$\tau_a = ml^2\ddot{\theta} - mlg\sin\theta, \tag{3.5}$$

where τ_a is the ankle actuator of the supporting leg which is available intermittently, m represents the total mass and l and θ denote the length and angle of the pendulum with respect to the vertical, respectively. This equation is a nonlinear equation and by linearizing this model about the vertically upward equilibrium ($\sin\theta \approx \theta$), we obtain:

$$\tau_a = ml^2\ddot{\theta} - mlg\theta, \tag{3.6}$$

according to this equation, to generate biped locomotion, τ_a should be composed of feedback and feedforward components. The feedback component makes the dynamic system stable and the feedforward component is determined to track the reference trajectory.

Linear Inverted Pendulum Model (LIPM)

As we explained in Section 2.2, in this model, the vertical motion of COM is restricted and according to (3.4), the motion equation of LIPM can be represented as follows:

$$\ddot{c}_x = \omega^2(c_x - p_x), \tag{3.7}$$

where $\omega = \sqrt{\frac{g}{c_z}}$ represents the natural frequency of the pendulum, c_z is the height of COM, c_x and p_x denote the positions of COM and ZMP, respectively. This equation can also be represented as a state space system:

$$\begin{bmatrix} \dot{c}_x \\ \ddot{c}_x \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \omega^2 & 0 \end{bmatrix} \begin{bmatrix} c_x \\ \dot{c}_x \end{bmatrix} + \begin{bmatrix} 0 \\ -\omega^2 \end{bmatrix} p_x. \tag{3.8}$$

Two Masses Inverted Pendulum Model (TMIPM)

In this model, to have a more accurate dynamics representation, a mass is assumed to residing at the foot of swing leg and the remaining masses of the robot is concentrated in another mass reside at the torso. The dynamics of this model can be represented in state space form using (3.4):

$$\begin{gathered}\begin{bmatrix} \dot{c}_x \\ \ddot{c}_x \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \omega^2 & 0 \end{bmatrix} \begin{bmatrix} c_x \\ \dot{c}_x \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\alpha & 1 \end{bmatrix} \begin{bmatrix} p_x \\ \beta \end{bmatrix} \\ \begin{cases} \alpha = \frac{g}{c_z} + \frac{m_c}{m_s c_z}(\ddot{z}_s + g) \\ \beta = \frac{m_c}{m_s c_z}(x_s(\ddot{z}_s + g) - \ddot{x}_s z_s) \end{cases}, \end{gathered} \tag{3.9}$$

where x_s and z_s are the positions of the swing leg in x and z directions, m_s and m_c represent the mass of swing leg and the remaining masses of the robot, respectively.

Multiple Masses Inverted Pendulum Model (MMIPM)

In this model, multiple masses are considered for modelling the swing leg. The dynamics of the system is represented by the following differential equation:

$$\ddot{c}_x = \omega^2(c_x - p_x) + \underbrace{\sum_{i=1} \frac{m_i}{m_c c_z}\Big((x_i - p_x)(g + \ddot{z}_i) - \ddot{x}_i z_i\Big)}_{f(t)}, \tag{3.10}$$

where c_z is the height of COM, m_i, x_i and z_i represent the mass and the positions of the i_{th} part of the swing leg. As it was explained before, there is no direct solution for this model and, in such situations, the trajectories of COM should be generated using an iterative algorithm. Thus, for generating the COM's trajectories, first, the system is assumed as a TMIPM and generate the trajectories of the COM using the (3.9) and predefined swing leg trajectories. Then, based on a direct kinematic approach, the motion of m_i will be determined and then based on them $f(t)$ will be determined. This procedure will be repeated until an appropriate solution is obtained.

Linear Inverted Pendulum Plus Flywheel (LIPPFM)

This model considers the momentum around the COM and uses a flywheel with centroidal moment of inertia and rotational angle limits to explicitly model centroidal angular momentum. The equations of motion of this model can be represented using the following state space system:

$$\begin{bmatrix} \dot{c}_x \\ \ddot{c}_x \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \omega^2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_x \\ \dot{c}_x \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\omega^2 & -(mL)^{-1} \\ 0 & 0 \\ 0 & I_w^{-1} \end{bmatrix} \begin{bmatrix} p_x \\ \tau_w \end{bmatrix}, \tag{3.11}$$

where m is the mass of flywheel, L, I_w and τ_w represent the length of the pendulum, the rotational inertia of the flywheel around the flywheel center of mass and the flywheel torque, respectively.

3.3 Walking with Push Recovery

In this section, the walking engine presented in the previous chapter will be extended. We will start by explaining our walk state machine followed by a three-dimensional (3D) ZMP planner. Furthermore, three push recovery strategies will be presented and organized as distinct modules in a layer and added to the walking engine (see Figure 3.3). This layer aims at providing recovery capabilities in hazard conditions like stair climbing and walking on uneven terrain.

3.3.1 Walk State Machine

Walking is a periodic locomotion that can be generated by repeating a series of steps and can be modeled using a state machine. To this end, we propose a state machine that is composed of four distinct states. This state machine is depicted in Figure 3.4 and, as is shown in this figure, the states are Idle, Initialize, Single Support, and Double Support. In the Idle state, the robot is standing

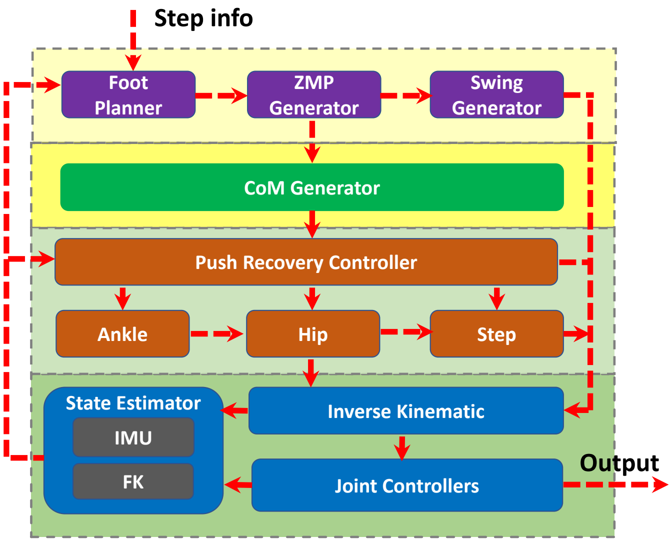


Figure 3.3: Overall architecture of our hierarchical walking engine. It is composed of four layers which are depicted by different colors.

in place, and no walking trajectories are commanded. During the `Initializing` state, the robot is going to be ready to start walking by shifting its COM towards the first support foot. During the `Single Support` as well as the `Double Support` states walking trajectories are generated and commanded. Moreover, a timer is associated with this walking state machine to trigger a state transition. The timer (t) will be reset once it reaches the duration of the double support state (T_{ds}). This state machine along with the hierarchical walking engine presented in the previous chapter can be used to generate dynamic locomotion for biped robots.

3.3.2 3D Footstep Planner

As it is mentioned in the previous chapter, the footstep planner has three main tasks which are (i) generating a set of predefined feet positions (ii) generating the ZMP trajectories, and (iii) generating the trajectories of the swing leg. In the model-based dynamic walking approaches, reference trajectories are generally planned according to the given step information. All of these trajectories should be generated based on the given step info subject to a set of constraints (e.g., maximum step

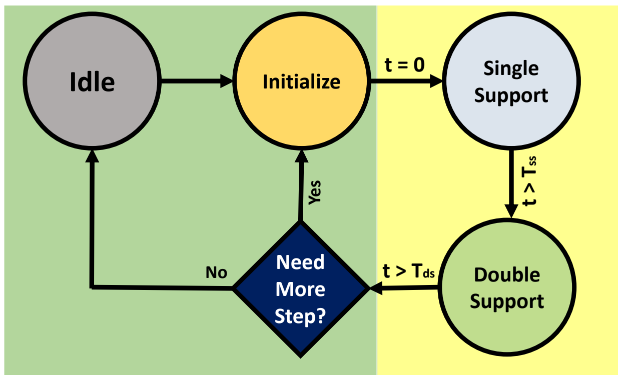


Figure 3.4: Walking state machine with associated timer. Each state has a specific duration.

length, minimum distance between feet, etc.). In our target, each step consists of two phases: single support, and double support, and can be defined as follow:

$$\text{Step} \equiv \{L_{sx}, L_{sy}, T_{ss}, T_{ds}\}, \tag{3.12}$$

where L_{sx}, L_{sy}, T_{ss} and T_{ds} represent step length, step width, single and double support duration, respectively. It should be noted that the step rotation is taken into account implicitly and a 2D rotation matrix is used to perform rotation to update L_{sx} and L_{sy}. These parameters should be selected based on the size of the robot, the capability of the robot, and the tasks that the robot should perform. In the remainder, we explain how these planned footsteps can be used to generate the reference trajectories of walking.

3.3.3 ZMP reference trajectory

The following assumptions are considered for planning the reference ZMP and COM trajectories:

- Robot's feet have a finite-size that equals to 2δ.
- Each step is composed of two individual phases, single support (ss) and double support (ds).
- During the single support phases, ZMP moves from heel to toe (the maximum distance from heel to toe is limited to be $\frac{4}{3}\delta$).
- During the double support phase, ZMP moves proportionally to the COM.
- The pendulum is a massless link and the length of the pendulum is fixed ($m = 0, I_p = 0, \ddot{Z}_c = 0$).

Using these assumptions, reference ZMP generator is formulated as follows:

$$r_{zmp} = \begin{cases} \begin{cases} (f_{i,x} - \frac{L_{ht}}{2}) + \frac{L_{ht}t}{T_{ss}} \\ f_{i,y} \\ f_{i,z} \end{cases} & 0 \leq t < T_{ss} \\ \begin{cases} (f_{i,x} + \frac{L_{ht}}{2}) + \frac{(L_{sx}-L_{ht})(t-T_{ss})}{T_{ds}} \\ f_{i,y} + \frac{(L_{sy})(t-T_{ss})}{T_{ds}} \\ f_{i,z} \end{cases} & T_{ss} \leq t < T_s \end{cases}, \tag{3.13}$$

where t represents the time which is reset at the end of each step ($t \geq T_{ss} + T_{ds}$), T_{ss} and T_{ds} are duration of single and double support phases, respectively. $f_i = [f_{i,x} \quad f_{i,y} \quad f_{i,z}]$ is a set of planned footsteps on a 3D surface ($i \in \mathbb{N}$), L_{sx}, L_{sy} and L_{ht} are the step length, the step width and the length of the heel to the toe, respectively. Thus, by determining these parameters and using (3.13), reference ZMP trajectory can be generated. After planning the footsteps and generating the ZMP trajectories, the swing leg trajectories should be generated. In case of considering a mass-less swing leg, these trajectories can be generated using arbitrary methods (e.g., polynomials, cubic spline, etc.) that reduces the effect of ground reaction force. In other cases, these trajectories should be generated according to the dynamics model of the system. To reduce this effect, we decomposed the swing leg trajectory into landing (i.e. Z_d) and lifting (i.e. Z_l) trajectories to have a smooth trajectory.

$$Z_l(t) = Z_s \frac{1 - (\cos(\pi t / T_s))}{2}, \tag{3.14}$$

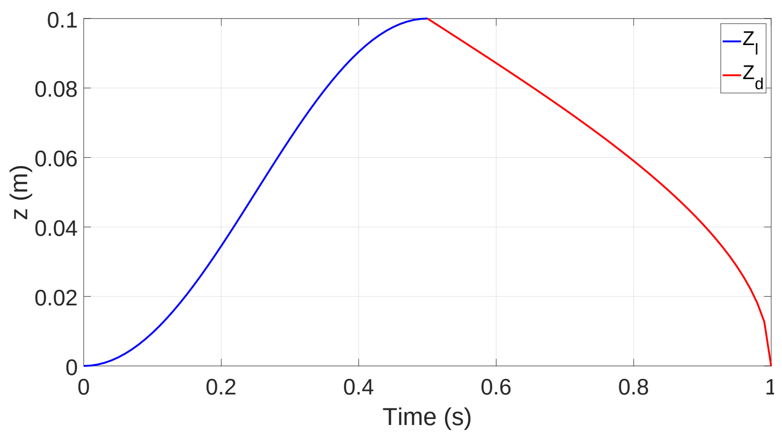


Figure 3.5: Swing leg trajectories for lifting (Z_l) and landing (Z_d).

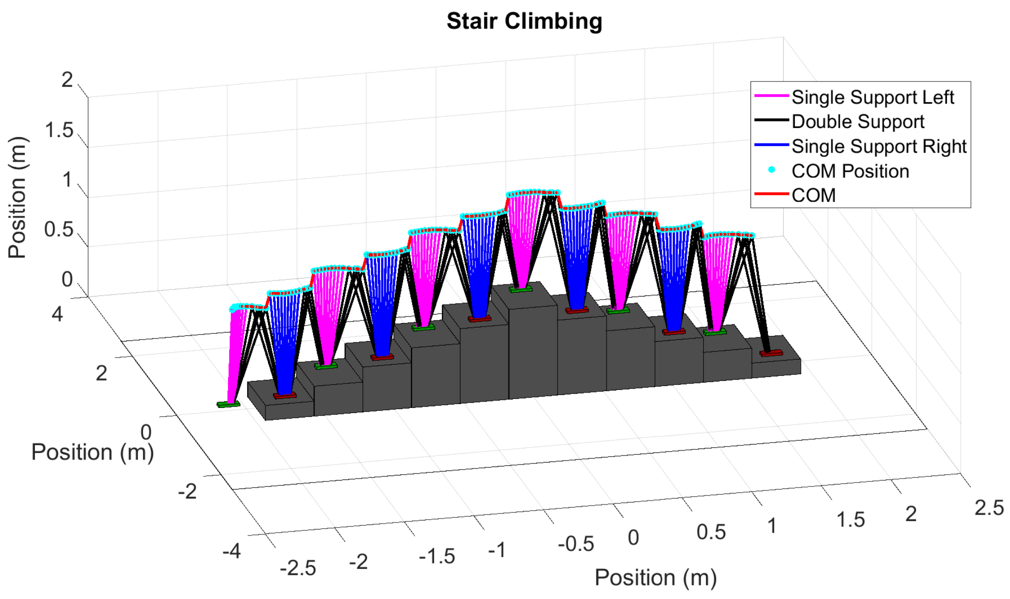


Figure 3.6: ZMP trajectory and corresponding COM trajectory for stair climbing.

$$Z_d(t) = Z_s \frac{(2\arcsin(t/T_s))}{\pi}, \tag{3.15}$$

where Z_s is the the swing leg height. These trajectories cause the swing leg move fast near to the ground. These trajectories are depicted in Figure 3.5. It should be mentioned that a proper actuation pattern for foot rotation to turn around the yaw axis will be determined by inverse kinematics. The trajectory planning method presented in this section is maneuverability powerful as well as computationally efficient. This method is able to generate reference trajectories of the walking over uneven 3D terrain surfaces in every control cycle just by planning two steps in advance. The maneuverability of this method is shown by generating a 3D ZMP and the corresponding COM trajectory over a set of stairs in Figure 3.6.

3.3.4 Push Recovery

To develop stable walking, several criteria have been defined [39, 41] and the most important one is keeping the ZMP inside a polygon that is defined by the foot or feet touching the ground (support polygon). The real position of the ZMP and the positions and velocity of the COM are generally used to estimate the state of the robot. In the rest of this section, we will develop and explain three independent human-inspired strategies to keep stability during walking and regain balance after an external push.

Ankle Strategy

According to (3.7), if the magnitude of the applied forces is small, the stability of the robot can be regained by applying a compensating torque at the ankle joints. This strategy tries to keep the COG inside the support polygon. In this strategy (see Figure 3.7 (a)), to apply more torque to the ankle joints, the arms can also be used. Generally, a PD controller is used to implement this strategy.

$$\theta_c = K_p\Phi + K_d\dot{\Phi}, \tag{3.16}$$

where θ_c is the compensate angle that will be added to the ankle's setpoints, K_p and K_d are the gains and Φ is the angular velocity of COM. K_p and K_d are set by an expert using a set of experiments.

Hip Strategy

During fast walking, while the COM accelerates forward, the Center of Pressure (COP) moves behind the COG. Ankle strategy tries to decelerate the COM and keep the COG inside the support polygon. It causes the COP to move in front of the COG and move towards the boundary of the support foot (see Figure 3.7 (b)). When the ankle strategy results in having the COP at the border of the support polygon, the robot can not regain its stability only by using the ankle strategy. According to (3.11) and (3.3), to control the acceleration of COM, beyond the ankle torque, the flywheel torque can be used. Therefore, to prevent a fall, in addition to the ankle and arm joints, the joints of the waist and hip should be used. Actually, the hip strategy tries to compensate for the disturbance by moving

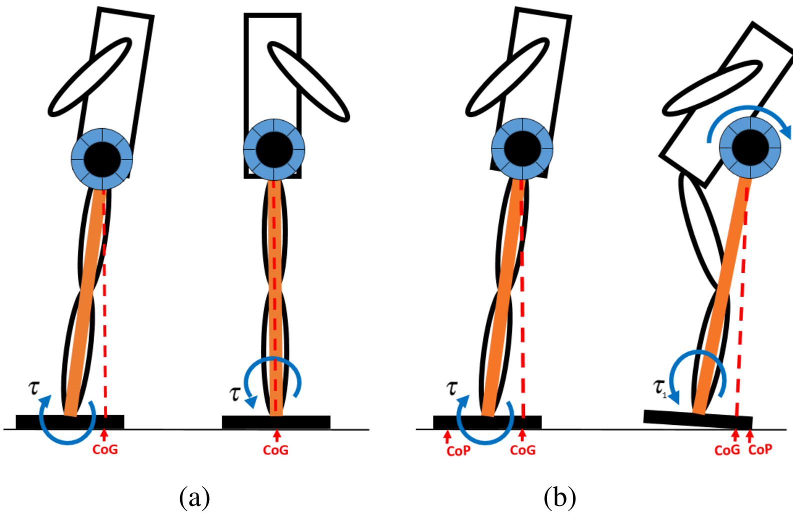


Figure 3.7: Push recovery strategies. (a) Ankle strategy. (b) Hip strategy.

COM and hip quickly. In some humanoid robots [48], a powerful position-controlled actuator is used at the waist to move the torso separately. To implement this strategy, two PD controllers are generally used. One of them is used to apply torque to the hip joints to create the acceleration of the upper body and the other one is used to decelerate the body. The setpoints and gains of the controllers are adequately updated based on the amplitude of the disturbance.

Step Strategy

Once the COG has left the support polygon, similar to a human, the robot should take a step to prevent falling. Taking a step tries to return the COG to the support polygon by moving the base of support. Capture point concept [37] may be used to define where to take a step to regain balance. By considering $\tau_w = 0$ in the (3.11), and using a conserved quantity which is called *Linear Inverted Pendulum Orbital Energy* [5], a mass-spring system with unit mass is obtained. The spring stiffness in this system is equal to $\frac{-g+\ddot{c}_z(t)}{c_z(t)}$. The total energy of this system is determined as follows:

$$E_{LIPM}(t) = \frac{1}{2}\dot{c}_x^2(t) - \frac{g+\ddot{c}_z(t)}{2c_z(t)}c_x^2(t), \tag{3.17}$$

by assuming $\frac{dE_{LIPM}(T)}{dT} = 0$ during walking, the Capture Point (CP) can be calculated by considering $E_{LIPM}(t) = 0$.

$$S_c = \dot{c}_x(t)\sqrt{\frac{c_z(t)}{g+\ddot{c}_z(t)}}. \tag{3.18}$$

Figure 3.8 shows the orbital energy and the corresponding capture point during walking. We have extended our Footstep Planner such that the target foot position can be replaced with the step strategy output. However, this replacement might cause the robot to kick the ground during stepping. To handle this issue, first, the orientation and height of swing and support legs are modified using the orientations of the hip to squarely land the swing leg on the field. Second, we use the compliance functionality provided by the servo motor to reduce the shock of inappropriate landing.

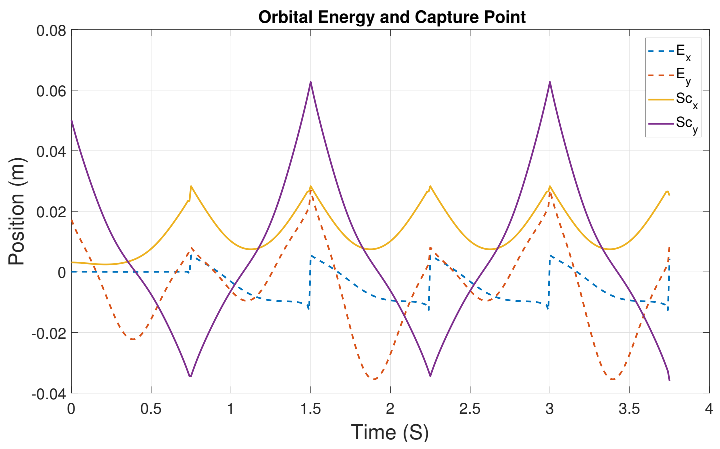


Figure 3.8: Orbital energy and corresponding capture point.

Table 3.1: Parameters used in the simulations.

name	description	value	min	max
m_c	mass (kg)	7.00		
m_1	mass of thigh (kg)	1.50		
m_2	mass of shin (kg)	1.50		
m_3	mass of foot (kg)	0.50		
c_z	height of COM (m)	0.45	0.40	0.50
L_0	length of pendulum (m)	0.50		
L_1	length of thigh (m)	0.28		
L_2	length of shin (m)	0.28		
δ	length of foot (m)	0.10		
τ_w	flywheel torque (N.m)	0.00	-5.00	5.00
$\ddot{c}_z$	Acceleration in Z-direction (m/s^2)	0.00	-0.07	0.07

3.4 Simulation

All the presented models are able to generate walking for a humanoid robot in a general walking scenario, but some of them provide more stable locomotion. To compare the performance of these models and find the best model to generate stable locomotion, a push recovery simulation scenario has been defined. The goal of this scenario is to examine the ability of models concerning regaining balance in challenging situations. In these simulations, the robot is considered in a single support phase and starts from a specified initial condition $(c_{x_0}, \dot{c}_{x_0})$ and the robot should regain its stability without taking a step. A simulation ends when the robot falls ($|c_x| > 0.2$m and $|\dot{c}_x| > 0.5$m/s) or after a maximum period of 5s. According to the results of these simulations, for each model, we can find a specific answer to this question: *"when and which strategies should be used to avoid falling?"*. Moreover, these numerical simulations validate the proposed formulations for each dynamics model. These simulations have been performed using MATLAB, and the most important parameters of the simulated robot as well as their ranges are shown in Table 3.1. In these simulations, for each dynamics model, a set of simulations has been performed according to the set of initial parameters assumed for the simulated robot. For each simulation, the simulated robot is started from single support with a specified initial conditions (position and velocity of COM). The initial condition is selected over the range of [-0.2 0.2] at the interval 0.02m for c_{x_0} and [-0.5 0.5] at interval 0.1m/s for $\dot{c}_{x_0}$ (for each model, 231 simulations were conducted). The simulation results are depicted in Figure 3.9. In these plots, each curve shows the result of a single simulation and yellow regions show the unstable regions which means the robot could not keep its stability and the green regions represent stable regions which means the robot is able to regain its balance. To assess the performance of each model, the results were compared together based on the success rate. A trial is successful if $||c||_2 < 0.02$ where c is the robot's state $(c, \dot{c})$, at the end of trial. The simulation results are summarized in Table 3.2. As the results show, EIPPFM is the ablest model and provides a larger stable region.

Table 3.2: Results comparison for each model.

Model	IPM	LIPM	TMIPM	MMIPM	LIPPFM	EIPPFM
Success rate	23.8%	22.6%	33.3%	37.6%	46.3%	48%

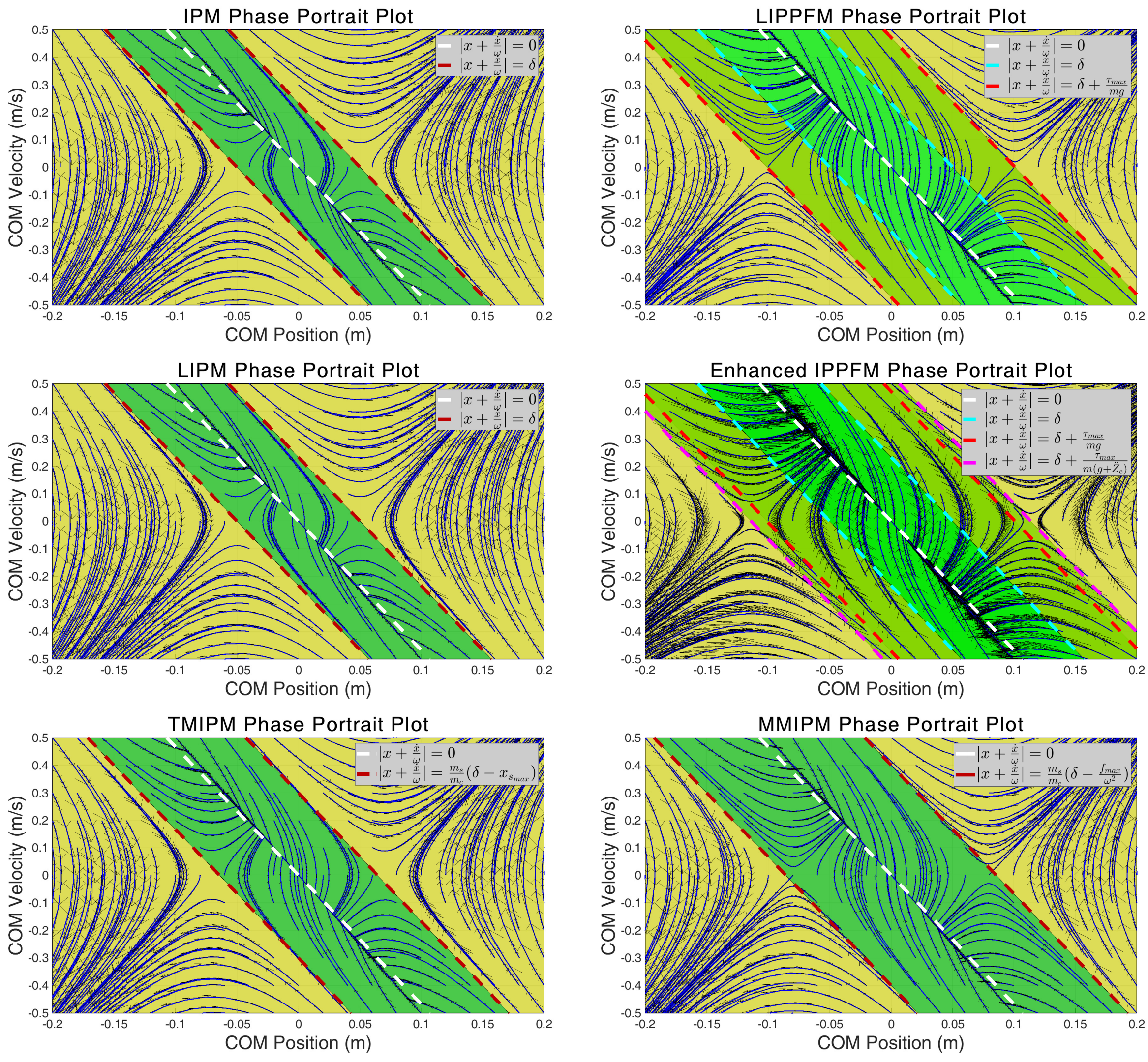


Figure 3.9: Push recovery evaluation results. Yellow regions represent unstable regions where robot should take a step to regain its stability. Green (including light and dark) regions represent the stable regions which mean robot is able to regain its balance. Dash lines show the border of each recovery strategies.

3.5 Experimental Results

To show the performance of the proposed walking engine, a set of experiments was set up, using a real teen-size humanoid robot named Ariana. Ariana has been built with aluminum brackets. The kinematics chains are powered by servo motors. The motion mechanism consists of 21 degrees of freedom distributed in six per leg, three per arm, one in the waist, and two degrees of freedom as a pan-tilt system holding the head. Ariana is 105cm tall and its weight is about 8kg. Ariana is equipped with a single-board computer (LP170c) for processing data, an ARM-based (i.e. LPC2368) motion control board, an inertial measurement unit (x-IO) sensor installed on the hip, and a webcam (Logitech C905).

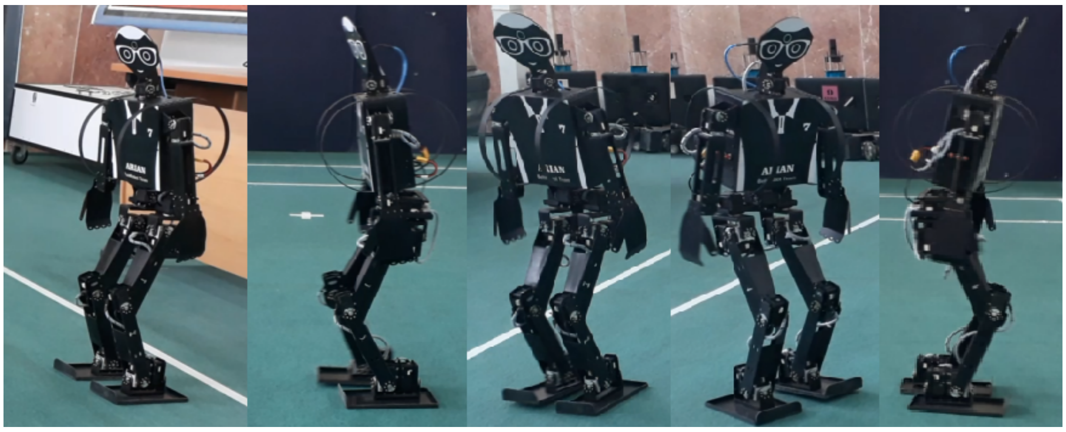


Figure 3.10: The sequences of the experimental test

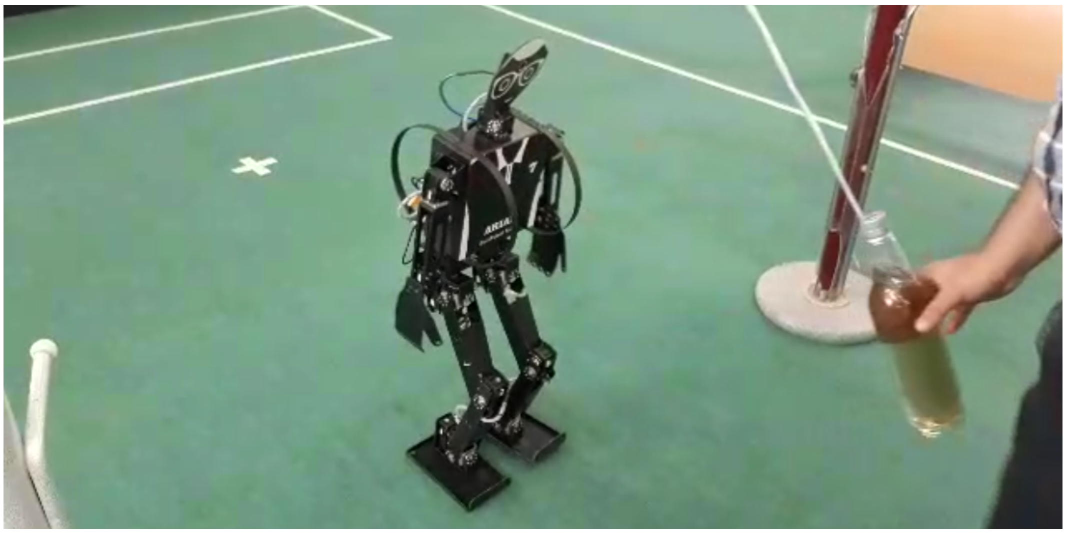


Figure 3.11: Push recovery experimental setup.

3.5.1 Walking

In this experiment, the robot initially stays at a certain position, and, when the start signal is generated, it should walk towards the opposite line which is 4m far from the initial line, make a U-turn there and walk back. The sequences of the experiment are shown in Figure 3.10. In this experiment, it was observed that the proposed walking engine is capable of providing fast and reliable walking for the robot. A video of this demonstration is available online at `https://goo.gl/7JYaqW`.

3.5.2 Push Recovery

To show the performance of the push recovery controller, an experiment was set up to apply nearly repeatable external disturbances for evaluating the withstand of the robot. In this experiment, a bottle with 1.5L capacity has been filled with water so that it is about 1.5 kg. The bottle is initially suspended by a rope and used as a pendulum to apply the push. The length of the rope will remain fixed for all trials. As it is shown in Figure 3.11, in all experiments, the robot initially stays in a certain position and walks in place. During the experiment, the bottle will be moved and released by a human operator

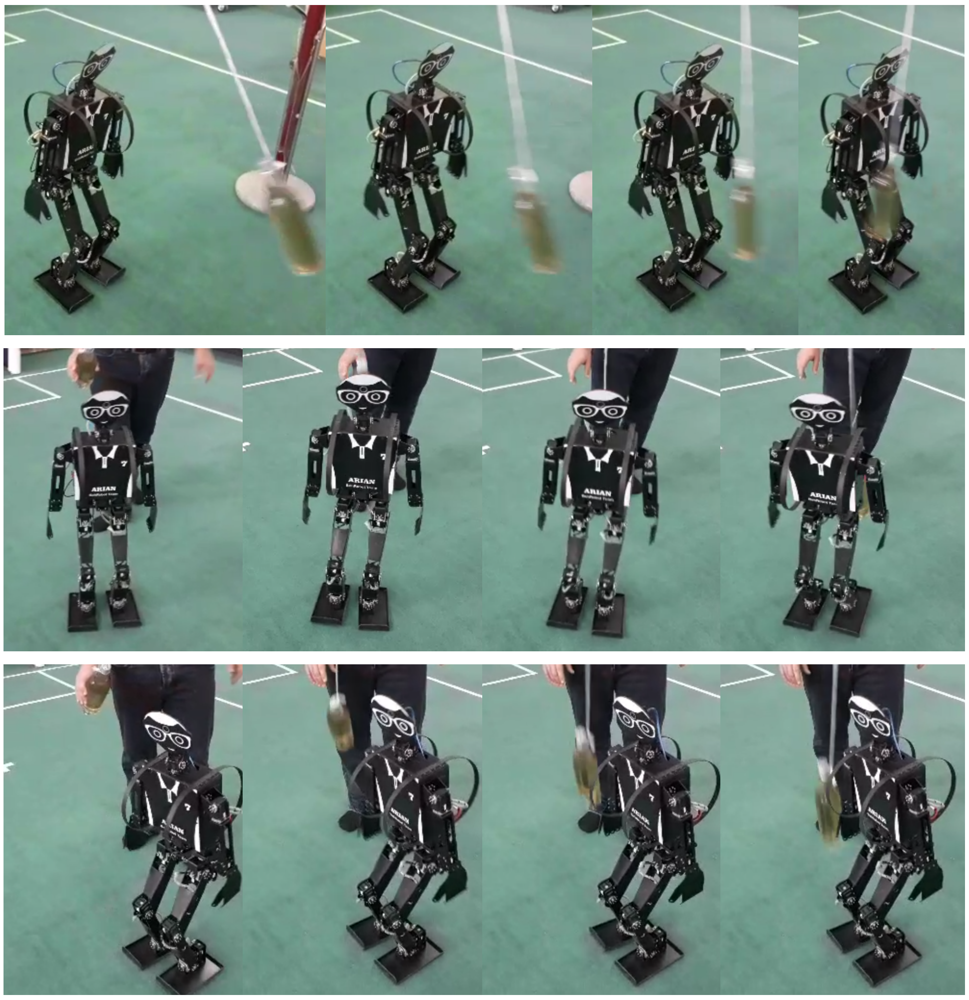


Figure 3.12: Push recovery test: Four snapshots of the forward, the backward and the sideward push recovery tests.

to apply unpredictable forces to the robot's body.

Push recovery evaluation

A push is successfully absorbed when after receiving the push, the robot returns to a stable standing or walking posture. Based on the proposed experimental setup, several push recovery trails were performed from the back, front, and side of the robot. Figure 3.12 shows several snapshots of these trails. Results showed that the system was able to react against strong pushes and could keep the stability of the robot after applying disturbance. A video of this demonstration is available online at `https://goo.gl/45hFL6`.

3.6 Summary

This chapter presented a closed-loop walking engine for a humanoid robot that is capable of providing stable walking with push recovery capability. We firstly extended the LIPM plus flywheel by considering the mass of the stance leg in our dynamics model. Moreover, we briefly reviewed a

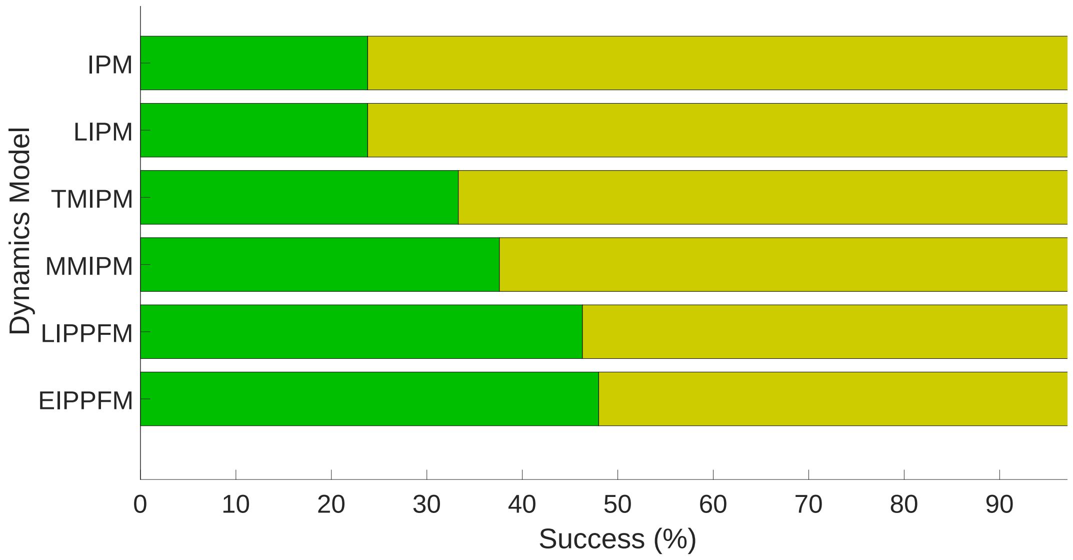


Figure 3.13: Summary of the simulation results in push recovery scenario. According to the results, the EIPPFM is the ablest model regarding keeping the stability of the robot in challenging situations.

set of well-known dynamics models and discussed their properties, followed by a brief explanation of how they generate the walking trajectories according to the input step parameters. Afterward, a 3D footstep planner along with a reference ZMP trajectory generator were presented. Moreover, three push recovery strategies were presented, formulated, and organized as distinct modules in a layer that was added to the walking engine. This layer aims at providing recovery capabilities in hazard conditions like stair climbing and walking on uneven terrain. Besides, an overall architecture of the proposed walk engine was presented to show how the generated trajectories are used to produce walking. We then presented a comparative study of the presented well-known dynamics models for balance recovery in humanoid robots. To validate the formulation and to have a comparison, a set of simulations have been carried out using MATLAB. The summary of the results is depicted in Figure 3.13. As this figure shows, EIPPFM is the ablest model to keep the stability of the robot even in very challenging conditions. Furthermore, the proposed walking engine has been deployed on a real robot, and walking and push recovery experiments have been designed and performed. The experiment results validated the performance of the walking engine. In the next chapter, we will present optimal control schemes for locomotion and focus on developing a set of optimal controllers to improve the performance and robustness of the presented walk engine.

Chapter 4

Optimal Control Schemes for Locomotion

This chapter focuses on designing a set of controllers to provide optimal and robust locomotion for humanoid robots. These controllers deal with finding control signals over a period of time such that an objective function is optimized. More specifically, the overall dynamics of the system will be described by a set of linear differential equations and the cost function will be described by a quadratic function. These controllers try to improve the tracking performance by minimizing undesired deviations and increasing the robustness.

This chapter will be started by presenting some related works along with a brief review of them. Then, the presented walking engine in the previous chapter will be updated by reformulating the controller as a Linear Quadratic Regulator (LQR) followed by a set of simulations including walking on uneven terrain and push recovery to show the performance of this controller. Afterwards, to eliminate the steady-state error, we will propose a Linear Quadratic Integral (LQI) by augmenting the LQR with an integrator. Then, the concept of Divergent Component of Motion (DCM) will be used to establish when and where a robot should take a step to prevent falling. Later, to provide robust locomotion, we will use LIPM and DCM to represent the overall dynamics of a humanoid robot as a state-space system, and, based on this system, we will illustrate how the reference trajectories can be planned and controlled. Indeed, using this system, we will formulate the problem of biped control as a Linear Quadratic Gaussian (LQG) that provides an optimal and robust solution using offline optimization. Then, we will analyze the robustness of the proposed controller performing some simulations regarding three types of uncertainties including measurement errors, unknown external disturbances, and COM height error. Besides, we will augment this controller with an online footstep adjustment controller based on the prediction of the DCM position at the end of each step, and we will show how this online modification increases the withstanding level of the robot against external pushes. Finally, at the end of this chapter, a specific dynamics model will be designed. Based on it, a fast and stable omnidirectional walking engine for the NAO humanoid robot will be developed. The parametric nature of this walking engine allows us to use the Genetic Algorithm (GA) to find the optimum parameters for generating maximum speed.

4.1 Related Work

Several successful approaches to develop biped locomotion have been proposed, and LIP-based methods [1, 5, 9] are the increasingly used approaches for developing dynamic walking in humanoid robots. Recent advances in humanoid robot locomotion have been based on decoupling COM dynamics into divergent and convergent components. Several researches [49, 50, 51, 52] showed that a robust stable walking could be developed just by controlling the divergent component of motion. In the remainder of this section, we briefly review some of them.

Kajita et al. [7] introduced the ZMP preview controller for developing stable humanoid walking. In this method, a predefined ZMP trajectory has been used to generate the trajectory of the COM, and a preview controller was designed to minimize ZMP tracking error using the jerk of COM. Later, a number of enhanced versions of LIPM have been proposed [1, 8, 9] which are computationally the same as LIPM. Unlike these models, some extended versions have been proposed that require considerably more computation and can only be utilized on robots that have enough computation power [10].

Tedrake et al. [13] formulated ZMP gait generation and feedback stabilization as a continuous time-varying LQR problem. Furthermore, they have derived a closed-form solution for this controller which could be computed in a reasonable amount of time for real-time recomputing the optimal controller. Their approach has been tested on the Atlas humanoid robot and the experiment results showed that their approach was able to generate stable walking patterns.

Kuindersma et al. [14] implemented a convex quadratic program (QP) to describe a whole-body dynamic walking controller. They have used an unconstrained model of the walking system and designed a time-varying LQR to compute the optimal cost-to-go for this model and used it as part of an objective function to compute inputs for the robot. To show the performance of their controller a set of experiments with a simulated Atlas has been carried out and the results showed their controller was able to compute control inputs efficiently.

Feng et al. [15] described a walking and manipulation controller system which was based on a cascade of online optimizations. They estimated a virtual force acting at the COM of a robot and used this force to compensate for the modeling errors of the COM. The performance of their controller has been tested on an Atlas humanoid robot during DARPA Robotics Challenge Finals and they were the only competitive walking humanoid team that always keeps its stability and did not fall.

Koch et al. [16] proposed a mathematical trajectory optimization to generate stable walking for humanoid robots. Their method used the dynamic model information of the robot and also efficient optimal control techniques to generate the trajectories of joint as well as actuator torques at the same time. They applied their method on a simulated HRP-2 humanoid robot to demonstrate the performance of their method. The simulation results showed that their approach was able to generate walking motion.

Mason et al. [17] proposed a full dynamic controller for humanoid robots based on a contact-consistent LQR design. This controller is computationally lightweight and it is able to provide optimal feedback controllers for whole-body coordination by explicitly considering the coupling between the different joints. To show the performance of this controller, several experiments have been carried out using a hydraulically actuated torque-controlled Sarcos humanoid. The experimental results showed this controller was not only able to reject strong impulses (e.g. up to 11.7N.s) but also was able to track squatting trajectories up to 1Hz without re-linearization.

Hong et al. [18] introduced a new walking pattern generator using LIPM and ZMP. Their walk engine consists of a pole-zero cancellation by series approximation (PZCSA) as a feed-forward controller and also an LQR as a state feedback controller. They illustrated the efficiency of this walking engine using several experiments on a simulated as well as a real humanoid robot. Results showed that their method was able to provide stable and smooth walking.

Kryczka et al. [53] introduced an algorithm for online planning of bipedal locomotion which was based on a nonlinear optimization technique. Their method tries to find a set of step parameters including step position and step time, which could control the robot from the current state to the desired state. They applied their method to the real humanoid platform COMAN and performed a set of experiments to show how the step time modification could extend the robustness of the controller and improve the ability to recover stability regardless of the disturbance source.

Takaneka et al. [49] were the first ones that proposed the DCM concept and used it to plan and real-time control humanoid walking. They showed that their method is able to generate all the walking trajectories every 5ms which was enough to react robustly against unknown obstacles and unwanted disturbances. Besides, according to their simulation results, they showed that to develop a stable locomotion, just the divergent component needs to be controlled.

Englsberger et al. [54] described the dynamics of LIPM as a state-space system by considering the position of COM and the Capture Point (CP) as the system states. They used a predefined ZMP trajectory as a reference signal, and they showed a robot can perform stabilized walking using their system. In [38] and [8], Pratt's model is used to define decision surfaces for determining when and which particular strategies (e.g., ankle, hip, or step) should be used to regain balance in the case of facing disturbances. Later, Englsberger et al. [50] extended DCM to 3D and introduced Enhanced Centroidal Moment Pivot point (eCMP) and also the Virtual Repellent Point (VRP) which could be used to encode the direction, magnitude, and total forces of the external push. Moreover, they showed how eCMP, VRP, and DCM can be used to generate robust three-dimensional bipedal walking. The capabilities of their approaches have been verified both in simulations and experiments with a real robot.

Khadiv et al. [55] proposed a method based on combining DCM tracking and step adjustment to stabilize biped locomotion. They used LIPM and a set of planned footsteps to generate DCM reference trajectories. Afterwards, they used Hierarchical Inverse Dynamics (HID) to apply the generated trajectories to a real robot. Moreover, they proposed a strategy to adjust the swing foot landing location based on DCM measurement. They carried out some simulations using active and passive ankle simulated robots to demonstrate the effectiveness of their method.

Hopkins et al. [56] introduced a novel framework for dynamic humanoid locomotion on uneven terrain using a time-varying extension to the DCM. They argued that by modifying the natural frequency of the DCM, the generic COM height trajectories could be achieved during stepping. They showed that their framework could generate DCM reference trajectories for dynamic walking on uneven terrain given vertical COM and ZMP trajectories [6]. Their method has been verified in simulation using a simulated humanoid robot.

Another limitation of LIP-based methods is restricting the vertical motion of the COM. To keep the COM at a constant height, knee joints have to be always bent. It is not only harmful to the knee joints but also causes them to consume more energy. These motivate the derivation of methods that release this constraint. Zhao and Sentis [57] introduced a 3D Prismatic Inverted Pendulum dynamics to design a 3D foot placement planning for biped robots. They solved it using numerical integration and showed their method is able to plan foot placement on irregular terrain.

Kamioka et al. [52] analytically derived a solution of DCM for a given arbitrary input function, and, according to this solution, they proposed a novel quadratic programming (QP) to optimize ZMP and foot placements simultaneously. They demonstrated the performance of the proposed algorithm by conducting a push recovery experiment using a real humanoid robot. The results showed that their method could compensate for disturbances in a hierarchical strategy.

In the remainder of this chapter, we will present a set of optimal control schemes and explain how ELIPPFM will be used as the core of these controllers. Furthermore, we will design and execute a set of simulations to assess the performance of the controllers.

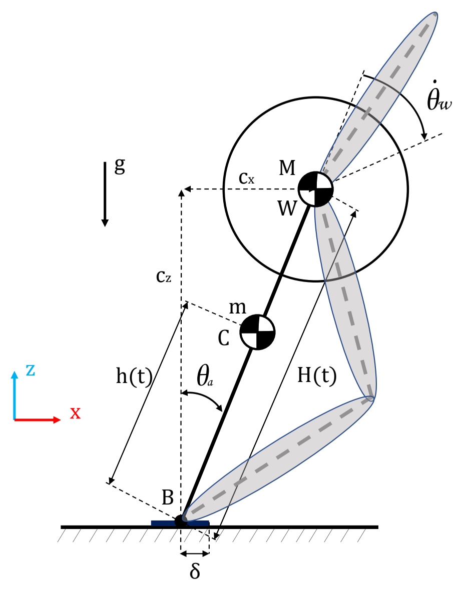


Figure 4.1: Enhanced version of linear inverted pendulum plus flywheel model.

4.2 Linear Quadratic Regulator

As we explained in the previous chapter, the enhanced LIPPFM (ELIPPFM) abstracts the overall dynamics of a humanoid robot into a flywheel that is connected to the ground via a rod. It should be noted that the motions in X and Y directions are equivalent and independent in this dynamics model. Therefore, we just focus on driving the equation for the X-direction only. For convenience, a schematic of this dynamics model is depicted in Figure 4.1 and the equations of motion of this model can be represented as a state-space system:

$$\frac{d}{dt}\begin{bmatrix}\theta_a\\ \dot{\theta}_a\\ \dot{\theta}_w\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ \frac{\mu(g+\ddot{c}_z)}{\gamma} & 0 & 0\\ \frac{-\mu(g+\ddot{c}_z)}{\gamma} & 0 & 0\end{bmatrix}\begin{bmatrix}\theta_a\\ \dot{\theta}_a\\ \dot{\theta}_w\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{1}{\gamma} & \frac{-1}{\gamma}\\ \frac{-1}{\gamma} & \frac{\gamma+I_w}{\gamma I_w}\end{bmatrix}\begin{bmatrix}\tau_a\\ \tau_w\end{bmatrix}$$
$$\begin{cases}\gamma = ML^2 + I_p\\ \mu = ml + ML\end{cases}, \tag{4.1}$$

where L and l are the lengths from base of the pendulum to flywheel center of mass and to pendulum center of mass respectively, g describes the gravity acceleration, I_p is rotational inertia of pendulum about the base of pendulum, I_w represents rotational inertia of flywheel around flywheel center of mass and c_z is the vertical position of the flywheel. $\theta = [\theta_a \quad \dot{\theta}_a \quad \dot{\theta}_w]^\top$ is a vector of pendulum's angle respecting to the vertical axis, its angular velocity and the angular velocity of the flywheel, respectively. $u_t = [\tau_a \quad \tau_w]^\top$ denotes a vector of corresponding torques acting in the direction of generalized coordinates θ. Using this system, an optimal controller for COM tracking can be formulated as follows:

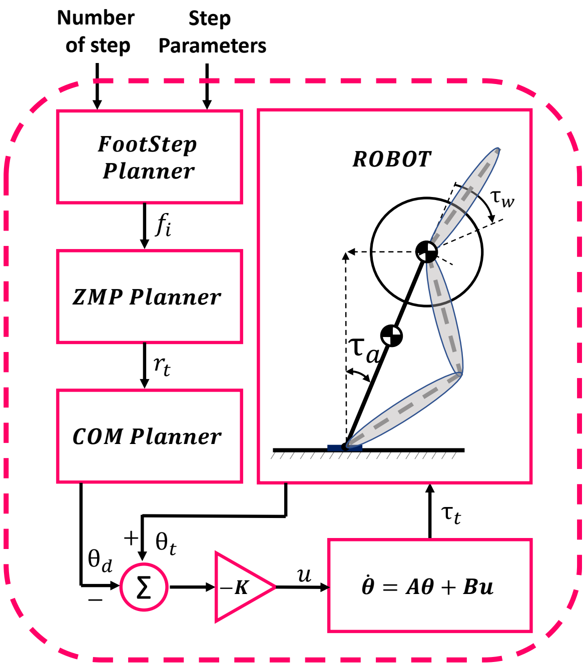


Figure 4.2: Abstract architecture of the proposed LQR controller.

$$
\begin{aligned}
& & \dot{\theta}_t &= A\theta_t + Bu_t \\
&\text{choose } u_t \text{ to minimize} & J_t &= \int_0^{\infty} (\theta_t{}^{\top} Q\theta_t + u_t{}^{\top} Ru_t)dt \\
&\text{subject to} & Q &= Q^{\top} > 0 \\
& & R &= R^{\top} > 0 \\
& & \dot{\theta}_t &= A\theta_t + Bu_t \\
& & \theta_0 &= \theta_{t_0} \\
& & \theta_f &= \theta_{t_f} \quad \forall t \geq t_f
\end{aligned}
, \tag{4.2}
$$

where θ_{t_0} and θ_{t_f} specify boundary conditions at initial and final times, respectively. Q, R represent a trade-off between tracking performance and cost of control effort. A particular control signal to minimize the cost function (J_t) is given by the following:

$$u_t^* = -\underbrace{R^{\top}B^{\top}P}_{K}\theta_t, \tag{4.3}$$

where P is the solution of the following differential equation which is called the Riccati Differential Equation (RDE):

$$-\dot{P} = A^{\top}P + PA - PBR^{-1}B^{\top}P + Q, \tag{4.4}$$

P_t can be easily obtained by standard numerical tools in linear algebra. According to these formulations, the optimal controller with optimal feedback gain K, is implemented as $u_t = -K(\theta_t - \theta_d)$ where θ_d represents the reference trajectory. Thus, by plugging this optimal controller into (4.2), a closed-loop system will be obtained as follows:

$$\dot{\theta}_t = (A - BK)\theta_t + BK\theta_d, \tag{4.5}$$

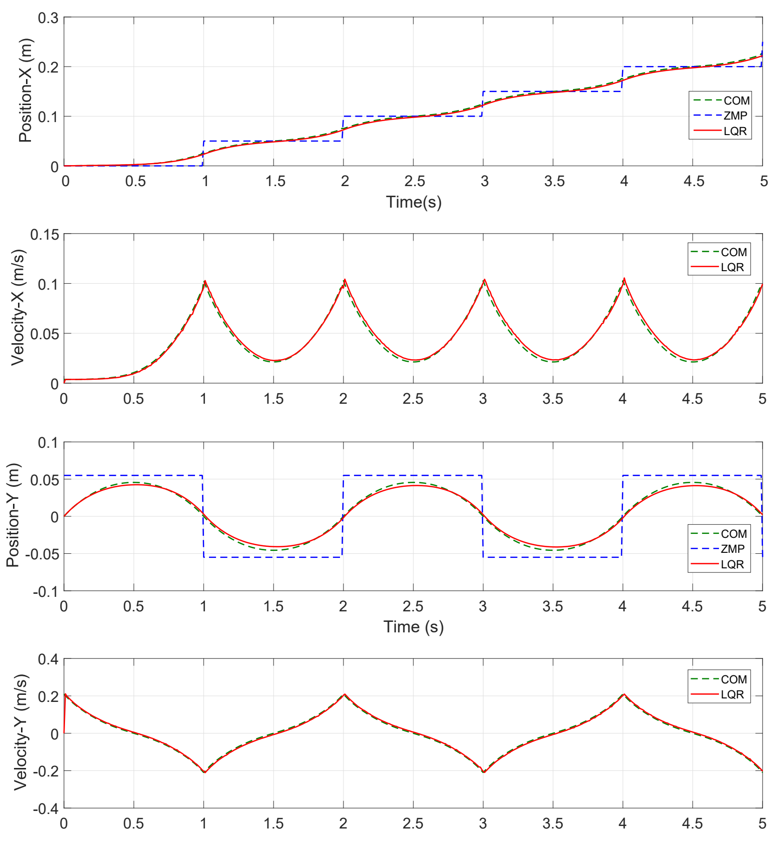


Figure 4.3: Exemplary trajectories for a five-steps forward walking: **dashed-lines** represent the references and **red solid lines** represent the states of the system.

where $\theta_d = [\arctan(\frac{c_x}{c_z}) \quad \frac{1}{1+(\frac{x_x}{x_z})^2} \quad 0]^\top$ represents the vector of desired states and can be obtained using the generated reference trajectories. The quality of this controller depends on the choice of Q as well as R and they generally are selected by trial and error. The overall architecture of this controller is depicted in Figure 4.2.

To check the performance of this controller, an experiment using a simulated robot in MATLAB was performed and the most important parameters of the simulated robot are shown in Table 4.1. In this simulation, a simple example of five-step forward walking with a step duration (SD) of 1s, step length (SL) of 0.05m and step width (SW) of 0.0m has been commanded and the controller should track the reference trajectories. It should be mentioned that we assumed the state of the system is fully observable in each control cycle. The results of this simulation are depicted in Figure 4.3. To demonstrate the performance of the proposed controller, another simulation scenario has been designed and called *Walking on uneven terrains*. The main focus of this simulation is on verifying the ability of the framework to generate stable walking on complex terrain.

Walking on Uneven Terrains

In this scenario, a robot is considered to be on a complex uneven (but known geometry) terrain with $\pm$15cm height variation. The robot is assumed to be able to find a reliable place for foot

Table 4.1: Parameters used in the simulations to check the performance of the proposed LQR controller.

M	m	H	h	τ_a	τ_w	I_w	I_p
3kg	1kg	0.65m	0.35m	10N.m	5N.m	1kg.m^2	1.3kg.m^2

placements and perform diagonal walking on this terrain. To this end, firstly, the target footsteps are planned according to the walking parameters and the given uneven terrain geometry. Then, the reference trajectories including ZMP and COM will be generated (based on the formulation presented in the previous chapters). After generating these trajectories, the proposed LQR controller will be used to track these reference trajectories. Table 4.2 shows the walking parameters which were used in this simulation.

Table 4.2: Walking parameters used to check the performance of the proposed LQR controller when walking in uneven terrains.

δ	T_s	T_{ss}	T_{ds}	L_{sx}	L_{sy}	L_{ht}	Q	R
0.06m	1s	0.7s	0.3s	0.5m	0.35m	0.04m	$10^6 \times I_{3\times3}$	$10^{-3} \times I_{3\times3}$

The simulation results are depicted in Figure 4.4 and as this figure shows, the proposed planner and controller were capable of generating and tracking the walking reference trajectories even in such complex terrains. As the plots at the bottom of this figure show, the controller tracks the reference trajectories.

4.2.1 Linear Quadratic Integral

In this section, to improve the performance of the presented LQR controller, an integral action will be augmented to this controller. This augmentation tries to eliminate the steady state error and improve the tracking performance. The overall architecture of this controller is depicted in Figure 4.5. As shown in this figure, the state-feedback control is defined as follow:

$$u = -K \begin{bmatrix} x - x_{des} \\ x_i \end{bmatrix}, \tag{4.6}$$

where x_{des} represents desired state and x_i the integrator output and K the gain. The value of K is determined as to minimize the following cost function:

$$J(u) = \int_0^\infty \{z^\intercal Q z + u^\intercal R u\} dt, \tag{4.7}$$

where $z = [x \quad x_i]^\intercal$ and Q and R are a trade-off between tracking performance and cost of control effort. Like the LQR controller, the quality of this controller depends on the choice of Q as well as R, and they generally are selected by trial and error. After selecting R and Q, to show the performance of this controller, an experiment has been carried out. This experiment aims at assessing the performance of the tracking. In this experiment, a set of reference trajectories was generated for a single step (*SD*=1s, *SL*=0.2m, *SW*=0.05m) and the controller should track the references. The results of this simulation are depicted in Figure 4.6. The results showed that the controller tracks the references perfectly.

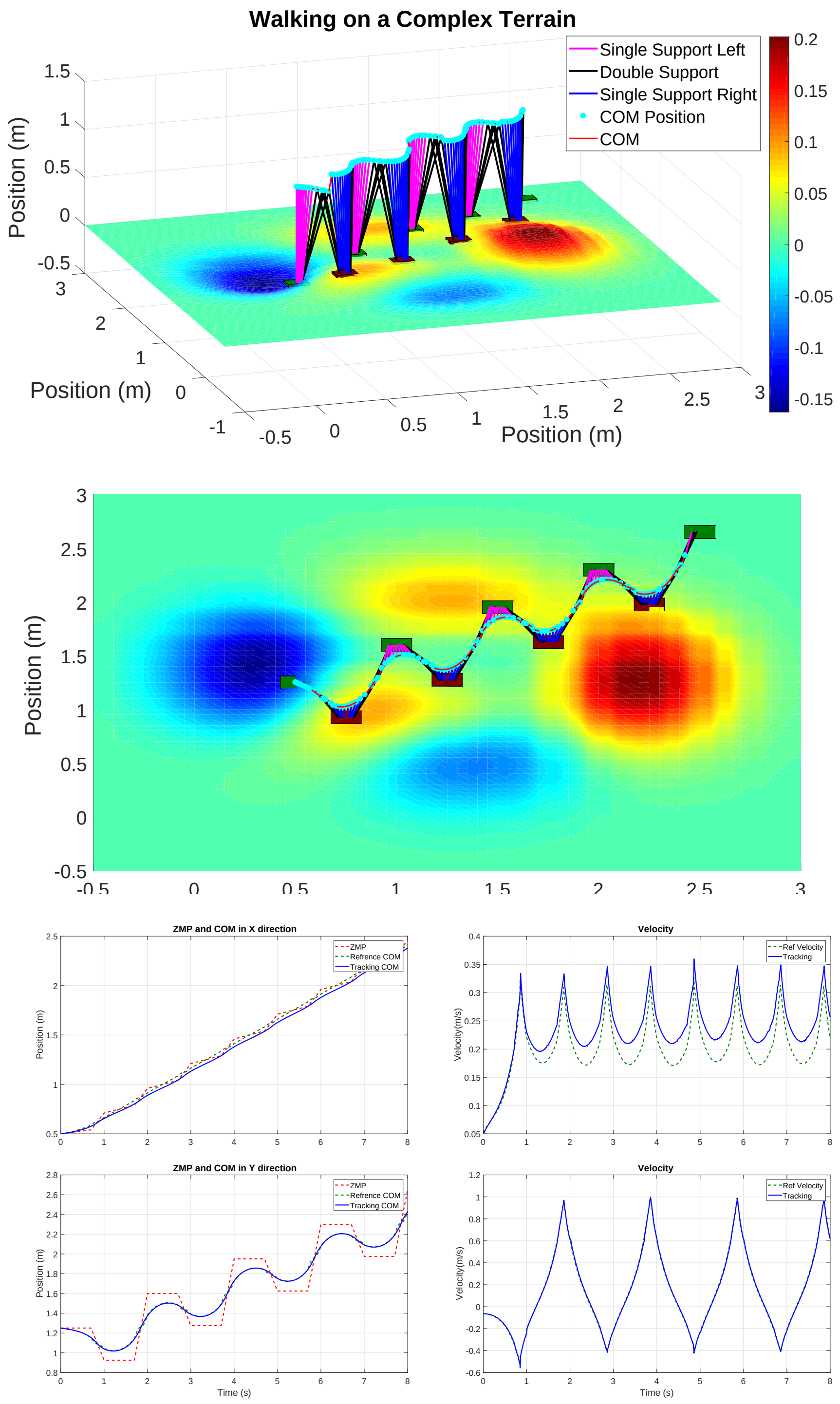


Figure 4.4: Simulation of walking on an uneven terrain. **Top**: the simulated robot is walking diagonally (forward-left); The red and green thin cubes on the ground denote the preplanned feet locations (f_i); **Middle**: top view of the simulation result shows the planned foot steps, ZMP and corresponding COM during walking; **Bottom**: reference position and velocity trajectories and corresponding tracking in X-Y.

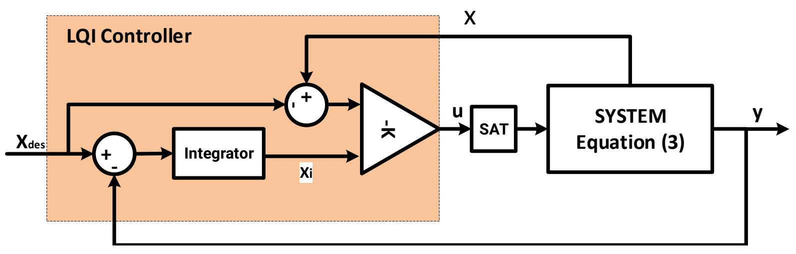


Figure 4.5: Overall architecture of the controller.

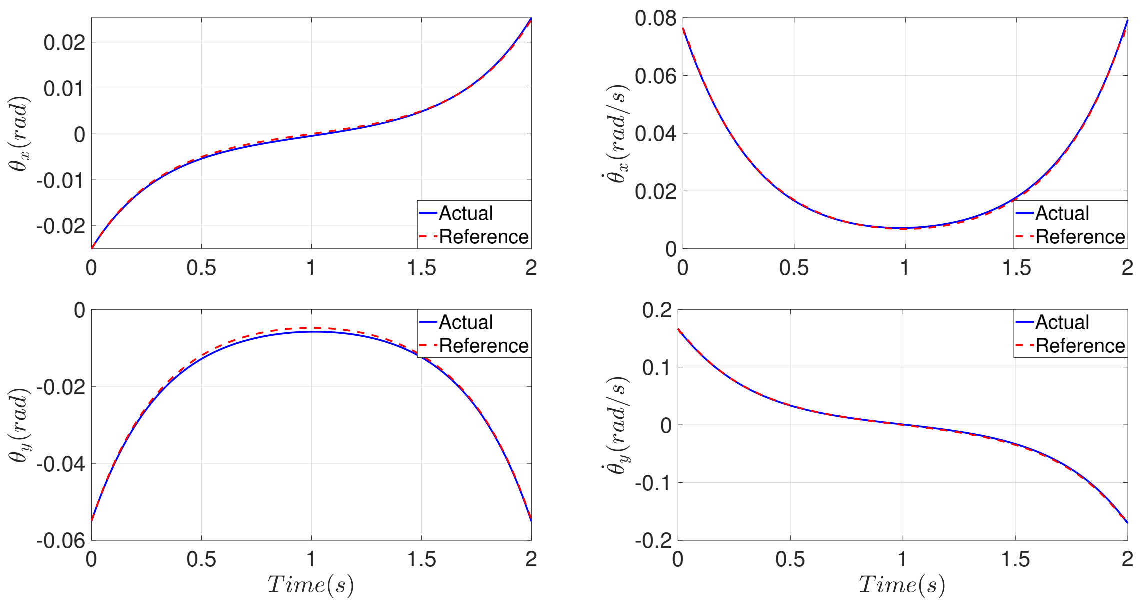


Figure 4.6: Results of reference trajectories tracking in X and Y direction. The dashed-red lines represent the references and the blue lines are the actual states of the system.

4.3 Divergent Component of Motion

Several previous works [48, 49, 58] describe the dynamics of a robot by decomposing it into stable and unstable parts. The unstable part is called Divergent Component of Motion (DCM) and it is a point on which the robot should step to come to rest. This point can be defined as follow:

$$\zeta_x = c_x + \frac{\dot{c}_x}{\omega}, \tag{4.8}$$

where ζ_x and c_x are positions of DCM and COM in X-direction, and $\omega = \sqrt{\frac{g+\ddot{c}_z}{c_z}}$ is the natural frequency of inverted pendulum. As is shown in Figure 4.1, x_c is equal to $H \times \sin(\theta_a)$, θ_a is considered to be small, therefore x_c can be approximated by $H \times \theta_a$. Now, by substituting $x_c = H \times \theta_a$ into (4.8) and by taking derivative from both sides and substituting $\ddot{\theta}_a$ from (4.1) into this equation, $\dot{\zeta}_x$ will be obtained as follow:

$$\dot{\zeta}_x = H\big(\dot{\theta}_a + \frac{\tau_a - \tau_w + \mu(g+\ddot{c}_z)\theta_a}{\omega\gamma}\big). \tag{4.9}$$

According to this equation and the proposed dynamics system (4.1), DCM can be measured using the measurement equation of this system at each time step as follows:

$$
\begin{gathered}
y = \begin{bmatrix} \theta_a \\ \dot{\theta}_a \\ \theta_w \\ \zeta_{mes} \\ \dot{\zeta}_{mes} \end{bmatrix} = Cx + Du \\
C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ H & \frac{H}{\omega} & 0 \\ H & \frac{H \times \mu (g + \ddot{Z}_c)}{\omega \gamma} & 0 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{H}{\omega\gamma} & \frac{-H}{\omega\gamma} \end{bmatrix},
\end{gathered}
\tag{4.10}
$$

where C and D represent the output and the feedthrough matrix, respectively.

4.3.1 Planning the Reference Trajectories

The planning process is started by placing a set of feet positions according to the input step parameters. To simplify the planning process, angular momentum around COM is held at zero and the pendulum is considered as a massless pendulum. Therefore the trajectory of COM can be planed using the analytical solution of LIPM as follow:

$$
c_x(t) = p_x + \frac{(p_x - c_{x_f}) \sinh\big(\omega(t - t_0)\big) + (c_{x_0} - p_x) \sinh\big(\omega(t - t_f)\big)}{\sinh(\omega(t_0 - t_f))}, \tag{4.11}
$$

where t_0, t_f, c_{x_0} and c_{x_f} are the times and corresponding positions of the COM at the start and the end of a step, respectively. p_x denotes the current support foot position which is assumed to be constant (center of current stance foot position) and transits to the next step at the end of each step instantaneously. It should be noted that c_{x_f} is considered to be in the middle of current support foot and next support foot. The corresponding COM velocity trajectory can be calculated using a derivation of (4.11). Now by plugging this trajectory into (4.8), DCM reference trajectory will be obtained and the corresponding DCM velocity trajectory can be determined using a derivation of this trajectory. Figure 4.7 shows the reference trajectories for five steps with a step duration of 2s, stride length of 0.1m and stride width of 0.05m.

4.3.2 Next Step Adjustment

In our proposed structure, the DCM reference trajectory is updated at each control cycle. Indeed, the generated DCM trajectory specifies the desired landing location of the swing leg according to the measured position and velocity of the robot's COM. As is shown in the top plots of Figure 4.7, in case of no disturbance, the landing location of the swing leg is equal to the next preplanned footsteps. In case of small and medium disturbances, the controller presented in the previous section tries to keep the stability of the robot by applying compensating torques at the ankle, the waist and the hips joints (which are modelled as a flywheel in our dynamics model) to prevent falling. In case of significant disturbances, the low-level controller can not keep the stability of the robot just by applying compensating torques because of their saturation, therefore the robot should change its swing leg landing location to prevent falling.

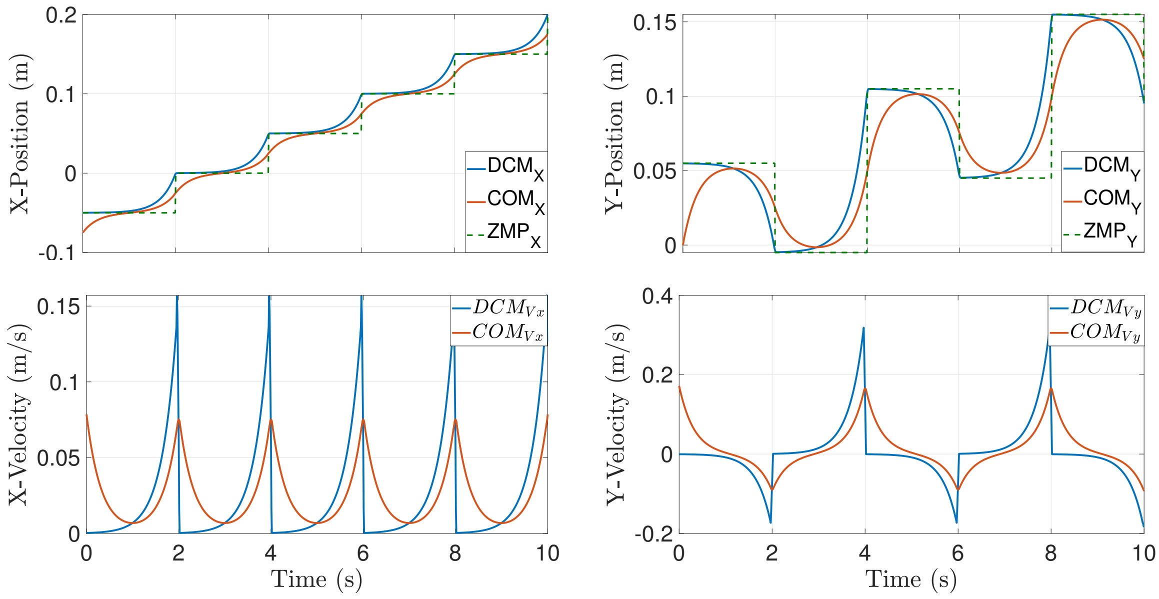


Figure 4.7: An example of generated reference trajectories including ZMP, COM and DCM reference trajectories for a five-steps plan.

Since DCM and its rate of change are measured at each control cycle (4.10), a disturbance can be detected defining two thresholds as follow :

$$\Delta\,\zeta = |\zeta_r - \zeta_x| > thr_1 \quad , \quad \Delta\,\dot{\zeta} = |\dot{\zeta}_r - \dot{\zeta}_x| > thr_2, \tag{4.12}$$

where ζ_r and ζ_x are the reference and actual DCM position at the end of step. It should be noted that thr_1 and thr_2 are defined by trial and error. Once one of these conditions is activated, the robot is considered to be pushed strongly, and the swing leg landing position should be modified as follows:

$$P_{sw_{new}} = P_{sw} + (\zeta_x - \zeta_r), \tag{4.13}$$

where $P_{sw_{new}}$ and P_{sw} represent the new and the planned swing leg positions. Figure 4.8 illustrates this procedure and shows how the swing leg position will be modified once a strong push is applied.

4.4 Simulation Results

To show the performance of the proposed system, a simulation scenario has been designed. This scenario is focused on demonstrating the robustness of the proposed walking system in the presence of disturbances. In this simulation, a humanoid robot is simulated using MATLAB according to the dynamics model presented in Figure 4.1 and all the reference trajectories are generated as described in Section 4.3. The most important parameters of the simulated robot are shown in Table 4.3.

In this simulation, the simulated robot should walk forward with the velocity of 0.2m/s and while the robot is walking, an external perturbation will be applied. This simulation has been repeated three times with different levels of impact to check the capability of the proposed controller. In these simulations, the robot is pushed forward by applying an external force to its COM at $t = 0.6$s and the impact duration is $\Delta t = 0.1$s. The simulation results are depicted in Figure 4.9. In this figure, the results of

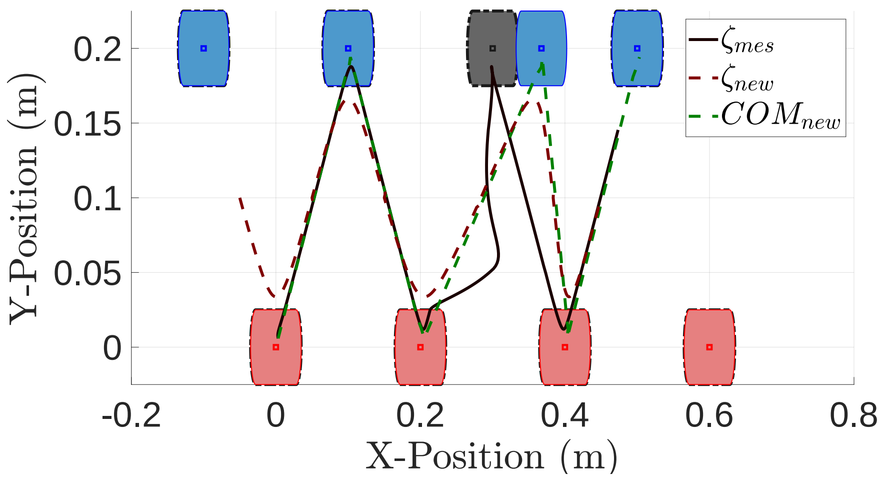


Figure 4.8: An example test of strong push in X direction. The simulated robot is strongly pushed forward at $t = 2.5s$ and the controller modified the landing location of swing leg and re-planned all the reference trajectories.

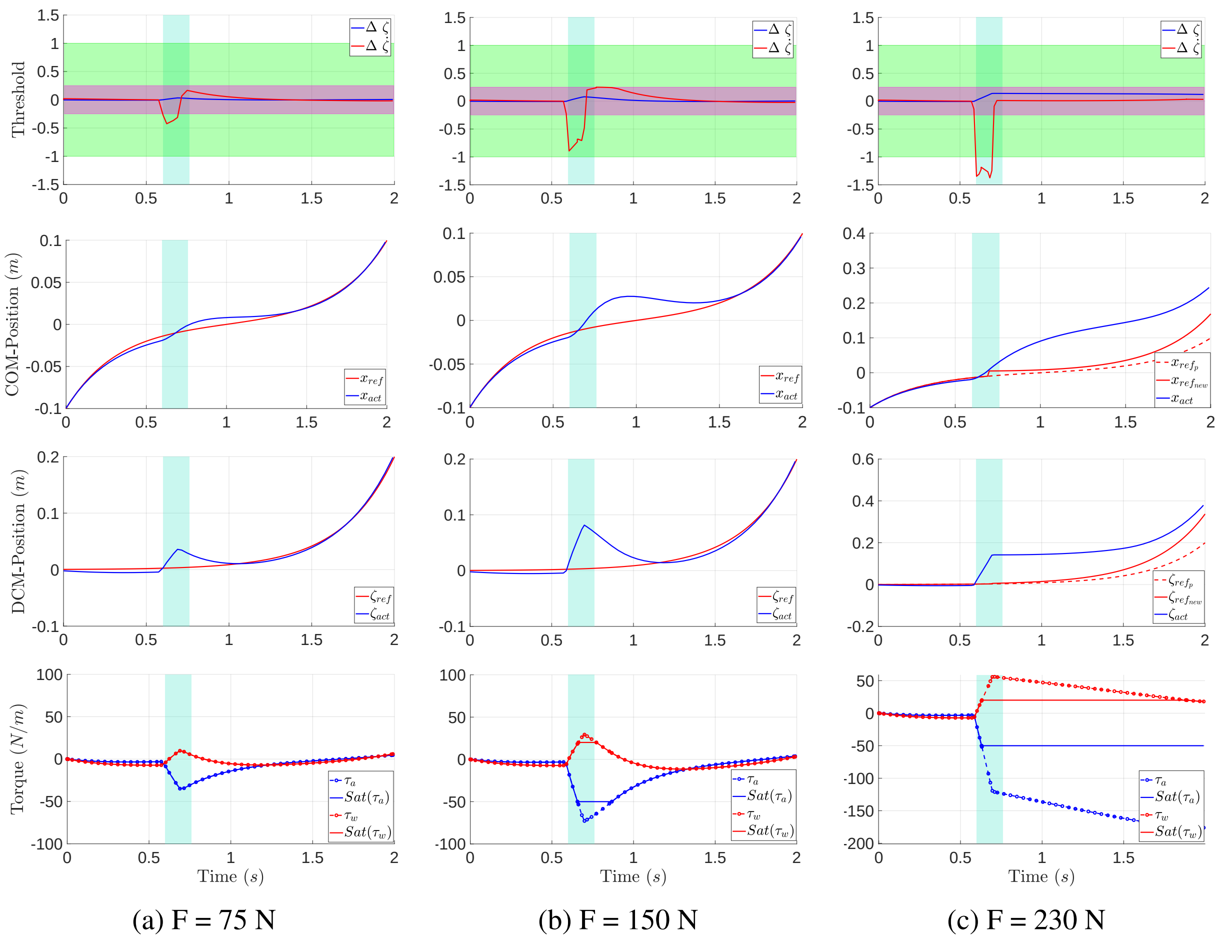


Figure 4.9: The simulation results of walking scenario with three levels of impact. Each column represents the result of a single simulation. In these plots, cyan regions show the impact period, magenta and green regions in threshold plots represent the thresholds for $\Delta\zeta$ and $\Delta\dot{\zeta}$ respectively.

Table 4.3: Parameters used in the simulations.

name	description	value	min	max
m	mass of pendulum (kg)	5.00		
M	mass of flywheel (kg)	50.00		
h	height of stance leg com (m)	0.50		
H	length of pendulum (m)	1.00		
I_p	rotational inertia of pendulum (kg.m^2)	0.12		
I_w	rotational inertia of flywheel (kg.m^2)	25.00		
τ_a	ankle torque (N.m)	0.00	-50.00	50.00
τ_w	flywheel torque (N.m)	0.00	-2.00	20.00
thr_1	threshold 1		0.	0.20
thr_2	threshold 2		-1.00	1.00

a single simulation are depicted in each column. As shown in the threshold plot of the first column, after applying an external force ($F = 75$N), none of the thresholds will be passed which means the low-level controller can negate the effect of this force by applying compensating torques (see torque plot). This simulation has been repeated with two times stronger disturbance ($F = 150$N). Although after applying this force, the COM and DCM deviate from the references none of the thresholds will be passed. As shown in the torque plot of this simulation, both of the input torques are saturated for a short period of time that means for a stronger push the robot can not keep its stability just by applying compensating torques. To check the performance of the controller in case of a very large disturbance, the simulation has been repeated with $F = 230$N. As it was expected, one of the thresholds is reached and the controller should modify the landing location of the swing leg and recalculate the reference trajectories. As shown in the torque plot of this simulation, both torques are saturated and the controller tries to keep the stability by adjusting the landing location of the swing leg and re-planing the reference trajectories based on this location. The new and previous COM and DCM reference trajectories are specified by the red line and dashed red respectively. It should be noted that the ability of the controller has been checked for more severe pushes and the robot was not able to withstand against $F > 245$N.

4.5 Linear Quadratic Gaussian

The presented controllers (LQR and LQI) are not robust against uncertainties like measurement noise. In this section, a Linear-Quadratic-Gaussian (LQG) controller will be designed to track the desired reference trajectories robustly. Moreover, the robustness of this controller will be assessed in presence of process disturbances and also measurement noise.

4.5.1 Overall architecture

The overall architecture of the proposed controller is depicted in Figure 4.10. As shown in this figure, a Kalman filter is designed and used to estimate the states of the system in the presence of measurement as well as process noises. Moreover, according to the observability of the state's error in each control cycle, an integrator is used to eliminate steady-state error. Using the estimated state and the output of the integrator, an optimal state-feedback control law for the tracking is designed as

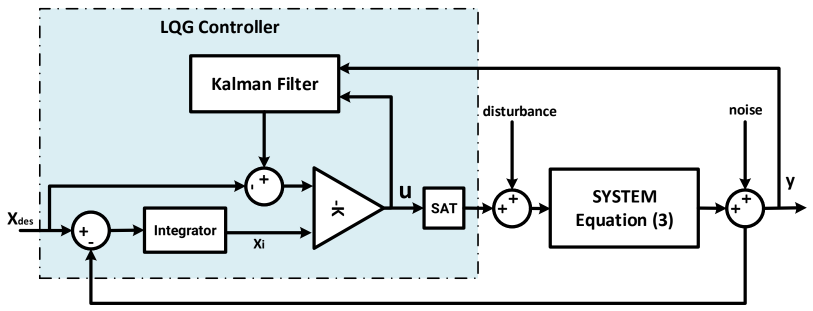


Figure 4.10: The overall architecture of the LQG controller.

follows:

$$u = -K \begin{bmatrix} \tilde{x} - x_{des} \\ x_i \end{bmatrix}, \tag{4.14}$$

where $\tilde{x}$ and x_{des} represent estimated states and desired states, respectively. x_i is the integrator output, K represents the optimal gain matrix which is designed to minimize the following cost function:

$$J(u) = \int_0^{\infty} \{z^{\intercal} Q z + u^{\intercal} R u\} dt, \tag{4.15}$$

where $z = [\tilde{x} \quad x_i]^{\intercal}$, Q and R are a trade-off between tracking performance and cost of control effort. There is a straightforward solution to find K by solving a differential equation which is called the Riccati Differential Equation (RDE). The quality of this controller depends on the choice of Q as well as R, and they generally are determined by trial and error.

4.5.2 Robustness analysis

The controller is the most important part of a real humanoid robot due to the unstable nature of this type of robot. In a real robot, measurements are never perfect and always are affected by unknown noises. A robust controller should guarantee the tracking performance in presence of measurement and process noises. The remainder of this section is dedicated to analyse the robustness of the proposed controller in the presence of uncertainties using a simulated robot. The mass of the simulated robot is 30kg, the height of COM is considered to be 1m and the foot length is 0.2m.

Robustness w.r.t. measurement errors

To examine this robustness, a simulation has been carried out. In this simulation, two independent zero-mean Gaussian are added to the measurements variables to simulate a noisy situation which occurs generally because of some reason like noisy outputs of the sensors and also using inaccurate dynamics model. In this simulation, example trajectories have been planned with a step duration (SD) of 1s, step length (*SL*) of 0.2m, and step width (*SW*) of 0.1m. The generated trajectories are used as desired trajectories and the controller should track them in presence of Gaussian noises. The results of this simulation are shown in Figure 4.11. As shown in this figure, the controller is robust regarding measurements noise and could track the references well.

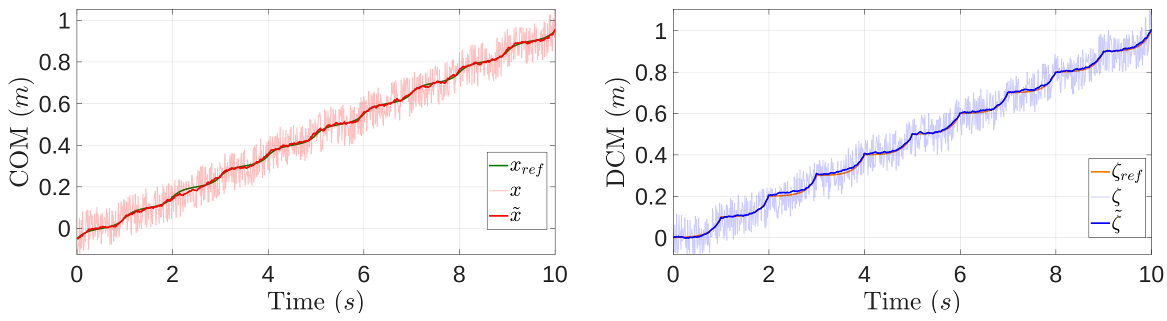


Figure 4.11: Simulation results of examining the robustness w.r.t. measurement errors. The measurements are modeled as a stochastic process by adding two independent Gaussian noises ($-0.05 \leq v \leq 0.05$) to the system output.

Robustness w.r.t. COM height error

Imagine a situation where a humanoid robot is carrying a heavy unknown weight object. In such situations, the height of the COM is changed, therefore the generated reference trajectories which are based on incorrect COM's height, may not guarantee the stability of the robot. To check this effect, a set of reference trajectories regarding the different height of COM ($c_z = c_z \pm 0.2$m) has been generated and depicted in Figure 4.12. These plots show that the incorrect height of COM has a significant effect in the frontal plane, but its effect in the sagittal plane is not significant. To examine the performance of the controller regarding this type of error, a simulation has been set up. In this simulation, a simulated robot should walk diagonally and the actual height of COM is considered $c_z = 1.2$m but the controller, state estimator, and the generated reference trajectories are designed based on $c_z = 1$m. The estimated and real positions of ZMP, COM, DCM have been recorded during the simulation and shown in Figure 4.13. The results showed that although the controller tracks the inaccurate reference trajectories, ZMP still remains inside the support polygon and the robot could perform stable walking.

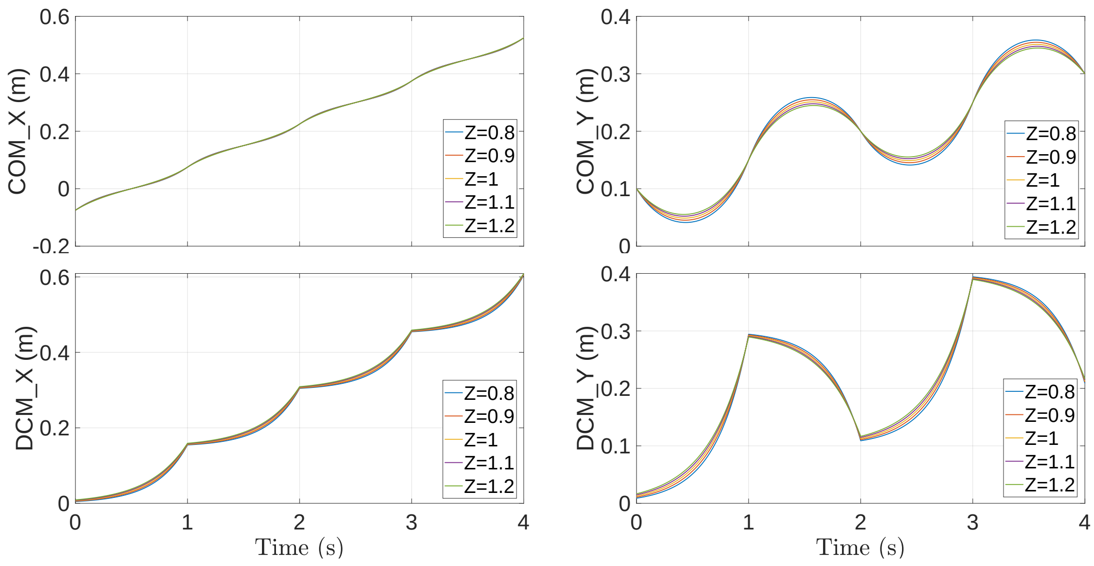


Figure 4.12: A set of reference trajectories for a 4-steps diagonal walk with regarding to different height of COM ($c_z = 1 \pm 0.2$m).

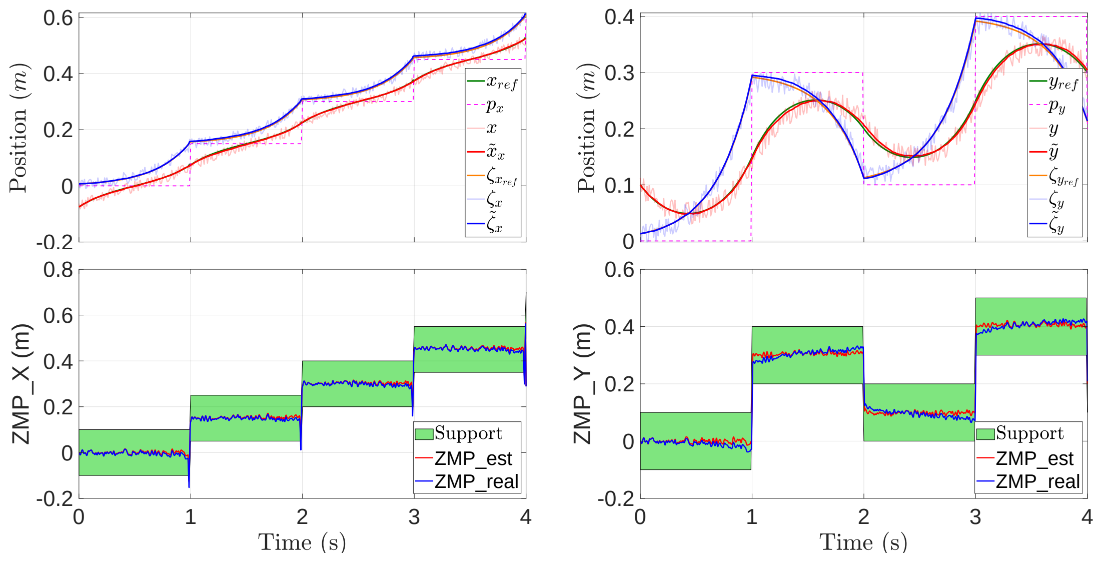


Figure 4.13: Simulation results of examining the robustness w.r.t. COM height error.

Robustness w.r.t. unknown external disturbance

In some situations, like collision with an obstacle or being pushed by someone or other similar situations, an unknown external force will be applied to the robot. A robust controller should be able to react against these types of disturbances and keep the stability of the robot. To analyse this robustness, a set of simulations has been carried out to check the withstanding level of the controller against these disturbances. In these simulations, while the robot is walking, a disturbance will be applied to its COM at $t = 2.5$s and the impact duration is $\Delta t = 10$ms. This simulation has been repeated six times with different directions and amplitude of disturbances. The simulation results are depicted in Figure 4.14. To determine the maximum level of withstanding of the controller, the amplitude of the impact has been increased until the robot could not track the references and fell. Based on these simulation results, $F = 91.4$N and $F = -97.2$N were the maximum level of withstanding of the controller.

As shown in ZMP plots of Figure 4.14, in case of a strong push, ZMP moves toward the edge of the support polygon, as a consequence robot starts to roll over. In case of a more severe force ($F > 90$N), the controller could not regain the stability of the robot because its output is bounded by the size of the support polygon. In such situations, the robot should change the landing location of the swing leg to regain its stability. According to the observability of DCM at each control cycle, the position of DCM at the end of the step can be predicted in advance by solving the DCM equation (4.8) as an initial value problem:

$$p_{x+1} = p_x + (\zeta_t - p_x)e^{w(T-t)}, \tag{4.16}$$

where p_{x+1} is the next footstep position, t, T and ζ_t, represent the time, step duration and measured DCM respectively. Based on this equation, the landing location of the swing leg can be adjusted by adding an offset according to the measured DCM at each control cycle. Unlike [19, 59] which were based on optimization methods, using the difference between the desired and the predicted landing location, we define a compliance margin and a slope to calculate an offset for modifying the swing leg landing location. Figure 4.15 graphically shows how this offset is calculated based on the DCM error. Indeed, a compliance margin is considered to prevent unnecessary modification and again is used to define a compliance slope that is proportional to the DCM error. Moreover, the modification offset should be saturated in a certain value to keep the new landing position inside a kinematically

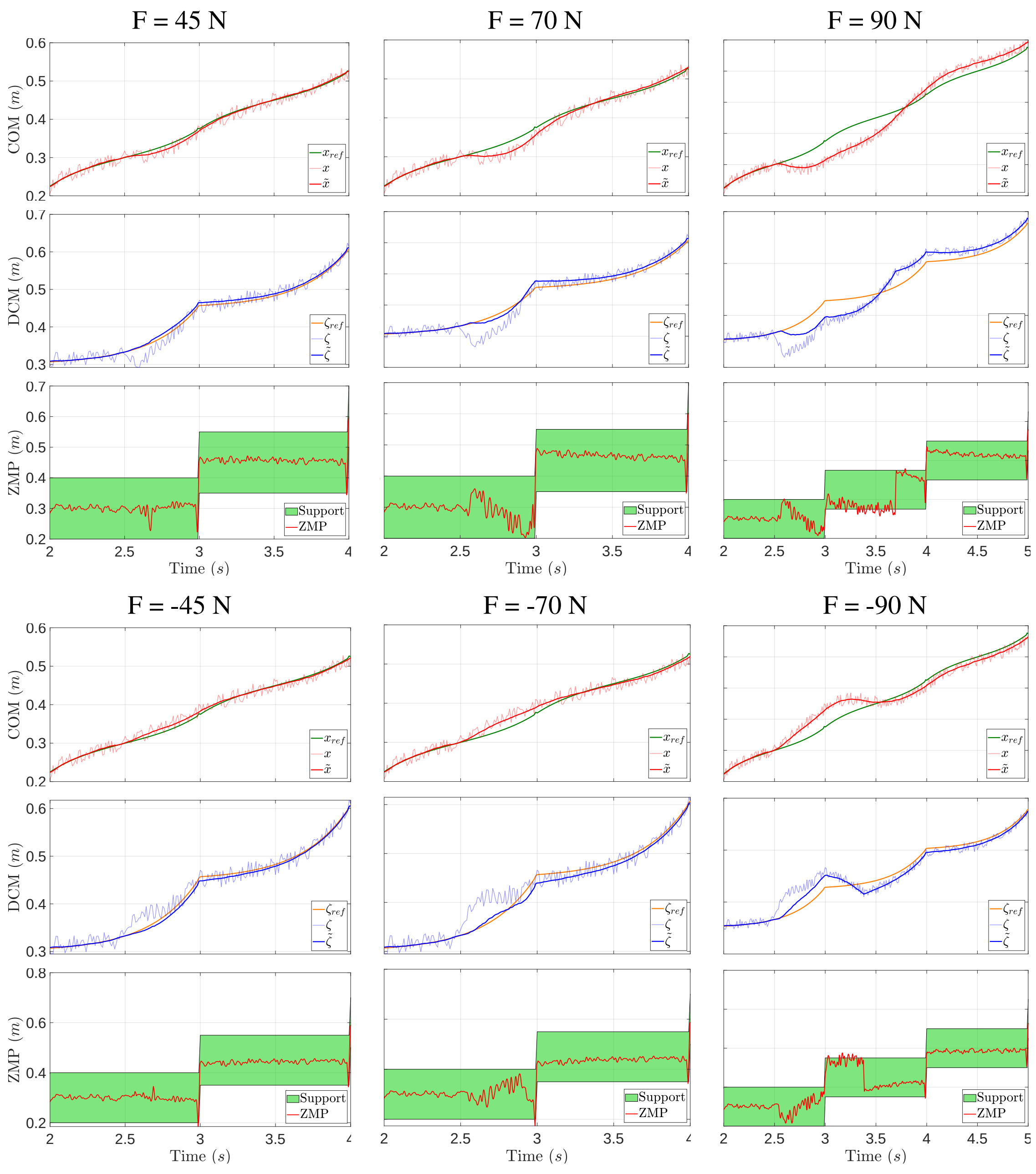


Figure 4.14: Simulation results of examining the robustness w.r.t. external disturbance. Each column represents the simulation results of forward and backward push with specified force.

reachable area of the robot. The compliance margin and slope are generally defined empirically. A simulation of online footstep adjustment is shown in Figure 4.16. In this simulation, the simulated robot should walk forward with a step length 0.2m and step duration of 1s and while the robot is walking, at $t = 2.5$s, the robot is subject to a severe forward push of $F = 120$N with an impact duration of $\Delta t = 10$ms, the compliance margin and slope being considered to be 0.025 and 1.2, respectively. As seen in Figure 4.16, after applying this push, the DCM (ζ_{mes}) totally deviates from its desired trajectory, which means the robot is starting to roll over and should change its swing leg landing position and replan the reference trajectories to regain its stability. The simulation results showed that

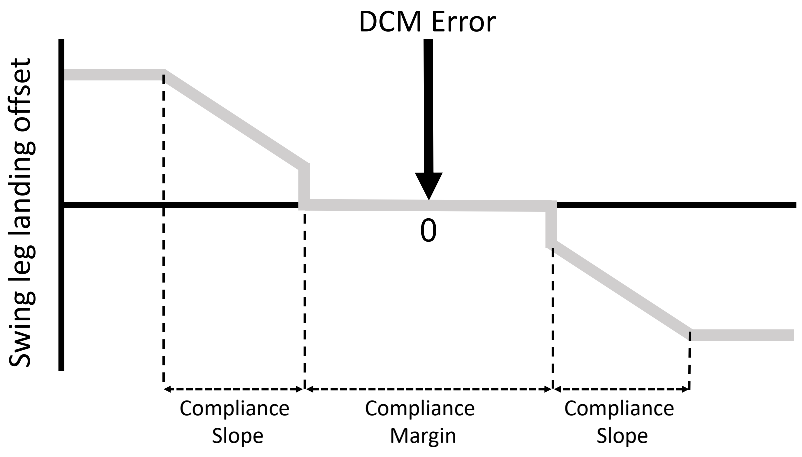


Figure 4.15: Adjusting the landing location of the swing leg based on DCM error.

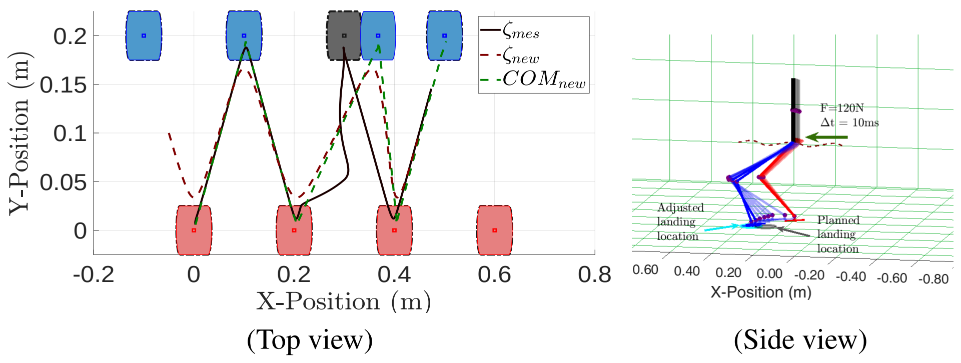


Figure 4.16: An example test of online foot step adjustment. The simulated robot is strongly pushed forward through the third step and the landing location of swing leg has been modified.

the controller could detect and react against this severe external push and could negate the effect of this disturbance.

4.6 Summary

In this chapter, we have tackled the problem of designing a robust walking controller. To this end, we used ELIPPM to design and develop a set of optimal control schemes. These controllers try to keep the stability of the robot by applying compensating torques at the ankle and hip of the robot. We showed that in case of large disturbances, these controllers can not keep the stability of the robot. To handle this situation, a high-level controller was designed based on DCM for adjusting the landing location of the swing leg to prevent falling. Particularly, we represented the overall dynamics of the system as a linear state-space system, and using that we formulated the controller as an LQG to generate a robust control solution based on offline optimization. The next step adjustment module was paired with the low-level controller to improve the withstanding level of the framework by online modification of step time and location according to the measured DCM in each control cycle. The functionalities of the controllers have been tested independently and the overall performance and robustness of the proposed framework were validated by performing several simulations. The simulation results showed that the next step adjustment module significantly improves the overall withstanding level of the framework. In the next chapter, we intend to use machine learning algorithms on top of the proposed LQG and propose a tight coupling between analytical control and Deep Reinforcement Learning.

Chapter 5

Deep Reinforcement Learning on Top of Analytical Control Approaches

This chapter proposes a modular framework to generate robust biped locomotion using a tight coupling between an analytical walking approach and deep reinforcement learning. This framework is composed of six main modules which are hierarchically connected to reduce the overall complexity and increase its flexibility. The core of this framework is a specific dynamics model which abstracts a humanoid's dynamics model into two masses for modeling the upper and lower body. This dynamics model is used to design an adaptive reference trajectories planner and an optimal controller which are fully parametric. Furthermore, a learning framework is developed based on Genetic Algorithm (GA) and Proximal Policy Optimization (PPO) to find the optimum parameters and to learn how to improve the stability of the robot by moving the arms and changing its center of mass (COM) height. A set of simulations are performed to validate the performance of the framework using the official RoboCup 3D League simulation environment. The results validate the performance of the framework, not only in creating a fast and stable gait but also in learning to improve the upper body efficiency.

5.1 Introduction

Developing robust locomotion for bipedal robots is a challenging problem that has been investigated for decades. Although several walking approaches have been proposed and walking performance has considerably improved, it still falls short of expectations in certain domains, such as speed and stability. The question is *how is it that humans can constantly change their direction when running while keeping their stability, but humanoids cannot?*

To find a good answer for this question, we start by reviewing recently proposed walking frameworks, consequently identifying four points of view related to the development of a fast and stable gait. In the first point of view, the fundamental framework's core is a dynamics model of the robot, based on which the walking planner and controller are designed. In this type of framework, to reduce the complexity of developing a whole-body dynamics model, some constraints are considered. Based on these constraints, an abstract model is designed instead of a real whole-body dynamics model [1, 5, 7, 9, 10, 60, 61, 62]. It should be mentioned that several studies exist where a whole-body dynamics model is developed [63, 64, 65]. In the second point of view, the core of the framework is a set of signal generators that are coupled together to generate endogenously rhythmic signals [31, 66, 67, 68]. This type of framework is called Central Pattern Generator (CPG)-based framework and is inspired by the neurophysiological studies on invertebrate and vertebrate animals [43, 69, 70]. These studies showed that rhythmic locomotion like walking, running, and swimming are generated by CPGs at the spinal cord that are connected together in a particular arrangement. In this type of framework, oscillators are assigned to each limb, typically to generate the setpoints (position, torque, etc.). Most humanoid robots have more than 20 Degrees of Freedom (DOF), therefore, adjusting the parameters of the oscillators is not only difficult but also trial-intensive [71]. Moreover, there is not a straight way to adapt sensory information to the oscillators. In the third point of view, walking trajectories are generated based on a heuristic algorithm such as reinforcement learning (RL), genetic algorithm (GA), etc. [66, 72, 73]. In this type of framework, the walking trajectories will be generated after a training period which needs many samples and takes a considerable amount of time. During training, the framework tries to learn how to generate the walking trajectories, subject to an objective function. In the fourth point of view, the framework is designed by combining the aforementioned approaches [21, 24, 71, 74, 75]. This type of framework is generally known as a hybrid walking framework. It tries to leverage the different capabilities of each approach to improve the final performance.

After studying all types of humanoid walking frameworks, to find the answer to the question raised at the beginning of this section, let us look at how babies start to walk. They start by learning to stand for a few seconds. They then improve the stability after many experiments, take a few steps, learn how to maintain equilibrium while moving; until, finally, after a long process of trial and error, a robust walking behavior emerges. This process shows how humans learn from previous experiences to improve its walking performance. Based on these explanations, we believe that the ability to learn from past experiences is one of the most important difference between human walking and robot walking. Particularly, a robot should be able to learn how to generate efficient locomotion according to different situations (e.g., learning to recover its balance from postural perturbations).

In the first two types of framework, the knowledge of robots is generally static, and does not evolve from past experiences. Therefore, they need to at least re-tune the parameters to be able to adapt to new environments. In the third type of framework, the learning process typically does not consider any dynamics model and is designed based on learning from scratch, which is trial intensive and not applicable to a real robot directly. Several research groups have been exploring how to learn from previous experiences to improve stability and robustness. We believe the fourth type is the best approach to develop a robust biped locomotion framework.

In this chapter, we propose a tight coupling between analytical control approaches and machine learning (ML) algorithms to develop a robust walking framework. Particularly, our contribution is a biped locomotion framework composed of two major components — an analytical planner and controller; and a fully connected neural network. The former is responsible for optimally controlling the overall state of the robot based on an abstract dynamics model. It is also responsible for generating reference trajectories using dynamic planners with genetically optimized parameters and overcome uncertainties up to a certain degree. The latter component — a fully connected network — is optimized with reinforcement learning to control the arms residuals and the COM height of the robot, thus improving the upper body efficiency, which impacts the overall stability and speed of the robot.

The remainder of this chapter is structured as follows: Section 5.2 provides an overview of related work. In Section 5.3, the concept of ZMP will be used to define a specific dynamics model which is composed of two masses. Afterwards, in Section 5.4, this dynamics model will be used to design an optimal controller which is able to track the walking reference trajectories, even in the presence of uncertainties. Section 5.5 explains how the problem of generating walking reference trajectories can be decomposed into five distinct planners. In Section 5.6, we will describe our learning approach and explain its structure. The overall architecture of the proposed framework will be presented in Section 5.7. In Section 5.8, three simulations scenarios will be designed to validate the performance of the proposed framework. According to the simulation results, its discussion and comparison with related work will be provided in Section 5.9. Finally, conclusions are presented in Section 5.10.

5.2 Related Work

Several of the proposed walking frameworks are based on learning approaches to generate stable locomotion for biped and multi-legged robots. Using ML algorithms for biped locomotion has made remarkable progress recently. These studies showed that using these algorithms on top of analytical approaches can improve robustness and performance significantly [71, 74]. In the remainder of this section, some recent proposed walking frameworks will be categorized and reviewed, focusing on those that use ML algorithms to improve their performance.

5.2.1 Combination of Model-based Walking and ML algorithms

MacAlpine et al. [74] designed and implemented a learning architecture to enable a humanoid soccer agent to perform omnidirectional walk. In their architecture, the overall dynamics of a humanoid robot is abstracted by a double inverted pendulum model which is parameterized to be able to learn a set of parameters for different tasks. The performance of their framework has been validated using a set of simulations that have been designed using SimSpark[1], a generic physical multiagent system simulator. The simulations results showed that their framework is able to learn multiple parameter sets according to the specified tasks.

Kasaei et al. [71] proposed a closed-loop model-based walking framework. Their dynamics model is composed of two masses that takes into account the lower and upper body dynamics of the robot. Based on this dynamics model, they generate walking reference trajectories and also designed an optimal controller to track these references. They showed the performance of their framework by performing a set of simulations using a simulated NAO robot in SimSpark. Moreover, they optimized the parameters using GA and showed that the maximum forward walking speed of the simulated robot reached 0.805m/s.

Koryakovskiy et al. [76] proposed two approaches for combining a Nonlinear Model Predictive Control (NMPC) with reinforcement learning to compensate model-mismatch. The first approach deals with learning a policy to compensate for control actions to minimize the same performance measure as their NMPC. The second approach was focused on learning a policy based on the difference between a transition predicted by NMPC and the actual transition. They performed a set of simulations to show the feasibility of both approaches and to compare their performances. The simulation results showed that the second approach was better than the first one. Moreover, They deployed the second approach on a real humanoid robot named Robot Leo to perform squat motion to validate the performance of their approach.

5.2.2 Combination of CPG-based Walking and ML algorithms

Gen Endo et al. [72] proposed a learning framework for a central pattern generator (CPG)-based biped locomotion controller using a policy gradient method. In their framework, the walking motion has been decomposed into stepping in place and a propulsive motion with a feedback pathways were acquired through their proposed policy gradient method. They showed that an appropriate feedback controller for straight walking can be obtained within 1,000 trials, on average. They also demonstrated

[1]http://simspark.sourceforge.net/

that the controller obtained in numerical simulation generates stable walking with a physical robot in the real world.

Missura et al. [2] proposed a walking framework that bootstraps a learning algorithm with a CPG-based walk engine. Their framework is composed of a feed-forward walking pattern generator, a state estimator, and a balance controller. In their framework, while the robot is walking, the balance controller adjusts the step size based on the estimated error and also learns how to improve the walking performance by adjusting the swing leg parameters. The performance of their framework has been validated using a set of experiments on a real humanoid robot. The results showed that their framework is able to keep the robot's stability even after applying a severe push.

5.2.3 Combination of CPG-ZMP based walking and ML algorithms

Massah et al. [25] developed a hybrid CPG-ZMP controller to generate stable locomotion for humanoid robots. In their approach, a set of non-linear oscillators were used to generate walking trajectories and two controllers were developed to handle small and large disturbances. They optimized the walking parameters using the differential evolution (DE) algorithm. The performance of their approach was demonstrated in the Webots robot simulator using the NAO humanoid robot.

In Chapter 2, we developed a hybrid CPG-ZMP based walking engine for biped robots which has a hierarchical structure and is fully parametric. We showed that this structure allows using a policy search algorithm to find the optimum walking parameters. To show this capability, we used an optimization technique based on Contextual Relative Entropy Policy Search with Covariance Matrix Adaptation (CREPS-CMA) [77] to tune the walking parameters. The performance of this walking engine has been validated by showing a fast and stable omnidirectional walk using a simulated NAO robot in Simspark (0.59m/s).

5.2.4 Learning to walk from scratch

Abreu et al. [73] applied a reinforcement learning algorithm to develop a fast and stable running behavior from scratch. In their approach, the environment has been represented by 80 states, and the action space is composed of 20 actions which were all the joint positions of a simulated humanoid robot. They used the Proximal Policy Optimization (PPO) based on the implementation provided by OpenAI [78]. The performance of their approach was shown by learning sprinting and stopping behaviors. The results demonstrated that both behaviors are stable and the sprinting speed stabilizes around 2.5m/s which was a considerable improvement.

Most of the aforementioned works combine a simplified model-based or a model-free approach with ML approaches to improve the performance of their walking. In the rest of this chapter, we develop an optimal closed-loop walking pattern generator based on a more complex dynamics model which takes into account the vertical motion of the COM and the torso's dynamics. Besides, we use the PPO algorithm, which is one of the most successful deep reinforcement learning methods, on top of our walking pattern generator to improve its robustness and efficiency and also to provide more human-like walking. An abstract overview of the proposed framework is depicted in Figure 5.1.

5.3 Dynamics Model and Stability Criteria

When designing a model-based walking, two general perspectives exist: (i) considering an abstract dynamics model which takes into account a trade-off between accuracy and simplicity; (ii) considering a whole-body dynamics model which is more accurate but, not only is it platform-dependent but

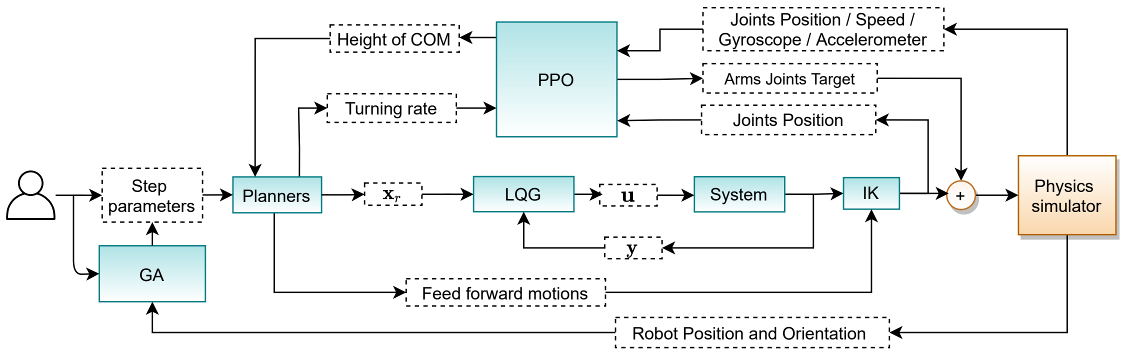


Figure 5.1: An abstract overview of the framework proposed for deep reinforcement learning on top of analytical control approaches. The highlighted boxes represent functional modules and the white boxes correspond to the exchange of data among them.

also resource-intensive due to its non-linear nature. In the rest of this section, the concept of ZMP (presented in 3.2.1) will be used to define an abstract dynamics model of a humanoid robot.

5.3.1 Dynamics Model

Although considering a full body dynamics model is not impossible, it generally needs powerful computational resources. Therefore, it is not affordable for real-time implementation. To reduce the complexity of the model and its computation cost, the overall dynamics is approximated by an abstract model. As already mentioned before (Section 3.2),the Linear Inverted Pendulum Model (LIPM) is a well-known abstract model in the community. LIPM is popular because it provides a simple, fast and efficient solution for walking dynamics that is suitable for real-time implementation. In this model, the overall dynamics of the robot is abstracted to a single mass that is connected to ground via a massless rod. Additionally, this model assumes that the vertical motion of the mass is restricted by a horizontally defined plane. According to these assumptions and using a set of predefined footsteps, the trajectory of COM can be obtained from a straightforward analytical solution which guarantees long-term stability. It should be mentioned that based on these assumptions, the equations in sagittal and frontal planes are equivalent and independent, therefore we just derive the equation in the sagittal plane. The schematic of this model is depicted in Figure 5.2(a). Using (3.4) and considering the LIPM's assumptions, the COM's motion equation can be obtained as follows:

$$\ddot{c}_x = \omega^2 (c_x - p_x), \tag{5.1}$$

where $\omega = \sqrt{\frac{g}{c_z}}$ is the natural frequency of the pendulum, p_x and c_x represent the positions of ZMP and COM, respectively. As aforementioned, LIPM tries to keep the COM's vertical position at a predefined position which causes the knee joints to be always bent. Indeed, walking with bent knees consumes more energy and does not resemble human walking [79]. To release this constraint and generate more energy-efficient and human-like walking, a sinusoidal motion is assigned to the vertical motion of the COM:

$$c_z = c_{z_0} + A_z \cos\left(\frac{2\pi}{StepTime} t + \phi\right), \tag{5.2}$$

where c_{z_0} denotes the COM's initial height, A_z is the amplitude, and ϕ represents the phase shift of the COM's vertical sinusoidal motion. The initial value of these parameters is determined by an expert.

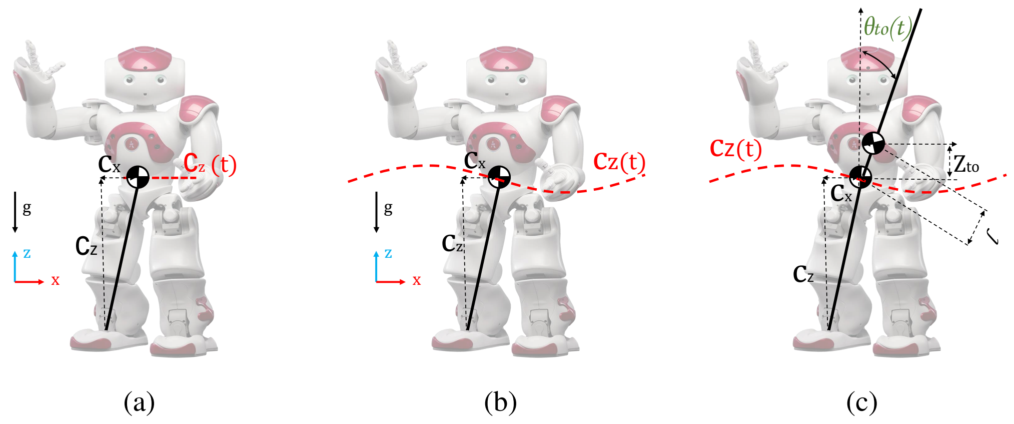


Figure 5.2: Schematics of the dynamics models: (a) LIPM; (b) LIPM with vertical motion of COM; (c) Proposed model.

Additionally, a controller can be designed to adjust these parameters based on sensory feedback. Although the current version of the dynamics model is able to provide fast and stable walking, it is not good enough to generate very fast walking. In fact, in some situations like when a push is applied, the COM accelerates forward, and, as a consequence, the ZMP goes behind the COG. In this situation, the robot tries to decelerate the COM by applying a compensating torque at its ankles, keeping the ZMP inside the support polygon. The compensating torque will be saturated once the ZMP is at the support polygon's boundary and, consequently, the robot is going to be unstable. In such situations, a human moves its torso to keep the ZMP inside the support polygon and prevent falling.

To consider the effect of the torso's motion in the dynamics model, another mass should be added to the dynamic model. This modification changes the dynamics model to be non-linear. Therefore, it does not have an analytical solution and it should be solved numerically. Biomechanical analysis of human walking showed that the torso motion can be represented by a sinusoidal function whose motion parameters are dependent on the current robot state and terrain conditions. The interesting point is that if the torso is considered as a mass with a small sinusoidal movement relative to the hip ($\sin(\theta_{to}) = \theta_{to}$), the dynamics model can keep its linearity. The schematic of this model is depicted in Figure 5.2 (c) and it can be represented by the following equation:

$$\begin{gathered}\ddot{c}_x = \mu(c_x + \frac{\alpha l}{1+\alpha}\theta_{to} - p_x) - \frac{\alpha\beta l}{1+\alpha\beta}\ddot{\theta}_{to} \\ \alpha = \frac{m_{to}}{m_c}, \quad \beta = \frac{z_{to}}{c_z}, \quad \mu = \frac{1+\alpha}{1+\alpha\beta}\omega^2, \quad x_{to} = c_x + l\theta_{to},\end{gathered} \tag{5.3}$$

where x_{to}, θ_{to} and l are the position, angle and length of the torso, m_c and m_{to} represent the masses of lower body and torso and z_{to} and c_z are the torso and the COM height, respectively.

5.4 Controller

In this section, an optimal controller will be designed for tracking the reference trajectories to minimize the tracking error. To do that, (5.3) will be represented as a linear state-space system. Then, this system will be discretized to be used in a discrete-time implementation. Afterward, we will explain how this system can be used to design an optimal controller. The process of designing the

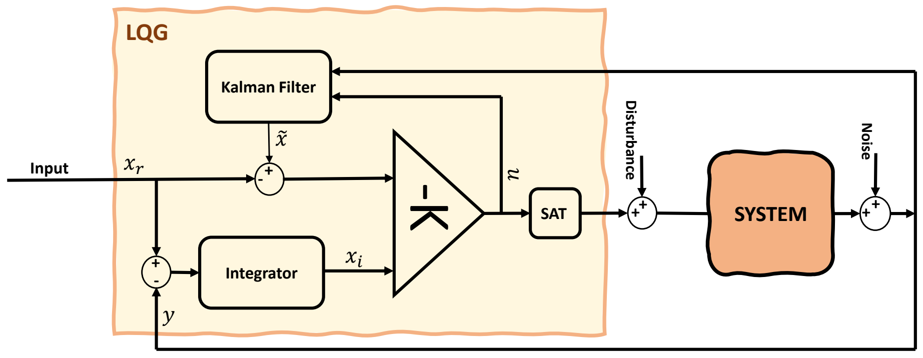


Figure 5.3: Overall architecture of the proposed LQG controller for tracking the reference trajectories.

controller starts by defining a linear state space system based on Equation 5.3:

$$\frac{d}{dt}\underbrace{\begin{bmatrix} c_x \\ \dot{c}_x \\ \theta_{to} \\ \dot{\theta}_{to} \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ \mu & 0 & \frac{\mu\alpha l}{1+\alpha} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_x \\ \dot{c}_x \\ \theta_{to} \\ \dot{\theta}_{to} \end{bmatrix}}_{X} + \underbrace{\begin{bmatrix} 0 & 0 \\ -\mu & \frac{-\alpha\beta l}{1+\alpha\beta} \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} p_x \\ \ddot{\theta}_{to} \end{bmatrix}}_{u}$$
$$y = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} c_x \\ \dot{c}_x \\ \theta_{to} \\ \dot{\theta}_{to} \end{bmatrix}}_{X}. \tag{5.4}$$

The presented system is a continuous system and should be discretized for implementation in discrete time. To discretize this system, we assume that $\dot{c}_x$ and $\dot{\theta}_{to}$ are linear. Therefore p_x and $\ddot{\theta}_{to}$ are constant within a control cycle. Thus, the discretized system can be represented as follows:

$$\begin{gathered} X(k+1) = A_d X(k) + B_d u(k) \\ y(k) = C_d X(k), \end{gathered} \tag{5.5}$$

where k represents the current sampling instance, A_d, B_d and C_d are the discretized versions of the A, B and C matrices in (5.4), respectively. According to this discretized dynamics model, an optimal closed-loop controller can be designed to track the reference trajectories. This controller is an LQG which is composed of two main modules: a state estimator and an optimal controller gain. The overall architecture of this controller is depicted in Figure 5.3. In the remaining of this section, each module will be explained and the overall performance of the controller will be validated.

5.4.1 State Estimator

An LQG controller is able to track the reference trajectories even in the presence of measurement noise. This controller uses a state estimator to cancel the effect of uncertainties which can be raised because of many aspects like errors in modeling the system, sensor noise, backlash of the gears, etc. In particular, this controller uses a state estimator to estimate the current state of the system according to the control inputs and the observations.

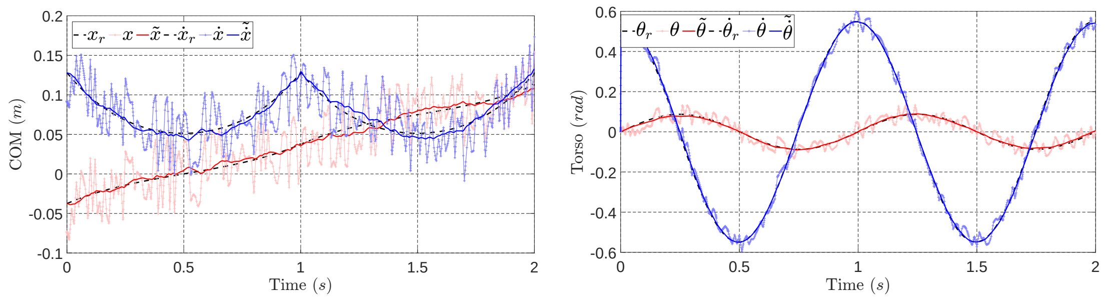


Figure 5.4: Simulation results of examining the state estimator performance. In this simulation, the measurements are affected by a Gaussian noise $\mathcal{N}(0, 6.25\text{e-}4)$ to simulate uncertainties. In these plots, light-blue and light-red lines represent the measurements, solid-blue and solid-red lines are the estimated values, dashed black lines represent the references.

In our target framework, we considered that the position of the joints is available through measurements and the torso orientation can be obtained based on an Inertial Measurement Unit (IMU) information which is mounted on the torso. Based on the joint information and using a Forward Kinematic (FK) model of the robot, the current configuration of the robot can be estimated. In this estimation, the support foot is considered to be in flat contact with the ground, which is not always true. Therefore, the whole body orientation with respect to the ground should be added to this estimation. To do so, the IMU information is used to rotate the current configuration. Based on this configuration, the COM position can be estimated at each control cycle and its velocity can be obtained from its position's derivative, followed by a first-order lag filter.

To validate the performance of this module, a simulation has been designed. In this simulation, the observations are modeled as a stochastic process by applying additive Gaussian noise to the measured states. The simulation results are shown in Figure 5.4. According to the simulation results, the state estimator is able to estimate the states in such conditions.

5.4.2 Optimal Gain

As shown in Figure 5.3, the optimal control law is obtained using the following formulation:

$$u = -K \begin{bmatrix} \tilde{x} - x_r \\ x_i \end{bmatrix}, \tag{5.6}$$

where $\tilde{x}$ and x_r denote the estimated states and the reference states, respectively. x_i is the integration of error which is used to eliminate the steady-state error, K represents the optimal gain of the controller that should be designed to minimize the following cost function:

$$J(u) = \int_0^\infty \{z^\intercal Q z + u^\intercal R u\} dt, \tag{5.7}$$

where $z = [\tilde{x} \quad x_i]^\intercal$, R and Q are positive-semidefinite and positive-definite matrices which are determined by an expert. In fact, these matrices determine a trade-off between cost of control effort and tracking performance. Therefore, the performance of the controller is sensitive to these matrices. It should be noted that there is a straightforward solution to determine K based on solving a differential equation named Riccati Differential Equation (RDE).

To check the performance of the proposed controller, a simulation has been performed. In this simulation, a set of reference trajectories has been generated and the controller should track these references in presence of measurement noise. The simulation results are shown in Figure 5.5. The

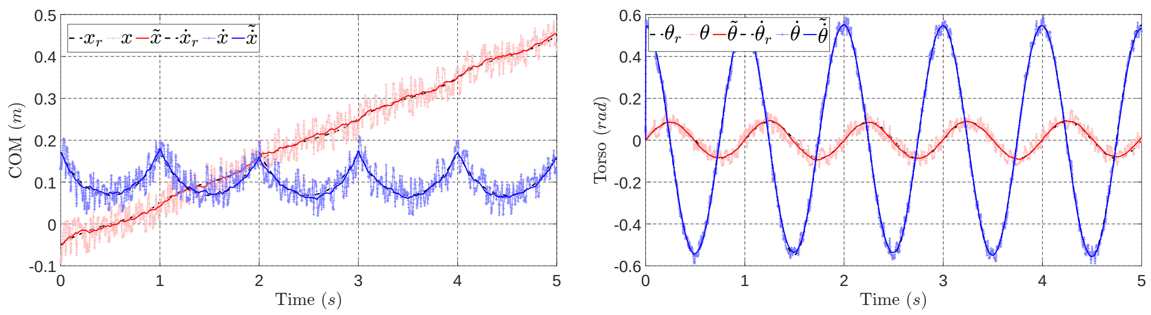


Figure 5.5: Simulation results of examining the controller performance in presence of noise. In this simulation, the controller should track a references trajectories in presence of the noise $\mathcal{N}$(0, 6.25e-4).

results showed that the controller is able to track the references even when the measurements are affected by noise. In the next section, we explain how the reference trajectories are generated.

5.5 Reference Trajectories Planner

Our walking reference trajectories planner is composed of five sub planners which are connected together hierarchically. The first level of this hierarchy is a footstep planner which generates a set of foot positions based on given step information, terrain information, and some predefined constraints (e.g., maximum and minimum step length, step width, distance between feet, etc.). To do so, we consider a state variable to represent the current state of the robot's feet:

$$s = (x_l, y_l, \theta_l, \phi_l, x_r, y_r, \theta_r, \phi_r), \tag{5.8}$$

where $x_l, y_l, \theta_l, x_r, y_r$ and θ_r are the position and orientation of the left and right feet, respectively. ϕ_l and ϕ_r represent the current swing state which is 1 if the foot is the swing foot and -1 otherwise. Walking is a periodic motion that is generated by moving the right and the left legs in alternation. Therefore, we parametrize a step action by a length and an angle from the swing foot position at the beginning of steps $a = (R, \sigma)$. According to the input parameters and the current state of the feet, an action should be taken and the state transits to a new state, $s' = t(s, a)$. Afterwards, the current footstep will be saved ($f_i \quad i \in \mathbb{N}$) and ϕ_l and ϕ_r will be toggled.

The second planner is the ZMP planner that uses the planned footstep information to generate ZMP reference trajectories. In our target framework, we used our ZMP reference planner presented in Section 3.3.3. The third planner is the swing leg planner which generates the swing leg trajectory using a cubic spline function. This planner uses three control points that are the position of the swing leg at the beginning of the step, the next footstep position and a point between them with a predefined height (Z_{swing}). The fourth planner is the global sinusoidal planner which generates three sinusoidal trajectories for the COM height, the torso angle and the arm position. The fifth planner is the hip planner which uses the generated ZMP and torso trajectories to generate hip trajectory. Indeed, these trajectories are used to determine the positions of the hip at the begging and the end of step. Using these positions, Equation (5.3) can be solved as a boundary value problem as follows:

$$c_x(t) = g_x + \frac{(g_x - c_{x_f})\sinh\left(\sqrt{\mu}(t-t_0)\right) + (c_{x_0} - g_x)\sinh\left(\sqrt{\mu}(t-t_f)\right)}{\sinh(\sqrt{\mu}(t_0 - t_f))}, \tag{5.9}$$

where $g_x = p_x - \frac{\alpha l}{1+\alpha}\theta_{to} + \frac{\alpha\beta l}{\mu(1+\alpha\beta)}\ddot{\theta}_{to}$ and t_0, t_f, c_{x_0} and c_f are the times and corresponding positions of the hip at the beginning and at the end of a step, respectively. In this work, T_{ds} is considered to be zero, which means ZMP transits to the next step at the end of each step instantaneously. Moreover, c_{x_f} is assumed to be in the middle of current support foot and next support foot ($\frac{f_i+f_{i+1}}{2}$).

5.6 Learning Framework

Our learning framework employs the Proximal Policy Optimization (PPO) algorithm, introduced by Schulman et al. [80], which was chosen due to its success in optimizing low-level skills concerning the NAO robot [73, 81, 82, 83, 84], and high-level skills [85], where it outperformed other algorithms such as TRPO or DDPG. The chosen implementation uses the clipped surrogate objective:

$$\begin{aligned} L(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t], \\ where\ \ r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}(a_t|s_t)}}, \end{aligned} \tag{5.10}$$

where $\hat{A}_t$ is an estimator of the advantage function at timestep t. The clip function clips the probability ratio $r_t(\theta)$ in the interval given by $[1-\varepsilon, 1+\varepsilon]$. This implementation alternates between sampling data from multiple parallel sources and performing several epochs of stochastic gradient ascent on the sampled data, to optimize the objective function.

The clipping parameter ε was set to 0.2, as suggested by Schulman et al. [80]. Also, as in the implementations published by OpenAI for the 3D humanoid environment [78], the entropy bonus was not used, and the number of optimization epochs and batches was set to 10 and 64, respectively. Some other hyperparameters were tuned using grid search: step size (2.5×10^{-4}); batch size (4096); and the Adaptive Moment Estimation (Adam) [86] optimizer was set to use a constant scheduler. Finally, the Generalized Advantage Estimation (GAE) [87] algorithm's parameters — gamma and lambda — were set to 0.99 and 0.95, respectively, accordingly to the ranges established by the GAE's authors as best-performing for 3D biped locomotion.

The policy is represented by a multilayer perceptron with two hidden layers of 64 neurons. The number of inputs, outputs, and the maximum number of time steps for the optimization are dependent on the scenario and will be described in section 5.8.3. The training session was parallelized to improve the optimization duration.

5.7 Overall Structure

In this section, the previously introduced planner and the controller will be coupled together to generate stable locomotion. To do that, we designed a modular framework composed of six main modules. The overall architecture of this framework is depicted in Figure 5.6. As shown in this figure, the walking process is controlled by a state machine which abstracts the process into four distinct states: *Idle*, *Initialize*, *Single Support* and *Double Support*. In this state machine, the transitions are triggered by a timer that is associated with each state. Additionally, it can be triggered by an emergency signal generated according to the controller's state in key moments, such as when a swift move is necessary to regain equilibrium after a strong external perturbation. The *Idle state* is the initial state in which the robot is standing in place and waiting to receive a walking signal, which can be generated by an operator or a path planning algorithm. That signal triggers the *Initialize state*, in which the walking parameters and configurations are loaded from a database. Afterwards, the robot is ready to walk by shifting its COM towards the first support foot. The next state is triggered after a predefined

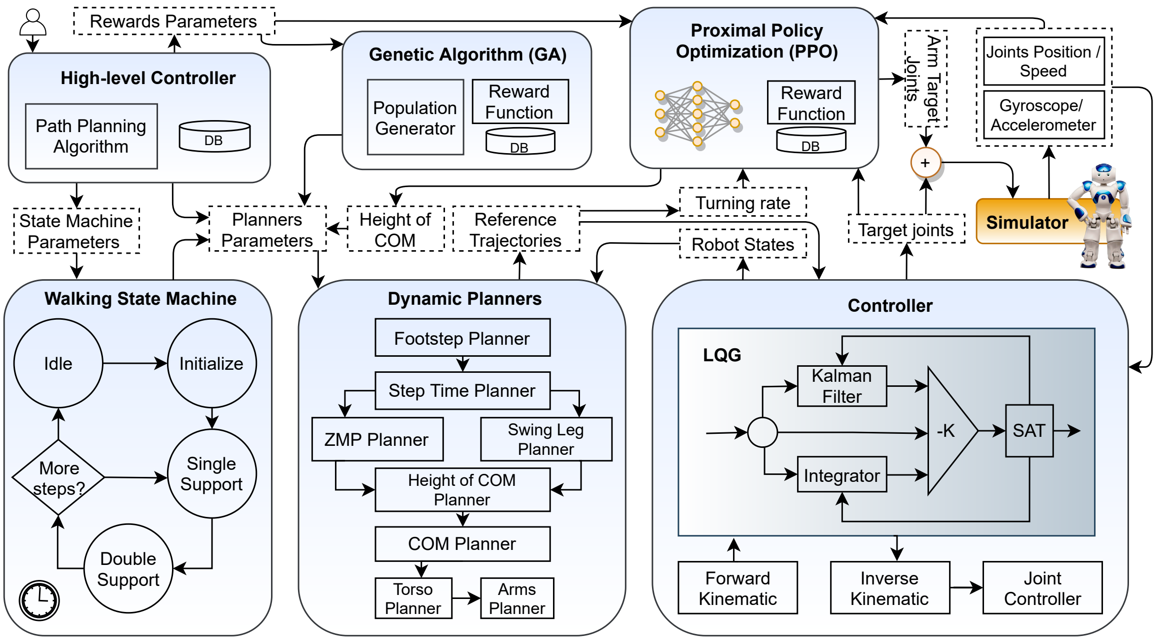


Figure 5.6: Overall architecture of the framework proposed for using deep reinforcement learning on Top of analytical control approaches. The highlighted boxes represent the main modules and the exchanged information among them is represented by the white boxes.

time. During *Single Support State* and *Double Support State*, the dynamics planner generates the walking reference trajectories according to the generated walking signal and the controller tries to track these references.

At the same time, the neural network receives a set of observations, including data from inertial sensors, joints' position and speed, target joint positions generated by the LQG controller, and the target turning rate. The network outputs residuals which are added to the target joint positions before being fed to the simulator, which runs the next simulation step. The neural network also adjusts the height of the COM, which is then used by the dynamic planners in the following iteration.

In the next section, a set of simulations will be carried out to verify the framework's performance. Moreover, we will show how the planner's parameters are optimized using a genetic learning approach, and what is the impact of the policy gradient algorithm on the performance of the framework.

5.8 Simulations

In this section, we introduce a set of simulation scenarios to validate the performance of the proposed framework. The simulation scenarios have been designed using the official RoboCup 3D simulation environment which is based on SimSpark, a multi-agent simulator. This simulator relies on the Open Dynamics Engine (ODE) to simulate rigid body dynamics. The physics engine is updated every 0.02s. The simulator can also be configured to update the physics engine just after receiving commands from all agents. This greatly improves simulation speed and provides a better environment for learning approaches.

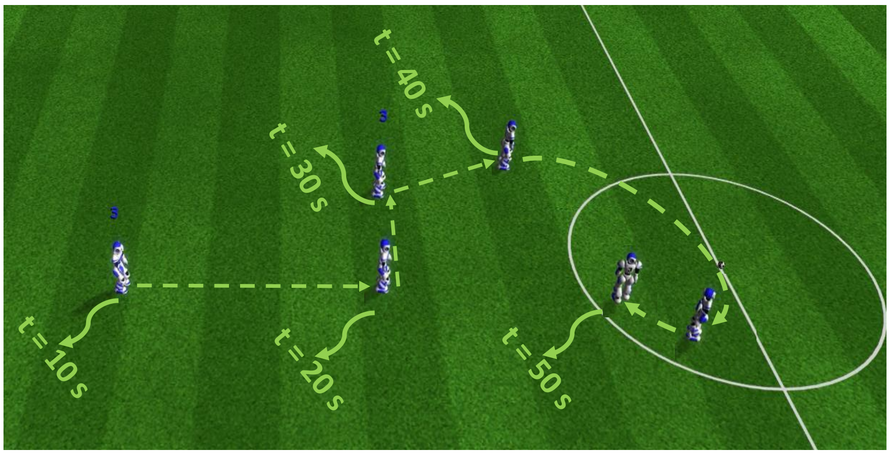


Figure 5.7: Omnidirectional walk scenario. In this scenario, the simulated robot should follow the commands that are generated by an operator: Green-dashed arrows show a set of commands that has been generated for this simulation, including forward walk, side walk, diagonal walk and turning while performing diagonal walking.

5.8.1 Omnidirectional Walk

This scenario is designed to demonstrate the performance of the framework in providing an omnidirectional walk. The simulated robot starts from an idle state and follows a command comprising length (X), width (Y), and angle (α) of the step, which is determined by an operator. Note that the step time is constant and set to 0.2s. To avoid discontinuity in the input command, a first-order lag filter is used, yielding a smooth transition.

At the beginning of this scenario, the robot is walking in place and all the setpoints are zero ($X = 0.0$m, $Y = 0.0$m, $\alpha = 0.0$deg/s). At $t = 10$s, the operator sets the step length ($X = 0.05$m) to generate forward walking; at $t = 20s$, the operator resets the step length ($X = 0.0$m) and sets the step width ($Y = 0.04$m) to generate side walking; at $t = 30$s, while the robot is performing side walking, the operator sets the step length ($X = 0.05$m) to generate diagonal walking; at $t = 40$s, while the robot is performing diagonal walking, the robot is commanded to turn right simultaneously, by setting the step angle ($\alpha = 10$deg); and finally, at $t = 60$s, all the setpoints are reset and the robot starts walking in place. A set of snapshots of this simulation is depicted in Figure 5.7. The simulation results showed that the framework was able to generate omnidirectional walking according to the input commands. A video of this simulation is available online at `https://youtu.be/yAkLvrT849A`.

5.8.2 Optimizing the Walking Parameters

This scenario is focused on optimizing the walking parameters to generate the fastest stable forward walk. In this scenario, the robot is placed 10m away from the halfway line and it should walk straight forward towards that line as fast as possible. Initially, the best parameters were hand-tuned and, after several attempts, the maximum walking speed did not exceed 0.53m/s. Afterwards, based on the parametric nature of the proposed planner and controller, a GA is used to optimize the parameters to improve the walking speed. To do that, 8 parameters of the framework have been selected to be optimized. These parameters are the step length (x), step width (y), step angle (α), the height of the swing leg (z_{sw}), duration of a step (T_{ss}), torso inclination (I_{to}), the amplitude of the COM (A_z) and amplitude of the torso movement (A_{to}). In our optimization scenario, the simulated robot should walk forward for 10 seconds and its performance will be evaluated based on the following cost function:

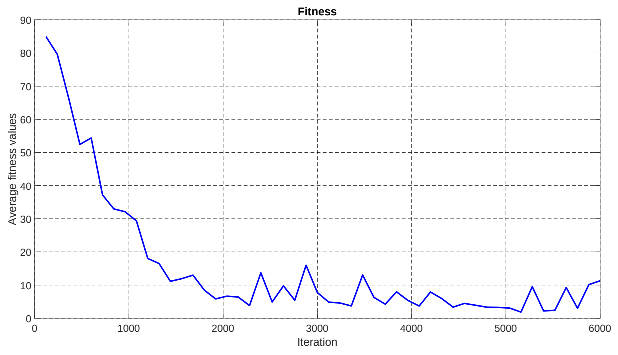


Figure 5.8: Evolution of fitness.

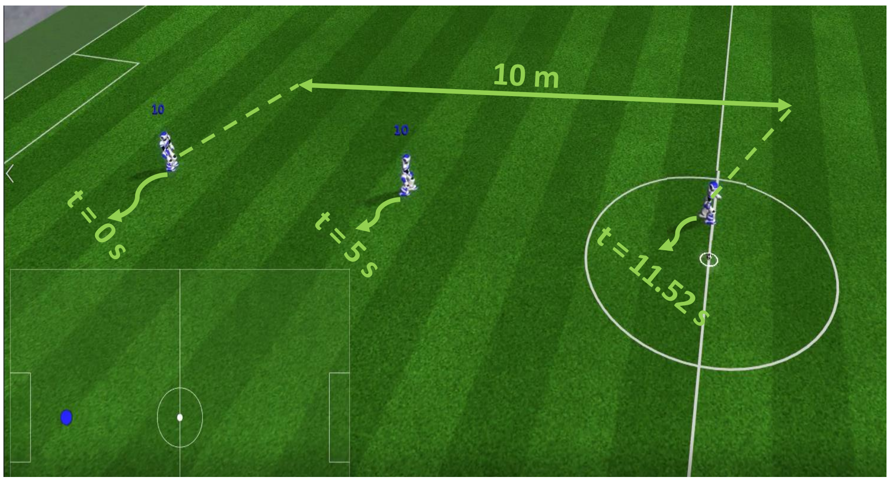


Figure 5.9: The optimization scenario and the results of an exemplary test after optimizing the parameters. In this test, the simulated robot has been placed at a specific point which is 10m far from the center of the field and it should walk towards the center as fast as possible. The results showed that the robot touched the midline at $t = 11.52$s.

$$f(\phi) = -|\delta x| + |\delta y| + \varepsilon, \tag{5.11}$$

where ϕ represents the selected parameters, δX and δY are the distance covered in X-axis and Y-axis, respectively, ε is used to penalize the robot when it falls during walking, being 100 in such cases and 0 otherwise. According to this cost function, the simulated robot is rewarded for a straightforward walk and it is penalized for deviation and falling. A slow and stable forward walking (0.11m/s) is used as an initial solution to start the optimization process. It should be noted that each iteration has been repeated three times and the average of the finesses was used to be sure about the walking performance. The cost values have been recorded for each iteration and the average cost values can be visualized in Figure 5.8. The average cost value starts at around 85 and after about 2000 iterations, it drops under 10, which is much better than the first solution. The optimization has been executed for 6000 iterations. After optimizing the parameters, the walking velocity reaches 0.866m/s, which is 63.4% faster than the best hand-tuned solution. The best parameters found by the GA are shown in Table 5.1.

The optimization scenario and a set of snapshots of a test are shown in Figure 5.9. A video of this simulated scenario is available online at `https://youtu.be/kVaQjt5wi0g`.

Table 5.1: The best parameters.

Parameter	Symbol	Value
Step duration	T_{ss}	0.1274s
Step length	x	0.09059m
Step width	y	0.010086m
Step angle	α	-0.2899deg
Height of swing	z_{sw}	0.038m
Torso inclination	I_{to}	5.601deg
Amplitude of height of COM	A_z	-0.004
Amplitude of torso movement	A_{to}	-1.9195

To compare the effectiveness of the dynamics model, this scenario has been repeated for the dynamics models (a) and (b) presented in Figure 5.2. To do so, the planner and the controller have been adjusted according to the dynamics models and then their parameters have been refined manually. Finally, this simulation scenario has been repeated to find the maximum forward speed of each model. The simulation results are summarized in Table 5.2. The simulation results validated that the sinusoidal motion of the height of COM improves the stability and allows the robot to move faster in comparison with fixed COM.

Table 5.2: Summary of the results in the maximum speed scenario.

Dynamics model	maximum speed
LIPM	0.590m/s
LIPM + vertical motion of COM	0.630m/s
LIPM + vertical motion of COM + Torso	0.866 m/s

5.8.3 Learning to Improve the Upper Body Efficiency

This scenario was designed to improve the efficiency of the walking gait in terms of speed and stability during the most common walking patterns – forward walking and turning. To mitigate the effects of the learning process on the maneuverability and predictability of the walking trajectory, only the arms actuators and COM height were optimized. The robot is initially placed in an arbitrary position within a squared area of side length 2m, as depicted in Figure 5.10. It then starts walking forward with the best parameters found in Section 5.8.2. When the robot steps out of the predefined area, it starts to turn in either direction at a random rate, $|\alpha| \in$ [30deg/s,60deg/s], until it is facing the square again. This process is repeated continuously until the episode ends with the robot falling (detected when its z coordinate drops below 0.3m). The fact that the robot runs at full speed when changing direction, and that it needs to constantly adapt to different turning rates makes this a very challenging scenario.

This optimization problem can be formalized as a Markov Decision Process (MDP) – a 4-tuple $\langle S, A, p, R\rangle$ – where S denotes the set of states, A the set of possible actions, and R the set of possible numerical rewards. The dynamics of the MDP is given by the state-transition probability function $p(s'|s,a) : S \times S \times A(s) \rightarrow [0,1]$ which gives the probability of ending in state s' given the current state s and action a.

The interaction between agent and environment is performed at discrete time steps ($t = 0, 0.02, \ldots$). The robot's behavior was optimized by the PPO algorithm, using 67 observed variables, as listed in Table 5.3. The first parameter indicates the current turning rate. The inertial sensors (gyroscope and

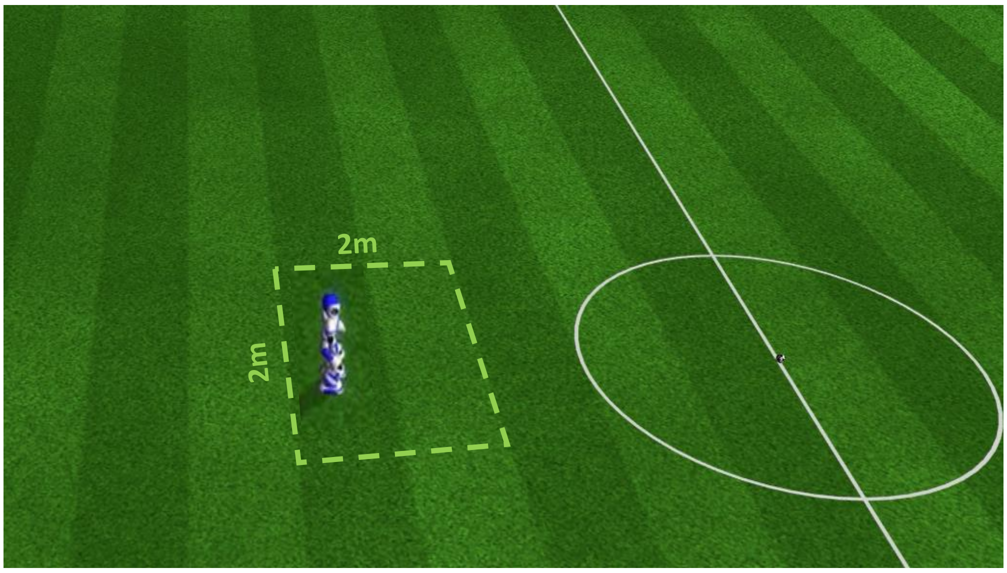


Figure 5.10: Stability optimization scenario. The robot exploits the most common walking patterns – forward walking and turning – to keep itself within a predefined squared area of side length 2m. When inside that area, the robot walks forward as fast as possible. Otherwise, it turns at a random rate until it is directed towards the area.

Table 5.3: State space for the stability optimization.

Parameter	Data size ($\times$ 32b)
α	1
Gyroscope	3
Accelerometer	3
Joints Position	20
Joints Speed	20
Controller Actions	20

accelerometer) are both composed of 1 variable per axis in a three-dimensional space. The position and speed of all joints (excluding the head) are important to obtain a correct state representation, even though the action space only controls a limited number of these joints. Finally, the joint positions computed by the analytical controller are fed to the algorithm and later added as residuals to the output values. These positions can be used by the network to predict the next analytical state so that the produced residuals can be adjusted accordingly. In preliminary tests, removing this information from the state space results in a loss of performance between 5% and 20%, depending on possible action space combinations.

The action space encompasses four angle variables per arm (shoulder roll, shoulder pitch, elbow yaw, elbow roll) and one variable to define the setpoint of the COM height at each step. The arm joints angles are computed by summing the analytical controller's output to the neural network's corresponding output. This forms a controller which uses the planner's arms control signals both as state data and action bias.

The objective of this scenario is to improve forward speed and stability at all times (i.e., when moving forward or turning). The former requirement is met by rewarding the agent for stepping forward and not sideways, which can be numerically translated into the scalar projection of its velocity vector $\vec{v}$ in the direction of its orientation unit vector $\vec{o}$. Let $\vec{v} = P_t - P_{t-1}$, where P_t and P_{t-1} are the

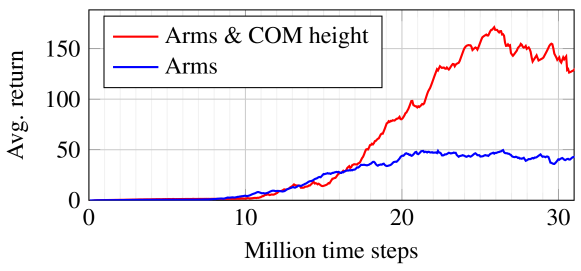


Figure 5.11: Learning curves considering only the arms (blue) and considering arms and COM height (red).

Table 5.4: Original scenario results – average speed and duration

Parameter	Ep. Length Mean ± SD (s)	Speed Mean ± SD (m/s)
Baseline	5.1 ± 2.9	0.602 ± 0.027
Arms	51.6 ± 31.8	0.710 ± 0.037
Arms & COM height	148.2 ± 153.0	0.956 ± 0.060

current and previous positions of the robot, respectively. The partial reward to motivate forward speed is then $\max(\vec{v} \cdot \vec{o}, 0)$, where $\cdot$ denotes the dot product. The minimum reward value is limited to zero because walking backward or sideways is not worse than falling. The second requirement — stability — is motivated by a constant k, set empirically to 0.01, that rewards the agent for staying alive. More precisely, it favors stability at the cost of lowering the speed. The complete immediate reward can then be formulated as:

$$r = \max(\vec{v} \cdot \vec{o}, 0) + k. \tag{5.12}$$

The learning algorithm was first applied to the arms actuators and later extended to the COM height. Figure 5.11 shows the average return evolution when learning only the arms (blue line) and after adding the COM height (red line). The former optimization plateaued at around 20M time steps, and the latter at around 26M time steps. It is important to note that the return obtained during the optimization was based on a stochastic policy whereas, in the following tests, we used the corresponding deterministic policy.

The results were divided into two sections: Original scenario – the robot is tested in the same scenario used for learning (see Figure 5.10) and the analysis delves into the same metrics used to define the reward function; Straightforward path – the robot's direction is constantly corrected to describe a linear path and the resulting behavior is analyzed in terms of efficiency.

Original Scenario Results

The robot was evaluated concerning stability and speed in the same scenario where the learning algorithm was applied. Stability was measured by the episode length since it terminates once the robot falls to the ground. No time limitation was imposed per episode. Speed was measured at every iteration and averaged at the end of the episode. Table 5.4 lists the average speed and duration results for 500 episodes. The first line corresponds to the walk optimized in Section 5.8.2, which is used as a baseline. The robot walks on average for 5.1 seconds with a standard deviation (SD) of 2.9s before falling, generally on the first or second sharp change of direction. The mean speed, from a standstill position to the end of each episode, was 0.602m/s with an SD of 0.027m/s.

After learning how to control the arms, the episode duration increased tenfold, on average, and the mean speed rose to 0.710m/s. Most falls occur at an advanced stage or during the initial sharp turns,

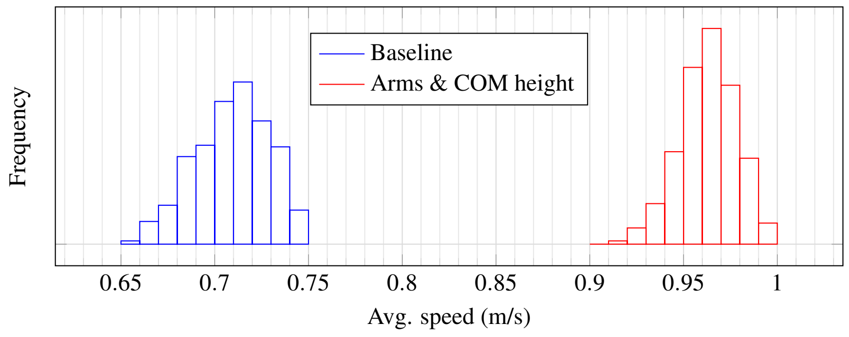


Figure 5.12: Mean speed comparison between the baseline (on the left) with the best optimization using the Arms & COM Height controller (on the right). These values were averaged for 500 successful episodes, where the robot runs for 12m in a straight line.

hence the larger standard deviation. Adding the COM height to the group of controlled variables increased the episode length to almost 29 times the initial value. When compared to the version without the COM height adaptation, this metric improved approximately 3 times. The mean speed rose to 0.956m/s, a gain of almost 60% in relation to the baseline. In every episode, the robot eventually falls. As aforementioned, when the robot steps out of the predefined area, it starts to turn with a random turning rate between 30 and 60deg/s, in either direction. Most falls occur when approximating the upper limit or when the rotation direction is inverted after the robot enters and exits the predefined area in a short time (e.g. when stepping over a corner of that area). A video comparing the baseline with the Arms & COM height optimization is available online at `https://youtu.be/hifFdgeuDIs`.

Straightforward Path Results

To evaluate the gait efficiency without taking stability into account, the displacement of certain joints, as well as the average speed, were analyzed, as discussed later in this section. Due to the challenging nature of the learning scenario, and to provide a fair comparison with the baseline algorithm, a simplified setup was developed. The objective is to compare the baseline and best optimization algorithms while the robot tries to describe a straight path of 12m in length. The turning parameter is computed at every iteration to maximize the path's linearity using a reactive proportional controller. After 12m, if the robot has not fallen, the episode is considered successful.

Figure 5.12 compares the baseline's mean speed for the entire path with its improved version using the Arms & COM height controller. In both cases, the speed values were averaged for 500 successful episodes. In comparison with Table 5.4, the baseline algorithm improved its mean speed from 0.602m/s to 0.704m/s. The Arms & COM height optimization went from 0.956m/s to 0.958m/s, indicating that the robot has a virtually constant speed, whether turning or not. The improvement of the optimized version in relation to the baseline is about 36%.

The total angular displacement performed by certain groups of joints during a successful episode was analyzed, as this metric provides a reasonable indicator of energy consumption, considering that the actuator's load is not disclosed by the server. Figure 5.13 compares the displacement sum of the arms joints with the displacement sum of all robot joints (except for the head). This analysis was performed for different stages of the linear path described by the robot. In the first meter, as expected, the robot spends more energy while gaining momentum, and then it stabilizes. Considering only the arms joints (shoulder roll, shoulder pitch, elbow yaw, elbow roll), the average angular displacement for the entire episode rose 49%. Despite this result, the same analysis performed for all joints yields

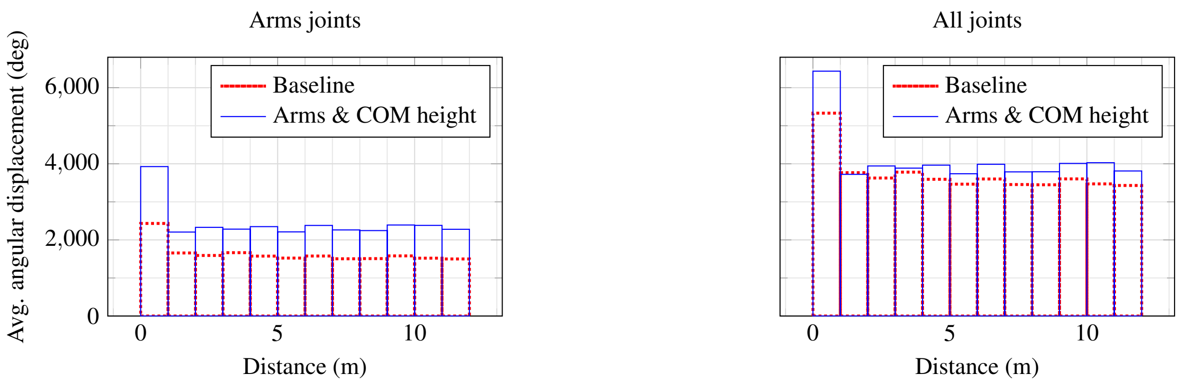


Figure 5.13: Angular displacement performed by all arms joints (sum) during an episode, averaged for 500 successful episodes (on the left). The linear path described by the robot was divided into sections of 1m, which are represented by each bar. The baseline is represented by the dotted red bars while the optimized version is represented by the solid blue bars. The same sort of analysis for all joints is depicted on the right.

an increment of only 10%. Therefore, without considering stability gains, the ratio of relative speed improvement to relative displacement increment in successful episodes is 3.6. In essence, the robot became much more energy-efficient, as a small rise in energy consumption led to a considerably faster gait.

5.9 Discussion

Simulation results showed that the framework is able to generate a fast and stable omnidirectional walk and improve its performance by learning how to control the arms and the height of the COM. Indeed, the results showed that providing a tight coupling between analytical approaches and ML improves the performance considerably. In the remainder of this section, we point out the features and limitations of the proposed framework and provide comparisons with the results of previous works.

5.9.1 Features

- **Architecture:** the modular architecture of the proposed framework provides some important properties such as reducing the complexity and increasing the flexibility. In comparison with approaches that are based on heuristic methods [22, 31, 66, 67, 68, 88] or based on learning from scratch [66, 72, 73], our framework is able to migrate to different humanoid platforms with small changes to the control module.

- **Computational efficiency:** unlike the approaches presented in [19, 59, 60, 76, 89] which are based on online optimization (e.g., MPC), our controller was designed on top of an offline optimization algorithm. Therefore, it does not need powerful computational resources and it can be deployed on any platform easily.

- **Considering the upper body dynamics:** most of the approaches presented in the literature used LIPM as their dynamics model, mainly due to its linearity and simplicity. Unlike LIPM-based approaches, we take into account the robot's upper body dynamics and we showed how this consideration helps to enhance the stability and speed of the robot while improving the energy efficiency as a ratio of mean speed to total angular displacement.

Table 5.5: Comparison Results

	Maximum speed	Ability of changing direction
[90]	1.180m/s	YES
Proposed framework	0.956m/s	YES
[71]	0.805m/s	YES
[88]	0.770m/s	YES
[21]	0.590m/s	YES
[81]	3.910m/s	NO
[73]	2.500m/s	NO
[84]	1.340m/s	NO
[23]	0.550m/s	NO
[22]	0.510m/s	NO

- **Release the height of COM constraint:** LIPM-based approaches assume a fixed vertical position for the COM. According to this assumption, the knee joints have to be bent while the robot is walking, which is harmful to the knee joints and causes additional energy consumption. Additionally, walking with bent knees is not very human-like. We released this constraint by assuming a sinusoidal movement for the vertical position of the COM. We showed that this assumption not only cancels the explained limitations but it also improves the stability.

- **Performance:** to have an entirely fair comparison, the performance of our framework should be compared with other frameworks in the same scenario and simulator. To do so, we took into consideration the maximum forward speed, and our proposed framework provides a faster walk than the agents in [21, 22, 23, 71, 88] and slower than [73, 81, 84, 90]. However, some of the faster examples are solely focused on sprinting forward, without the basic ability to change direction [73, 81, 84]. The comparison results are summarized in Table 5.5.

- **Learning flexibility:** we believe that a humanoid robot should be able to learn from experience, not only to create a new behavior but also to improve its skills. Additionally, it should be able to reuse its knowledge in different scenarios. Learning how to control the arms and the COM height had a positive effect under different conditions in which the robot was not explicitly trained. The robot preserved its stability and speed when subjected to constant orientation adjustments to move in a straight line. Furthermore, we kept the learning module on top of the others to allow situations where generalization is not a conceivable solution. This is an improvement over learning from scratch approaches, as it builds upon a logical and reliable initial solution. This analytical layer is less prone to modeling errors than the learning layer, which is critical when transferring the knowledge to a real robot. After turning the control module to new conditions, the neural network can be partially retrained by leveraging existing knowledge of similar tasks. This architecture allows for a plethora of modular optimizations aimed at stability, speed, energy efficiency, path optimization, context awareness problems (including prevention and recovery), etc.

- **Controller and Robustness:** some approaches [21, 74] used a dynamics model just to generate a feed-forward walk and did not consider any controller to track the references. Other approaches that are based on learning from scratch [66, 72, 73] do not take into account any controller explicitly. Instead, they use a learning algorithm to develop a controller implicitly. Unlike these approaches, we believe that a robust controller is an essential module of a walking framework due to the unstable nature of a humanoid robot. More specifically, when deploying

Table 5.6: Summary of the results in the maximum speed scenario.

Dynamics model	ML algorithm	maximum speed
LIPM	GA	0.590m/s
LIPM + vertical motion of COM	GA	0.630m/s
LIPM + vertical motion of COM+ Torso	GA	0.866m/s
LIPM	GA+PPO	0.710m/s
LIPM + vertical motion of COM	GA+PPO	0.741m/s
LIPM + vertical motion of COM+ Torso	GA+PPO	0.956m/s

the framework on a real robot, using a closed-loop walking is the best approach because it provides a better stability guarantee. Moreover, as we showed, the ML algorithms can be used on top of this controller to improve its performance. The summary of the results in the maximum speed scenario is presented in the Table 5.6.

5.9.2 Limitations

- **Swing leg dynamics:** the legs of a humanoid robot are generally composed of six joints and have non-negligible masses. In our dynamics models, the swing leg is considered to be massless, which affects the controller performance. Taking into account the inertia and mass of the swing leg can minimize tracking errors and improve the controller's performance [1, 91].

- **Reality Gap:** the disparity between reality and simulation is a matter of concern when employing offline ML techniques. Learning to improve the upper body efficiency took between 20 and 26 million time steps. Other works have shown the optimization of robotic tasks, such as squatting [76], using RL combined with an analytical controller, in under 10M time steps. Haarnoja et al. also demonstrated that learning humanoid tasks from scratch can also be performed in about the same period of time [92]. However, it must be noted that these approaches employed distinct environments with different robots, directly influencing the complexity of the task. Learning to run using the NAO robot in SimSpark can take close to 200M time steps [73, 81]. Nevertheless, 20-26 million time steps can still be characterized as poor sample efficiency, as it takes a considerable amount of time and must be performed in a simulated environment. The gap between both worlds largely affects the transferability of knowledge to the real robot. Despite the scientific community's considerable effort to reduce this gap, it remains an issue when dealing with intricate robot models. Additionally, it is not possible to learn directly on the real robot due to the high potential of mechanical damage.

5.10 Summary

In this chapter, we have tackled the problem of developing a robust biped locomotion framework by proposing a tight coupling between an analytical control approach and a reinforcement learning approach. The overall architecture of the framework was composed of six distinct modules which were hierarchically structured. We abstracted the overall dynamics of a humanoid robot into two masses. Then, we used the ZMP concept and some assumptions to represent this dynamics model as a linear state-space system. The system was composed of four states and we explained how it can be used to plan and control the walking reference trajectories. Particularly, the planner was composed of five sub-planners and the controller was formulated as an LQG controller, which is not only robust against uncertainties but also provides a promising solution using offline optimization. We analyzed the performance of the controller in the presence of uncertainties using simulations and the results validated its performance. Moreover, we illustrated how the parametric nature of the framework allows us to use the PPO algorithm on top of an analytical control approach to improve the performance of the framework. Finally, the performance of the proposed framework was validated in several simulated scenarios. The first two scenarios were focused on examining the ability of the framework in generating an omnidirectional walk and finding the maximum velocity of the forward walk. The third scenario was designed to assess the capability of the learning module in improving the framework's performance. The robot learned how to move its arms and COM height to improve stability, speed, and energy efficiency. This limited action space enabled the robot to learn how to walk without falling for much longer periods (almost 29 times longer), while also improving the speed by 60% when walking forward or turning.

In the next chapter, we will design a more accurate dynamics model by considering the mass of the swing leg to improve the framework's performance. Additionally, the controller will be formulated as a Model Predictive Controller scheme subject to some constraints.

Chapter 6

Model Predictive Control Scheme for Locomotion

Biped robots are inherently unstable because of their complex kinematics as well as dynamics. Despite many research efforts in developing biped locomotion, the performance of biped locomotion is still far from the expectations. This chapter proposes a model-based framework to generate stable biped locomotion. The core of this framework is an abstract dynamics model which is composed of three masses to consider the dynamics of stance leg, torso, and swing leg for minimizing the tracking problems. According to this dynamics model, we propose a modular walking reference trajectories planner which takes into account obstacles to plan all the references. Moreover, this dynamics model is used to formulate the controller as a Model Predictive Control (MPC) scheme which can consider some constraints in the states of the system, inputs, outputs, and also mixed input-output. The performance and the robustness of the proposed framework are validated by performing several numerical simulations using MATLAB. Moreover, the framework is deployed on a simulated torque-controlled humanoid to verify its performance and robustness. The simulation results show that the proposed framework is capable of generating biped locomotion robustly.

6.1 Introduction

Humanoid robots are more adapted to our real environment for helping us to perform our daily-life tasks. Developing a robust walking framework for humanoid robots has been researched for decades, and it is still a challenging problem in the robotics community. The complexity of this problem derives from several different aspects like considering an accurate hybrid dynamics model, designing appropriate controllers, and formulating suitable reference trajectory planners. The wide range of applications for these types of robots motivates researchers to tackle such a complex problem using different approaches [1, 71, 93, 94, 95]. As we explained in the previous chapters, some of these approaches try to generate walking patterns by generating rhythmic patterns for each limb. Some other are inspired by neurophysiological studies on humans and animals. Moreover, some approaches are designed based on learning from scratch and mostly are based on RL algorithms [73, 96]. Unlike these approaches, the fundamental core of the model-based approach is a dynamics model of the robot. The main idea of these approaches is modelling the overall behavior of the system abstractly, then planners and controllers will be formulated according to this model. First question in designing a model-based walking is *how accurate should the model be?* To answer this question, two points of view exist: (i) using a whole-body dynamics model, (ii) using an abstract model. We believe that for designing a dynamics model of a system, a trade-off between accuracy and simplicity should be taken into account. Although the whole body dynamics model is more accurate than an abstract model, it is not only platform-dependent but also needs a high computational effort. The main idea behind using an abstract model is to fade the complexity of the control system and organizing the control system as a hierarchy. In hierarchical control approaches, a simplified dynamics model is used to abstract the overall behaviors of the system, and then these behaviors will be converted to individual actuator inputs using a detailed full-body inverse dynamics [1]. In such approaches, the overall performance of the system depends on the matching between the abstract model and the exact model.

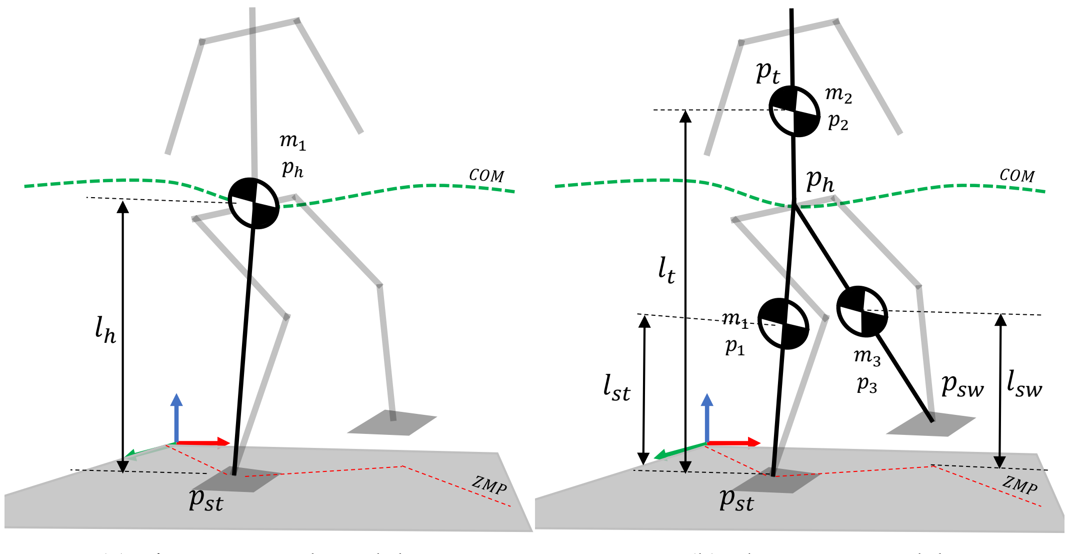
In this chapter, we proposed a modular walking framework capable of generating robust locomotion for humanoid robots. This chapter follows our previous works on model-based biped locomotion frameworks in the previous chapters. In this framework, the overall dynamics of the robot will be modelled using three masses which take into account the dynamics of the legs and the torso. According to this model, the problem of dynamic locomotion will be formulated as a linear MPC to predict the behavior of the system over a prediction horizon and to determine the optimum control inputs. Additionally, the process of generating walking reference trajectories will be decomposed into three levels including (i) path and footstep planning, (ii) planning ZMP, hip, and swing reference trajectories, and (iii) planning a set of reference trajectories for the controller. Besides, the proposed framework will be deployed on a full-size simulated torque-controlled humanoid robot and we will conduct a set of simulations to assess its robustness, performance, and portability. Furthermore, the results of several simulations will be presented to show the performance and the robustness of the proposed walking scheme. Indeed, our contribution is twofold. First, the development of a modular framework that reduces the complexity and increases the flexibility of generating robust locomotion. Second, formulating the walking controller as a linear MPC based on the three-mass model. The remainder of this chapter is structured as follows: Section 6.2 gives an overview of the related work. The overall architecture of the framework will be presented in Section 6.3. Later, in Section 6.4, the reference trajectory planners will be presented. Afterwards, the three-mass dynamics model will be reviewed in Section 6.5 and then we will explain how this model will be used to design a walking controller based on the MPC scheme in Section 6.6. In Section 6.7, a set of simulation scenarios will be designed and executed to examine the tracking performance and robustness of the proposed controller. Afterwards, in Section 6.8, a baseline framework based on LIPM will be used to compare and highlight the effectiveness of the proposed framework. Finally, a summary of this chapter will be presented in Section 6.9.

6.2 Related Work

Several simplified dynamics models have been proposed that abstract the overall dynamics of humanoid robots. In this section, some of these models will be reviewed briefly and then some recent works that used these models either to generate robust locomotion or to develop push recovery strategies will be presented.

6.2.1 Dynamics Models

As we mentioned in previous chapters, LIPM is one of the common dynamics models in the literature and its popularity originates from its linearity and simplicity and also from its ability to generate a feasible, fast, and efficient trajectory of the COM [97]. This model describes the dynamics of a humanoid robot just by considering a single mass that is connected to the ground via a mass-less



(a) Linear Inverted Pendulum (b) Three-mass model

Figure 6.1: Abstract dynamics models: (a) represents the Linear Inverted Pendulum Model (LIPM) which abstracts the overall dynamics of a humanoid robot into a single mass with restricted vertical motion; (b) represents the three-mass model which takes into account the dynamics of the legs and the torso with restricted vertical motions.

rod (see Figure 6.1 (a)). In this model, the single mass is assumed to move along a horizontal plane, and based on this assumption, the motion equations in sagittal and frontal planes are decoupled and independent. Several studies used this model to develop an online walking generator based on optimal control [61] and also linear MPC [98, 99, 100]. Several extensions of LIPM have been proposed to increase the accuracy of this model while keeping the simplicity level [1, 8, 9, 101]. The three-mass model is one of the extended versions of LIPM [9, 49, 102]. As it is shown in Figure 6.1 (b), this model considers the masses of legs and the torso to increase the accuracy of the model. It should be mentioned that to keep the model linear, the vertical motions of the masses are considered to be smooth and the vertical accelerations are neglected.

Kajita et al. [97] introduced the Three Dimensional Linear Inverted Pendulum Model (3D-LIPM) and explained how this model can be used to generate walking. Afterwards, in [7], they used the concept of ZMP to develop a control framework based on preview control to generate stable locomotion. The performance of their framework has been validated using several simulations.

Albert and Gerth [47] extended LIPM by considering the mass of the swing leg to improve gait stability and they named it Two-Mass Inverted Pendulum Model (TMIPM). In their approach, the swing leg and the ZMP trajectories should be generated first, then the trajectory of the COM will be generated by solving a linear differential equation. They proposed an extended version of TMIPM which considers the mass of the thigh, the shank, and the foot of the swinging leg which has been called Multiple Masses Inverted Pendulum Model (MMIPM). TMIPM and MMIPM are more accurate than LIPM, they need an iterative algorithm to define the COM trajectory due to the dependency of masses' motions on each other.

Shimmyo et al. [9] extended the LIPM by considering three masses to decrease the modeling error and improve the performance. In their model, the masses were located on the torso, the right leg, and the left leg. They made two assumptions to design a preview controller which were considering the constant vertical position for the masses and also constant mass distribution. The performance of their approach has been validated through several experiments.

Faraji and Ijspeert [1] proposed a dynamics model which is composed of three linear pendulums to model the dynamics of the legs and the torso and they named it 3LP. They argued that, due to the linearity of this model, it is fast and computationally similar to LIPM such that it can be used in modern control architectures from the computational perspective. The performance of their model has been proven using a set of simulations. Simulation results showed that their framework was able to generate robust walking with a wide range of speeds without requiring off-line optimization and tuning of parameters.

Several extended versions of LIPM have been proposed to consider the momentum around the COM [37] but they do not take into account the dynamics of the legs specifically. In these models, the legs are considered to be mass-less and their motions do not have any effect on dynamics. In the previous chapters, we extended the dynamics model presented in [37] by considering the mass of the stance leg, and we explained how this model can be represented by a differential equation which can be used to design a controller to plan and track the walking reference trajectories.

6.2.2 Locomotion Framework

Herdt et al. [19] proposed an online walking generator with automatic footstep placement. They used LIPM to formulate the walking pattern generator as a linear MPC. In their framework, instead of considering a set of predefined footsteps, they introduce new control variables to generate footsteps automatically. The performance of their approach has been validated through a set of simulations. The simulation results showed that the framework can track a given reference speed of COM that can be modified at any time.

Brasseur et al. [99] designed a robust linear MPC to generate online 3D locomotion for humanoid robots in a single computation with guaranteed kinematic and dynamic feasibility. They used Newton and Euler equations of motion to model the dynamics of a humanoid robot. They considered some assumptions to reduce the complexity of the model which are almost the same as LIPM's assumptions, but unlike LIPM, they take into account the vertical motion of the COM to generate more efficient locomotion in terms of energy and speed. The performance of their approach has been tested in two simulation scenarios including walking on a flat train and climbing stairs using a simulated HRP-2 robot. The simulation results validated the performance of their approach.

Jianwen Luo et al. [103] proposed a hierarchical control structure to generate robust biped locomotion. Their structure is composed of three independent control levels. At the highest level, they used A^* to plan a path to the destination, and then they designed an MPC based on LIPM to generate the COM trajectory. At the middle level, they designed a hybrid controller which is composed of two controllers to control the oscillation and to eliminate the chattering problem. At the lowest level, they used a whole-body operational space (WBOS) controller to compute the joint torques analytically. The effectiveness of their approach has been validated through simulation.

Most of the aforementioned works use LIPM as their dynamics model and do not take into account the dynamics of the swing leg and the torso. In this chapter, we propose a modular framework to generate robust biped locomotion which takes into account the dynamics of the torso and swing leg. Particularly, we use the three-mass model as the core of this framework, and based on that, we design an MPC controller to formulate walking as a set of quadratic functions and a set of time-varying constraints based on states, inputs, and outputs of the system. This controller is not only able to track the reference trajectories optimally (regarding an objective function) even in presence of measurement noise but also it is robust against uncertainties such as external disturbances. Furthermore, we will perform a set of simulations to explore the impacts of the proposed controller and to validate the performance of the framework.

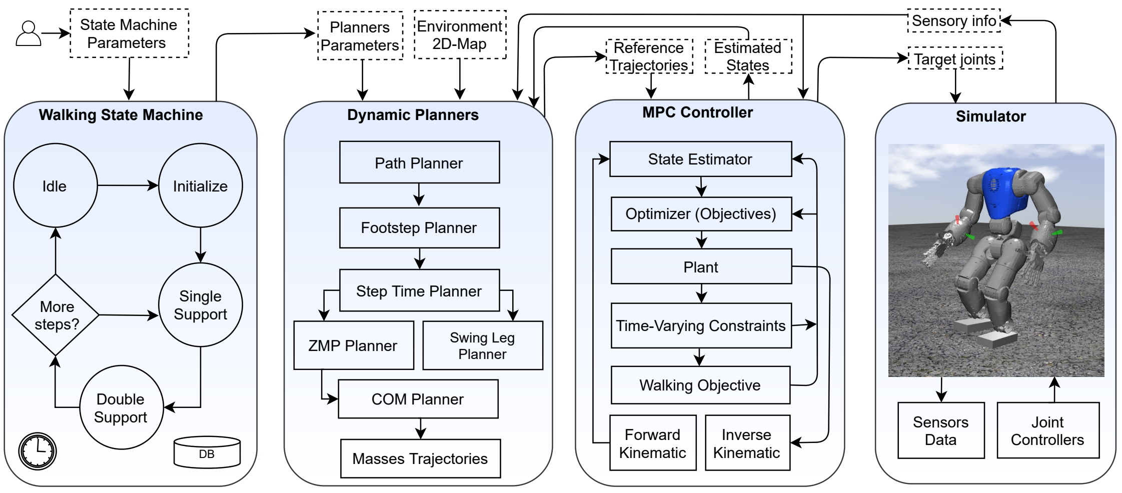


Figure 6.2: Overall architecture of the proposed MPC-based framework. This framework is composed of four main modules which are connected together to generate robust locomotion. The highlighted boxes represent the functional blocks and the exchanged information among the modules are represented by the white boxes.

6.3 Architecture

This section is focused on presenting the overall architecture of the proposed framework which is depicted in Figure 6.2. This framework is composed of four main modules which are coupled together to generate robust biped locomotion. The first module is the `Walking State Machine` which is responsible for controlling the overall process of walking. According to the periodic nature of the walking, this state machine is composed of four main states such that state transitions are triggered based on an associated timer and also state conditions. In the *Idle state*, the robot stands and is waiting for a start walking command. Once a walking command is received, the state transits to the *Initialize state* and the robot shifts its COM to the first support foot to be ready for taking the first step. In the *Single Support State* and the *Double Support State*, the planners' parameters (e.g., robot's target position and orientation, max speed, etc.) will be updated. These parameters along with a 2D map of the environment and the sensory information provided by the `Simulator` as well as the estimated states provided by the `MPC Controller` will be fed into the `Dynamic Planners` to generate a set of reference trajectories. `Dynamic Planners` starts by generating a collision-free path which will be used to plan a set of reference trajectories including footsteps, ZMP, swing, COM, and masses trajectories. Finally, these references will be fed into the `MPC Controller`. These trajectories along with the sensory information (e.g., joint encoders, IMU, robot's position and orientation, etc.) provided by the `Simulator` will be used to estimate the robot's states. The estimated states and a set of objectives are used by the optimizer to specify the control inputs subject to a set of time-varying constraints for tracking the generated reference trajectories. The corresponding joint motions will be generated using the *Inverse Kinematics* which takes into account kinematic feasibility constraints. The target joint setpoints will be fed into the `Simulator` which is responsible for simulating the interaction of the robot with the environment and producing sensory information as well as the global position and orientation of the robot in the simulated environment to be used by `Dynamic Planners`. In the following sections, the details of each module will be explained separately, and then a set of simulation scenarios will be designed and conducted to validate the performance of the framework.

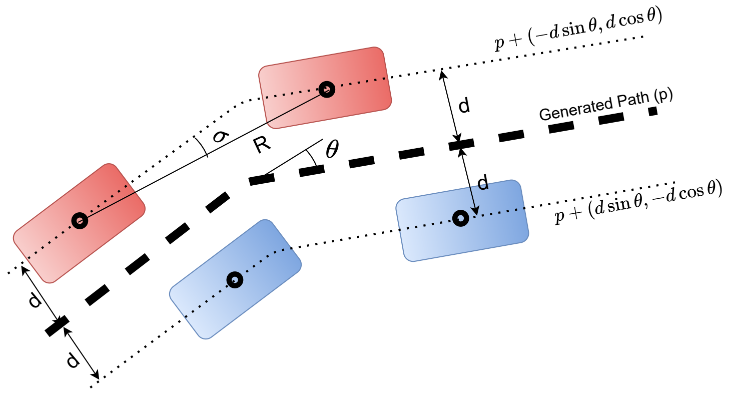

$p + (-d\sin\theta, d\cos\theta)$
d
Generated Path (p)
σ
R
θ
d
$p + (d\sin\theta, -d\cos\theta)$
d
d

Figure 6.3: Example of planning footsteps. After generating a collision-free path (thick dashed black line), two lines parallel to the path will be generated (dotted black lines) which are d m away from the generated path where $2d$ is the distance between feet. Then, the left and right footsteps (red and blue rectangles) should be selected based on these paths so that R m away from its previous one. Afterwards, σ will be determined based on the generated footsteps using the inverse tangent function.

6.4 Dynamic Planners

The `Dynamic Planners` module is responsible for the planning of the reference trajectories according to the input parameters as well as the environment's structure. To reduce the complexity of the planning, the planning process is divided into a set of sub-planners which are solved separately and connected hierarchically (see Figure 6.2). In the rest of this section, we will explain how the reference trajectories will be generated.

6.4.1 A* Footstep Planning

Footstep planning is generally based on a graph search algorithm with a rich history [104, 105]. In our target framework, the footstep planning is composed of two stages: *(i)* generating a collision-free 2D body path; *(ii)* planning the footsteps based on the generated path. In the first stage, the environment is modeled as a 2D grid map consisting of cells that are marked as free or occupied. In this work, the size of the cell is considered to be 0.01m^2, the height of obstacles is not considered, and the robot can not step over them. In order to have a safe distance from the obstacle, the size of the obstacles is considered larger than the real size (scale = 1.1). In this stage, after modeling the environment, the A* search method is applied to find an optimum path over the free cells from the current position of the robot to the goal. Euclidean distance to the goal is used to guide the search. In the second stage, the footsteps should be generated according to the generated path. To do so, a state variable is defined to describe the current state of the robot's feet:

$$s = (x_l, y_l, \theta_l, \phi_l, x_r, y_r, \theta_r, \phi_r) \tag{6.1}$$

where $x_l, y_l, \theta_l, x_r, y_r$ and θ_r represent the current position and orientation of the left and the right feet, respectively. ϕ_l and ϕ_r represent the state of each foot which is 1 if the foot is the swing foot, and -1 otherwise. According to the nature of walking which is generated by moving the right and left legs, in alternation, a step action is parameterized by a distance and an angle from the swing foot position at the beginning of step $a = (R, \sigma)$. Based on the current state and the generated path, action should be taken. In this chapter, we consider a constant step size and σ will be determined according to the generated path and the current state of the feet. Figure 6.3 shows an example of planning footsteps after generating the collision-free path.

It is worth mentioning that in some scenarios, like teleoperation tasks, an operator wants to determine the actions without any autonomous path planning. In such a scenario, the step positions will be planned just according to the input command. It should be mentioned that the action is always passed through a first-order lag filter to ensure a smooth update. By applying the selected action, the state transits to a new state, $s' = t(s,a)$. It should be noted that after each transition, the current footstep will be saved ($f_i \quad i \in \mathbb{N}$), also ϕ_l and ϕ_r will be toggled. After generating the footsteps, the step time should be planned based on the robot's target speed.

6.4.2 ZMP, Hip and Swing Reference Trajectories

Studies on human locomotion showed that, while a human is walking, ZMP moves from heel to the toe during the single support phase and it moves towards the next supporting foot during the double support phase [41, 106]. In this chapter, we do not consider the ZMP movement during the single support, but instead, we keep it in the middle of the support foot to prevent it from approaching to the support polygon edge and to avoid the loss of equilibrium of the robot in some situation like unexpected external disturbances. Thus, the ZMP reference trajectory planning can be formulated based on the generated footsteps as follows:

$$p_{st} = \begin{cases} f_i & 0 \leq t < T_{ss} \\ f_i + \frac{SL(t-T_{ss})}{T_{ds}} & T_{ss} \leq t < T_{ss} + T_{ds} \end{cases}, \tag{6.2}$$

where $p_{st} = [p_{st}^x \quad p_{st}^y]^\top$ is the generated ZMP and t, T_{ss}, T_{ds} represent the time, duration of single and double support phases, respectively. $SL = [SL^x \quad SL^y]$ is a vector that represents the step length and step width which are determined based on (R, σ) and $f_i = [f_i^x \quad f_i^y]$ represents the planned foot steps on a 2D surface. It should be noted that t will be reset at the end of each step $(t \geq T_{ss} + T_{ds})$.

After generating the ZMP reference trajectory, the hip trajectory will be planed according to the generated ZMP. To do that, we assume that the COM of the robot is located at the middle of the hip, and, based on this assumption, the overall dynamics of the robot is firstly restricted into COM and then the reference trajectory for the hip will be generated using the analytical solution of the LIPM as follows:

$$p_h(t) = p_{st} + \frac{(p_{st} - p_{h_f})\sinh\big((t-t_0)\omega\big) + (p_{h_0} - p_{st})\sinh\big((t-t_f)\omega\big)}{\sinh((t_0 - t_f)\omega)}, \tag{6.3}$$

where t_0 and t_f represent the beginning and the ending times of a step, $\omega = \sqrt{\frac{g}{l_h}}$ is the natural frequency of LIPM and p_{h_0}, p_{h_f} are the corresponding positions of COM at these times, respectively.

After generating the ZMP and the hip trajectories, the swing trajectory should be planned. To have a smooth trajectory during lifting and landing of the swing leg, a Bézier curve is used to generate this trajectory according to the generated footsteps and a predefined swing height.

6.4.3 Masses Reference Trajectories

The trajectories of the masses can be easily generated based on the geometric relations between the generated hip, ZMP, and swing leg trajectories. According to Figure 6.1 (b), the mass of the stance leg is located in the middle of the line between the ZMP and the hip. Similarly, the mass of

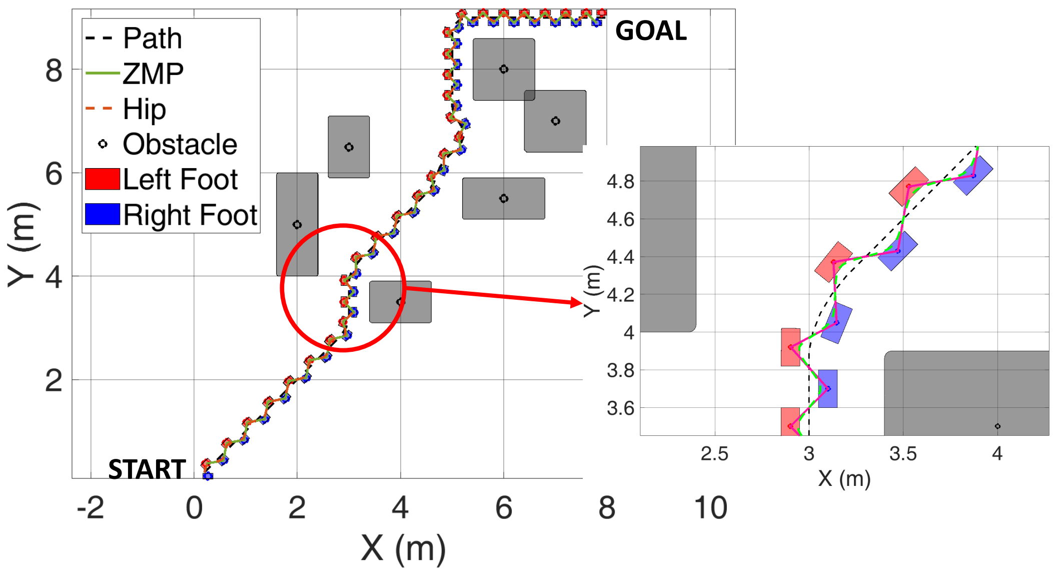


Figure 6.4: An example walking references trajectories: the gray rectangles represent the occupied cells that the robot can not step; the black-dashed line represents the output of the A* path planner ($R = 0.1$m); red and blue rectangles represent the footsteps which are generated based on the output of the A*; the magenta line is the ZMP which is generated based on the outputs of the footstep planner; the lime-dashed line is the hip reference trajectory.

the swing leg is located in the middle of the line between the swing foot and the hip. To show the performance of the planners presented in this section, an example path planning scenario has been set up, which is depicted in Figure 6.4. In this scenario, the robot stands at the *START* point and wants to reach the *GOAL* point. The planning process is started by generating an optimum obstacle-free path (black-dashed line). As it is shown in Figure 6.4, the generated path is far from the obstacles enough and, based on this path, the footsteps, corresponding ZMP and hip trajectories have been generated successfully.

6.5 Dynamics Model

In this section, the concept of ZMP (presented in Chapter 2) will be used to define the overall dynamics of a biped robot as a state-space system.

6.5.1 Three-mass Dynamics Model

The three-mass model abstracts the dynamics model of a biped robot by considering three masses. As shown in Figure 6.1 (b), the masses are placed at the legs and the torso of the robot. To simplify and linearize the model, each mass is restricted to move along a horizontal plane. According to this assumption, the ZMP equations are independent and equivalent in the frontal and sagittal planes. Thus, in the remainder of this chapter, just the equations in the sagittal plane will be considered. Based on (3.4), a state-space system can be defined to analyze the behavior of the system:

$$
\begin{aligned}
&\frac{d}{dt}\underbrace{\begin{bmatrix} c_1^x \\ \dot{c}_1^x \\ \ddot{c}_1^x \\ c_2^x \\ \dot{c}_2^x \\ \ddot{c}_2^x \\ c_3^x \\ \dot{c}_3^x \\ \ddot{c}_3^x \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1^x \\ \dot{c}_1^x \\ \ddot{c}_1^x \\ c_2^x \\ \dot{c}_2^x \\ \ddot{c}_2^x \\ c_3^x \\ \dot{c}_3^x \\ \ddot{c}_3^x \end{bmatrix}}_{X} + \underbrace{\begin{bmatrix} 0&0&0 \\ 0&0&0 \\ 1&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&1&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&1 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} \dddot{c}_1^x \\ \dddot{c}_2^x \\ \dddot{c}_3^x \end{bmatrix}}_{U} \\
&y = \underbrace{\begin{bmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ \frac{m_1}{M}&0&\frac{-m_1z_1}{Mg}&\frac{m_2}{M}&0&\frac{-m_2z_2}{Mg}&\frac{m_3}{M}&0&\frac{-m_3z_3}{Mg} \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} c_1^x \\ \dot{c}_1^x \\ \ddot{c}_1^x \\ c_2^x \\ \dot{c}_2^x \\ \ddot{c}_2^x \\ c_3^x \\ \dot{c}_3^x \\ \ddot{c}_3^x \end{bmatrix}}_{X},
\end{aligned}
\tag{6.4}
$$

where g represents the acceleration of gravity, c_k and $\ddot{c}_k$ $(k = 1,2,3)$ denote the ground projection of the position and acceleration of each mass, z_k and $\ddot{z}_k$ are vertical position and vertical acceleration of each mass, respectively. $\dddot{c}_1^x$, $\dddot{c}_2^x$ and $\dddot{c}_3^x$ are the manipulated variables in jerk dimension, M is total mass of the robot, m_1, m_2 and m_3 represent the mass of stance leg, torso and swing leg, respectively. Based on the measurement equation (y), the positions of stance leg, swing leg, and also ZMP are measured at each control cycle. In the next section, the system will be discretized to discrete-time implementation and we will explain what type of walking objectives and constraints should be considered to generate stable dynamic walking.

6.6 Online Walking Controller Based on MPC

In this section, the problem of the online walking controller is formulated as a linear MPC which is not only robust to uncertainties but also able to consider some constraints in the states, inputs, and outputs. To do that, firstly, the presented continuous system (6.4) should be discretized for implementation in discrete time. Afterwards, the walking objective will be formulated as a set of quadratic functions and finally, walking constraints will be formulated as linear functions of the states, inputs, and outputs.

6.6.1 Discrete Dynamics Model

To discretize the system, we assume that $\ddot{c}_1, \ddot{c}_2$ and $\ddot{c}_3$ are linear and, based on this assumption, $\dddot{c}_1, \dddot{c}_2$ and $\dddot{c}_3$ are constant within a control cycle. Thus, the discretized system can be represented as follows:

$$
\begin{aligned}
X(k+1) &= A_d X(k) + B_d u(k) \\
y(k) &= C_d X(k)
\end{aligned}
, \tag{6.5}
$$

where k represents the current sampling instance, A_d, B_d and C_d are the discretized version of the A, B and C matrices in (6.4), respectively. Based on this system, the state vector $X(k)$ can be estimated at each control cycle. Thus, according to the estimated states, some objectives, and constraints, the problem of determining the control inputs can be formulated as an optimization problem (a quadratic program (QP)) at each control cycle. The optimization solution specifies the control inputs which are used until the next control cycle. This optimization just considers the current timeslot; to take into account the future timeslots, a finite time horizon is considered but just the current timeslot will be applied. Indeed, the optimization solution determines N_c (control horizon) future moves ($\Delta U = [\Delta u(k), \Delta u(k+1), ..., \Delta u(k+N_c-1)]^\top$) based on the future behavior of the system ($Y = [y(k+1|k), y(k+2|k), ..., y(k+N_p|k)]^\top$) over a prediction horizon of N_p.

6.6.2 Walking Objective

Walking is a periodic locomotion which can be decomposed into two main phases: *single support* and *double support*. In the double support phase, the robot shifts its ZMP to the stance foot and during the single support, its swing leg moves towards the next step position. In order to develop stable locomotion, the robot should be able to track a set of reference trajectories while keeping its stability. As mentioned before, a popular approach to guarantee the stability of the robot is keeping the ZMP within the support polygon. According to (6.4), the position of the stance leg, swing leg, and ZMP are measured at each control cycle. Thus, the following terms should be taken into account in the objective function to keep the outputs at or near the references:

$$\begin{aligned} d_1 &= ||p_z - r_z||^2 \\ d_2 &= ||p_{st} - r_{st}||^2 \\ d_3 &= ||p_{sw} - r_{sw}||^2, \end{aligned} \tag{6.6}$$

where $p_z, r_z, p_{st}, r_{st}, p_{sw}$ and r_{sw} are measured and reference ZMP, and the measure and reference positions of the stance and swing legs, respectively. Moreover, to generate smooth trajectories which are compatible with the robot structure, the control inputs should be considered into the objectives for smoothing the motions:

$$\begin{aligned} d_4 &= ||\dddot{c}_1^x||^2 \\ d_5 &= ||\dddot{c}_2^x||^2 \\ d_6 &= ||\dddot{c}_3^x||^2. \end{aligned} \tag{6.7}$$

According to the explained terms, the following cost function is defined to find an optimal set of control inputs:

$$J(z_k) = \sum_{i=1}^{N_p} \sum_{j=1}^{6} \alpha_j d_j(z_k), \tag{6.8}$$

where k is the current control interval, $z_k^\top = \{\Delta u(k|k)^\top \quad \Delta u(k+1|k)^\top ... \Delta u(k+N_p-1|k)^\top\}$ is the QP decision and α_j represents a positive gain that is assigned to each objective.

6.6.3 Time-Varying Constraints

To ensure the feasibility of the solution that is found by the MPC, a set of constraints should be considered. For instance, the solution should be kinematically reachable by the robot and the ZMP

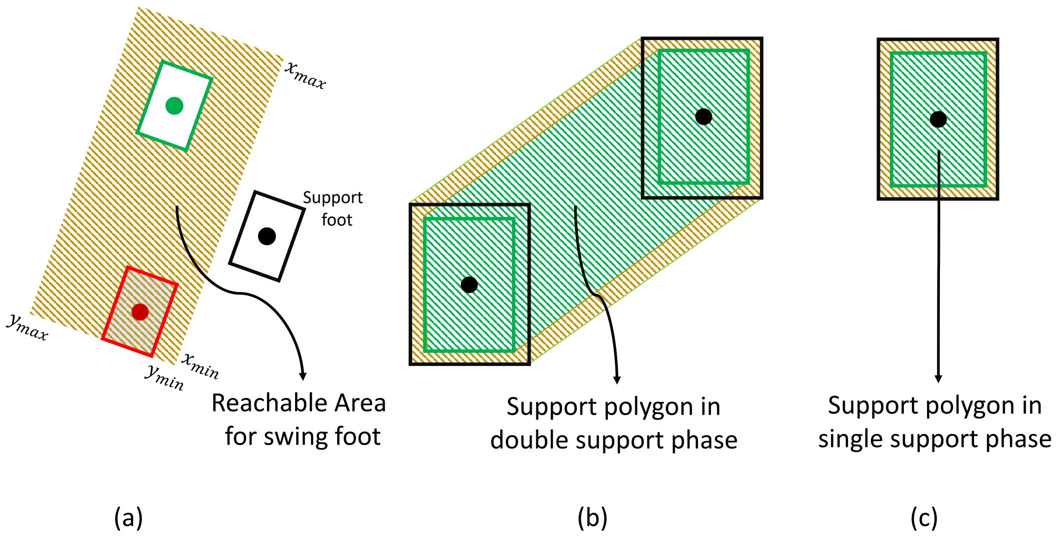

x_{max}
Support foot
y_{max}
x_{min}
y_{min}
Reachable Area for swing foot
Support polygon in double support phase
Support polygon in single support phase
(a)
(b)
(c)

Figure 6.5: Graphical representations of the constraints: **(a)** kinematically reachable area for the swing leg: red rectangle represents the position of the swing leg at the beginning of step, green rectangle represents the landing location of the swing leg; **(b)** ZMP constraint during double support phase: green area is considered as the support polygon to avoid ZMP from being close to the borders; **(c)** ZMP constraint during single support phase.

should be kept inside the support polygon. Generally, a set of mixed input/output constraints can be specified in the following form:

$$Eu(k+j|k)+Fy(k+j|k) \leq G+\varepsilon, \tag{6.9}$$

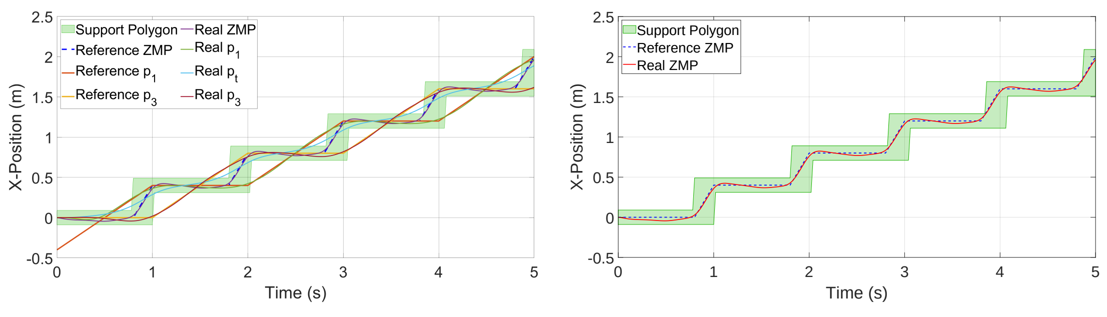
where j varies from 0 to N_p, k is current control cycle, E,F and G are time-variant matrices, where each row represents a linear constraint, $u(k+j|k)$ and $u(k+j|k)$ are vectors of manipulated variables and output variables (stance leg, swing leg and ZMP), respectively. ε is used to specify a constraint to be soft or hard. Additionally, using this equation, the inputs and outputs can be bounded to specified limitations. It should be noted that the constraints in the sagittal plane are similar to those in the frontal plane. In our target framework, the constraints are time-varying and they will be determined at the beginning of each walking phase. Graphical representations of the constraints are depicted in Figure 6.5. Figure 6.5 (a) shows that the landing position of the swing leg should be restricted in a kinematically reachable area. This area can be approximated by a rectangle which is defined as follows:

$$E=\begin{bmatrix}0&0&0\\0&0&0\end{bmatrix}, F=\begin{bmatrix}0&1&0\\0&-1&0\end{bmatrix}, G=\begin{bmatrix}x_{max}\\x_{min}\end{bmatrix}, \varepsilon=\begin{bmatrix}0\\0\end{bmatrix}, \tag{6.10}$$

where x_{min} and x_{max} are defined based on the current position of the support foot and the robot capability. In addition to these constraints, to keep the ZMP within the support polygon, the following constraints are considered:

$$E=\begin{bmatrix}0&0&0\\0&0&0\end{bmatrix}, F=\begin{bmatrix}0&0&1\\0&0&-1\end{bmatrix}, G=\begin{bmatrix}z_{x_{max}}\\z_{x_{min}}\end{bmatrix}, \varepsilon=\begin{bmatrix}0\\0\end{bmatrix}, \tag{6.11}$$

where $z_{x_{min}}$ and $z_{x_{max}}$ are defined based on the walking phase and the size of the foot (please look at Figure 6.5 (b, c)). It should be mentioned that the foot size of the robot is considered a bit smaller (scale = 0.9) than the real foot size to prevent ZMP from being too close to the borders of the support polygon.



(a) (b)

Figure 6.6: The simulation results of analyzing tracking performance: **(a)** represents the positions of the masses while walking; **(b)** represents the reference ZMP and the real ZMP.

6.7 Simulation

In this section, a set of simulation scenarios will be designed to validate the performance and examine the robustness of the proposed framework. We firstly performed three simulations using MATLAB to evaluate the performance and robustness of the framework. Afterwards, we will deploy our framework on a simulated COMAN (a passively compliant humanoid) [107] humanoid robot to validate the performance of the proposed framework on a full-size humanoid robot.

6.7.1 Simulation using MATLAB

To perform the simulations using MATLAB, a humanoid robot has been simulated according to the dynamics model presented in Section 6.5. The most important parameters of the simulated robot and the controller are shown in Table 6.1.

Table 6.1: Parameters used in the MATLAB simulations.

m_1	m_2	m_3	l_{st}	l_t	l_{sw}	foot length	T_s	N_p	N_c	$\alpha_{1,2,3}$	$\alpha_{4,5,6}$
15kg	50kg	15kg	0.5m	1.2m	0.5m	0.1m	0.02s	80	20	20	0.1

Tracking Performance

To check the tracking performance of the controller, the simulated robot is commanded to perform a five-step forward walk (step size = 0.8m and step time = 1s). In this simulation, the simulated robot is considered to be stopped and stands at the beginning of the simulation and the reference trajectories have been generated based on the presented methods in Section 6.4. The planned trajectories at the end of Section 6.4 are used as the input references and the controller should track these trajectories while keeping the stability of the robot. The simulation results are depicted in Figure 6.6. The results show that the controller is able to track the references and the ZMP is always inside the support polygon during walking. Another interesting point in the results is the actual trajectory of the torso (see $Real_{p_t}$ in Figure 6.6). We did not determine any references for the torso explicitly, but it moves as we expected. As it is shown in Figure 6.6(**a**), the torso (p_t) is almost near the support foot during the single support phase, then by starting the double support phase, it moves towards the next support foot.

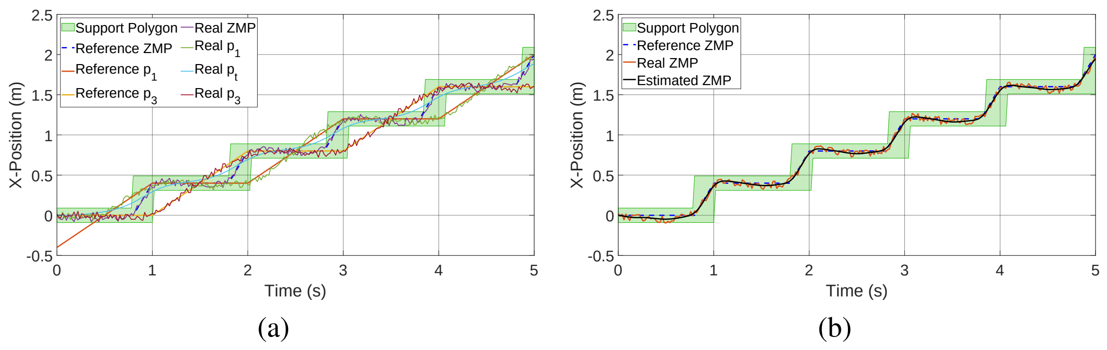


Figure 6.7: The simulation results of examining the robustness w.r.t. measurement noise: **(a)** represents the real and estimated position of the masses; **(b)** represents the real and estimated ZMP.

Robustness w.r.t. Measurement Noise

In the real world, measurements are always affected by noise, therefore they are never perfect. Noise can arise because of many reasons like the simplification in the modeling, discretizing and some mechanical uncertainties (e.g., backlash of gears), etc. A robust controller should be able to estimate the correct state using noisy measurements and minimize the effect of noise. To examine the robustness of the proposed controller regarding measurement noise, the measurements are modeled as a stochastic process by adding Gaussian noise ($-0.05\text{m} \leq v_i \leq 0.05\text{m} \quad i = 1, 2, 3$) to the system output and the previous scenario has been repeated. The simulation results are shown in Figure 6.7. As can be seen, the controller is robust against the measurement noise and it can track the references even in the presence of noise.

Robustness w.r.t. External Disturbance

A robust controller should be able to reject an unwanted external disturbance that can occur in some situations like when a robot hits an obstacle or when it has been pushed by someone. In such situations, the controller cancels the effect of the impact and tries to keep ZMP inside the support polygon by applying a compensating torque. To examine the robustness of the controller w.r.t. external disturbances, an unpredictable external force is applied to the torso of the robot while it is performing the previous scenario. The impact has been applied at $t = 1.6$s and the impact duration is $\Delta t = 100$ms. This simulation was repeated multiple times with different amplitudes (-300N $\leq F \leq 300N$). Moreover, to have realistic simulations, the measurements are confounded by measurement noise ($-0.05\text{m} \leq v_i \leq 0.05\text{m} \quad i = 1, 2, 3$). The simulation results are shown in Figure 6.8. Each plot represents the result of a single simulation. As these plots show, after applying an external force, the ZMP (fictitious ZMP [108]) goes out of the support polygon quickly whose distance from the support polygon edge proportionally relates to the intensity of the perturbation. The controller regains it back and keeps the stability of the robot. We increase the amplitude of the impact to find the maximum withstanding of the controller. After performing these simulations, F=435N and F=-395N were the maximum levels of withstanding of the controller. According to the simulation results, the proposed controller is robust against external disturbance.

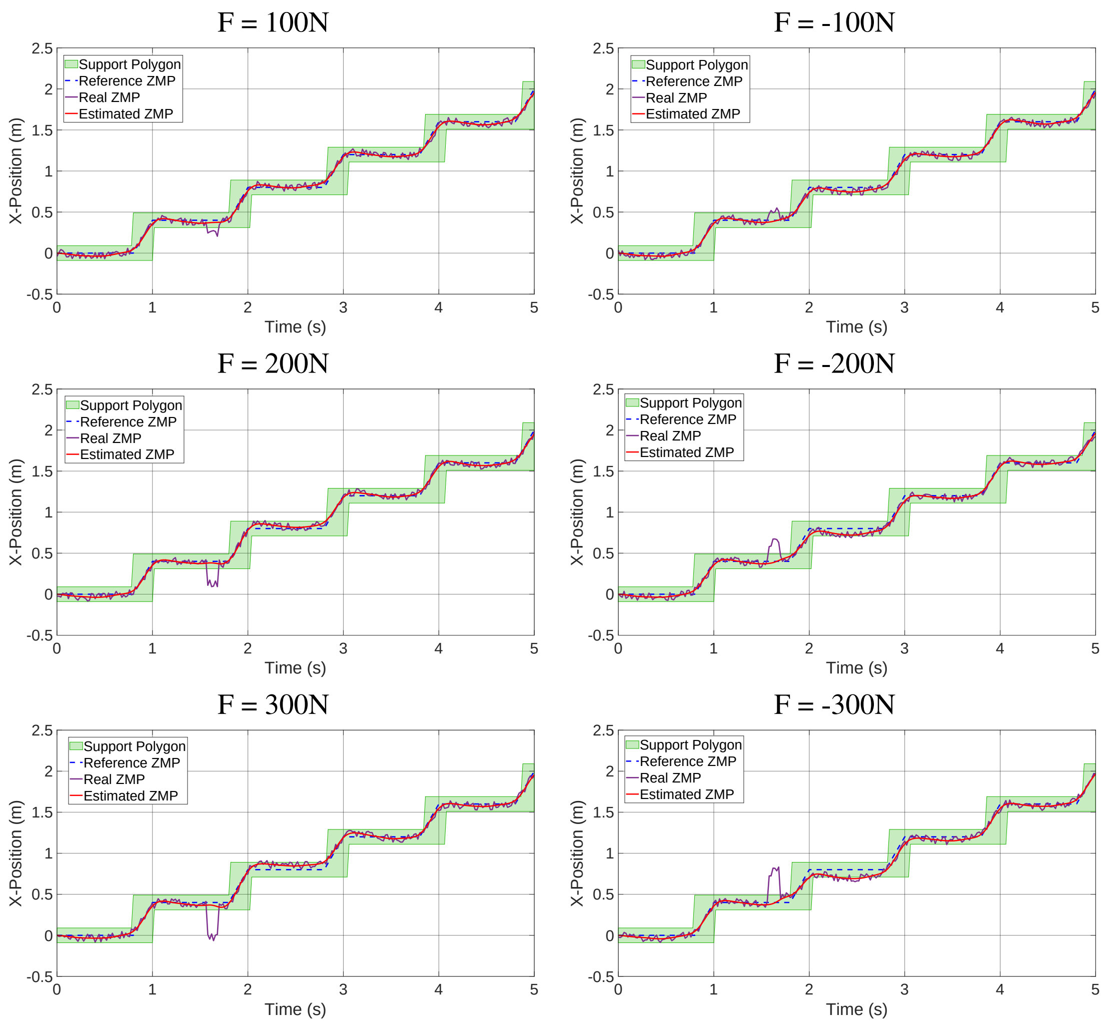


Figure 6.8: The simulation results of examining the robustness w.r.t. external disturbance: each plot represents a single simulation result. As the results show, after applying external force, ZMP (fictitious ZMP) goes out of support polygon for a moment whose distance from the support polygon edge proportionally relates to the intensity of the perturbation.

Checking the Overall Performance

To check the overall performance of the proposed framework, the path planning scenario which has been presented at the end of Section. 6.4, is used (see Figure 6.4). The simulation results are shown in Figure 6.9. The results showed that the planner was able to generate walking reference trajectories and the controller was able to track the generated references successfully. A video of this simulation is available online at `https://youtu.be/zyOmgvaXyuA`.

6.7.2 Simulation using COMAN

To validate the portability and platform-independency of the proposed framework and to show the performance of the framework in controlling a full humanoid robot, we performed a set of simulations using a simulated COMAN humanoid in the Gazebo simulator which is an open-source simulation environment developed by the Open Source Robotics Foundation (OSRF). The simulated robot is

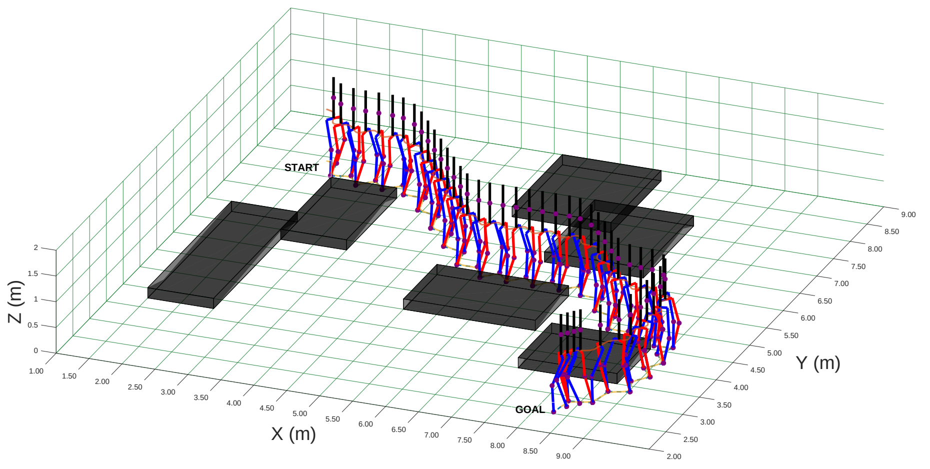


Figure 6.9: The simulation result of examining the overall performance of the proposed planner and controller. The simulated robot starts from START point and should walk towards the GOAL point while avoiding the obstacles.

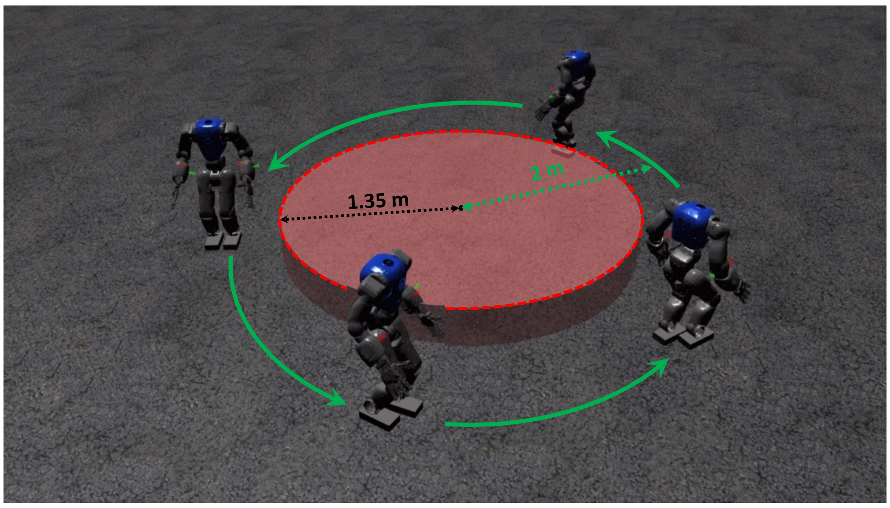
0.95m tall, weighs 31kg, and has 23 DOF (6 per leg, 4 per arm, and 3 between the hip and torso). This robot is equipped with the usual joint position and velocity sensors, an IMU on its hip, and torque/force sensors at its ankles.

Walking Around A Disk

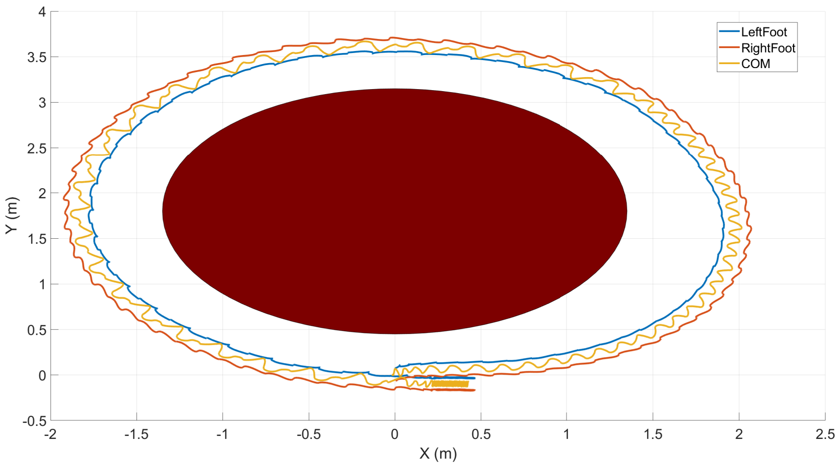
This simulation is focused on testing the performance of the framework for combining turning and forward walking. In this simulation, the robot initially stays next to a large disk of radius 1.35m so that the center of the disk is 1.8m far from the robot. The robot is waiting to receive a start signal and once the signal is generated, the robot should walk around the disk and return to the initial point while trying to keep 2m distance from the center of the disk during walking. In this simulation, we fixed the step size by 0.15m and the turning angle will be determined based on the current position of the robot. The sequences of the experiment are shown in Figure 6.10 (a). In this simulation, the positions of the feet and COM have been recorded and they are depicted in Figure 6.10 (b). The results showed that the framework is able to combine steering and forward walking command to follow a specific path. A video of this simulation is available online at `https://youtu.be/E8PGY05WzIQ`.

Omnidirectional Walking

This scenario is focused on validating the performance of the proposed framework for providing omnidirectional walking. In this scenario, the simulated robot should track deterministic setpoints including step length (X), step width (Y), and step angle (α). At the beginning of the simulation, all the setpoints are zero and the robot is walking in place. At $t = 2$s, the robot is commanded to walk forward ($X = 0.1$m, $Y =$0.0m, $\alpha = 0.0$deg/s); At $t = 10$s, the robot is commanded to walk diagonally ($X = 0.10m, Y = 0.05m, \alpha = 0.0$deg/s); at $t = 20$s, while it is performing diagonal walking, it is commanded to turn right simultaneously ($X = 0.2$m, $Y = 0.05$m, $\alpha = 15$deg/s); Finally, at $t = 32$s, all set points will be reset and the robot will walking in place. An overview of this scenario is depicted in Figure 6.11 (a). During this simulation, the positions of the feet and COM have been recorded to examine the behavior of the COM while walking and they are depicted in Figure 6.11 (b). According

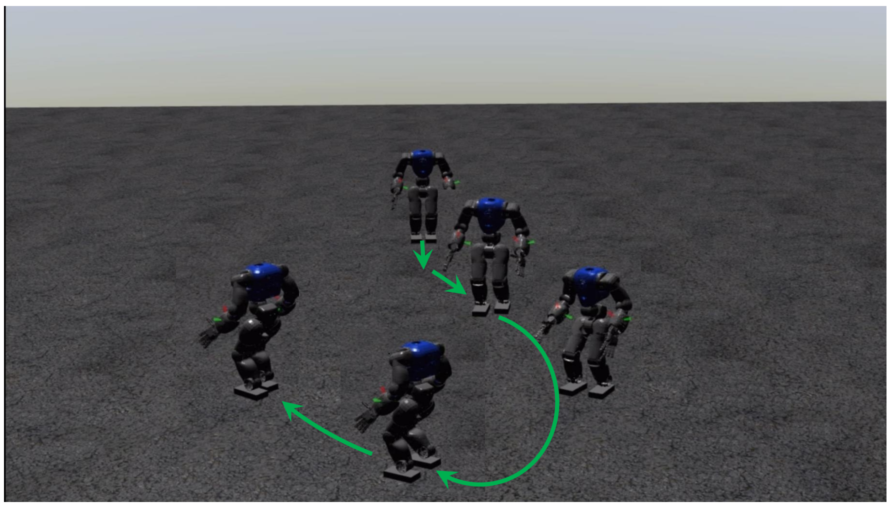


(a)

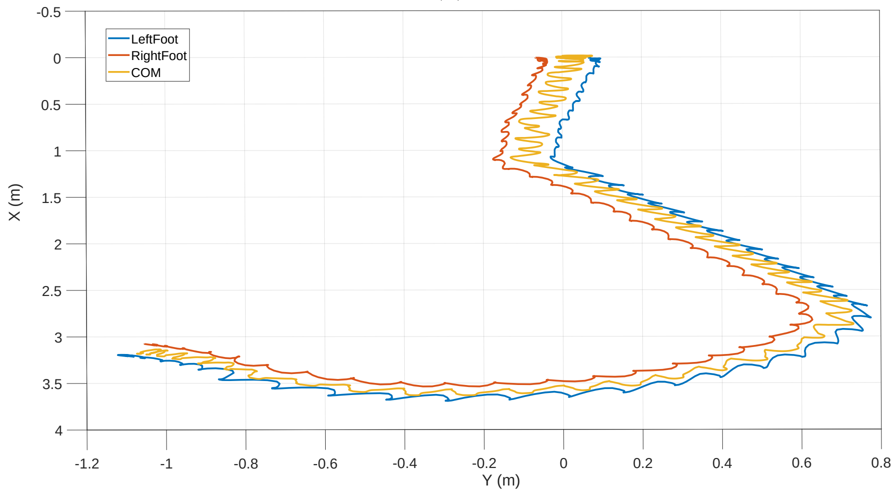


(b)

Figure 6.10: Walking around a disk: the robot should walk around the disk while keeping 2m distance from the center of the disk. **(a)** An overview of the scenario; **(b)** The feet and COM in the XY plane.



(a)



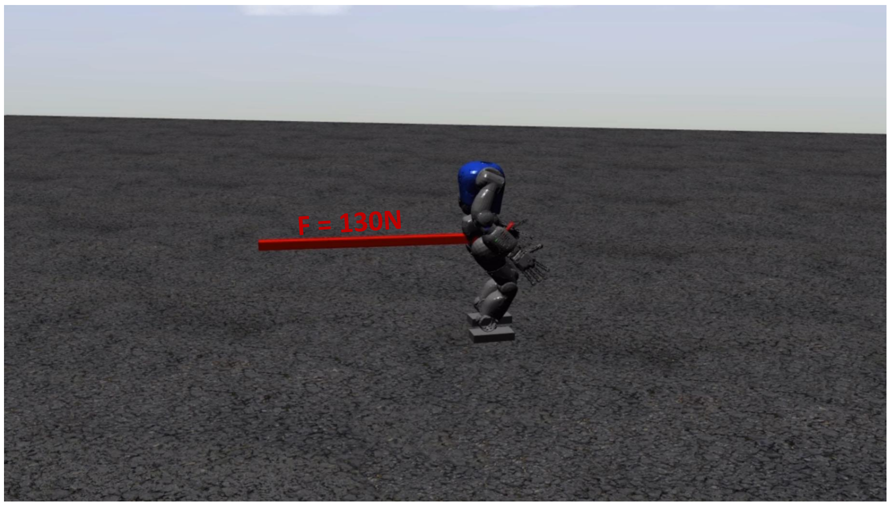
(b)

Figure 6.11: Omnidirectional walking scenario: **(a)** An overview of the omnidirectional walking scenario; **(b)** The feet and COM in the XY plane.

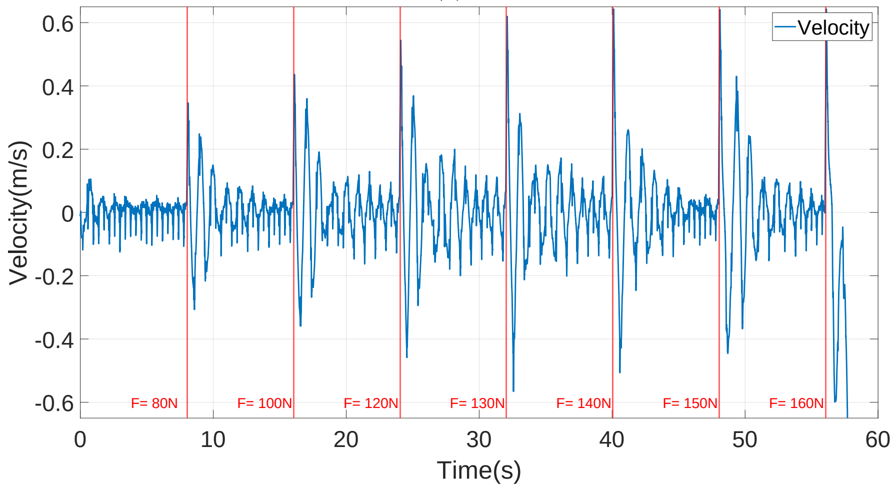
to the recorded data, the COM tends to the support foot during the single support phase and moves to the next support foot during the double support phase. It should be mentioned that we used a first-order lag filter to update the new setpoints to have a smooth updating. The results showed that the framework can combine all the input commands simultaneously to provide an omnidirectional walking. A video of this simulation is available online at `https://youtu.be/1R56m4t6cHs`

Push Recovery

The goal of this simulation is to evaluate the withstanding of the framework in terms of external disturbance rejection. In this scenario, while the robot is walking in place ($steptime = 0.4$s), an impulsive external disturbance will be applied to the middle of its hip and the robot should reject the disturbance and keep its stability while continuing walking in place. To validate the robustness of the framework and to characterize the maximum level of withstanding of the robot, this simulation will be repeated with different amplitudes and a fixed duration of impact (100ms). The amplitude of the



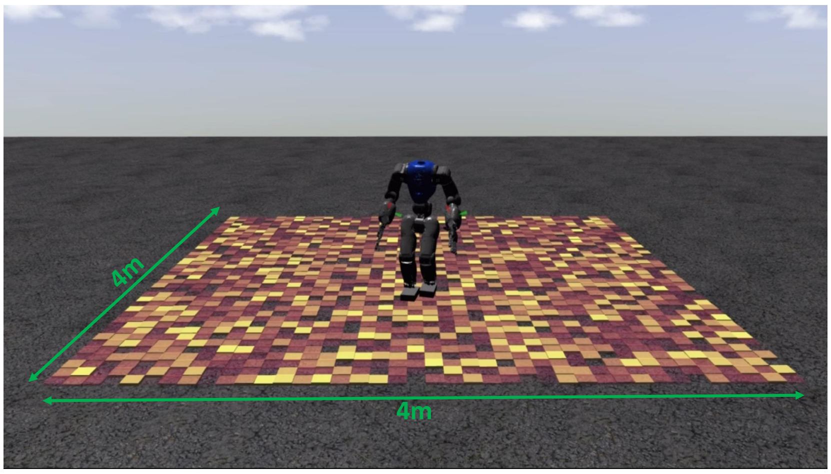
(a)



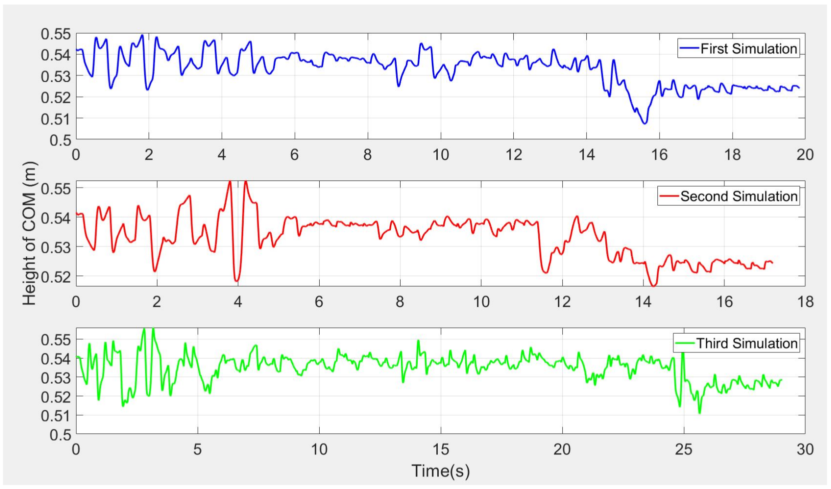
(b)

Figure 6.12: Push recovery scenario: while the robot is walking in place, an impulsive push will be applied. **(a)** A snapshot of push recovery scenario; **(b)** The blue line represents the velocity of COM in X direction during the simulation and the pink rectangles represent the push time and duration.

first impact is 80N and it will be increased every 8s by 20N while the amplitude is less than 120N then being changed to 10N until the robot falls. Figure 6.12 (a) shows a snapshot of the simulation while the robot is subject to a severe push of 130N and capable of rejecting this disturbance and keeping its stability. The simulation results show that the framework is robust against external disturbances and $F = 150\text{N}$ was the maximum level of withstanding of the robot. Figure 6.12 (b) shows the velocity of COM in X direction during this simulation. In this figure, the pink rectangles represent the push duration and as it is shown in this figure, after applying a push, the COM's velocity increases impressively and the controller was able to keep the stability. A video of this simulation is available online at `https://youtu.be/os3Dex07Op0`.



(a)



(b)

Figure 6.13: Walking on an uneven terrain: **(a)** A snapshot of the scenario; **(b)** The height variation of COM while robot is walking on the uneven terrain.

Walking on Uneven Terrains

This scenario is designed to validate the performance of the framework for generating walking on uneven terrain. In this simulation, the robot is placed on an uneven terrain within a square area of side length 4m. The robot does not have any information about the terrain and should walk forward to step out of this area. In this simulation, the uneven terrain is generated by placing a set of tiles with the same size but random height (from 25mm to 50mm) next to each other. To have a clear representation of the generated terrain, the color of each tile is considered as a function of its height (see Figure 6.13 (a)). This simulation has been repeated using three different tile's size (100cm^2, 400cm^2 and 625cm^2). The complexity of the terrain depends to the size and the height of the tiles, the terrain that is generated by the larger tile being more challenging than the others because of more narrow edges and more ups and downs which cause slipping the feet. The height variations of the COM have been recorded during the simulations and are presented in Figure 6.13 (b). The simulation results showed that the framework is capable of providing stable walking on such terrains and handle such uncertainties. A video of this simulation is available online at `https://youtu.be/e9MK6Jy1KHg`.

6.8 Experiments

The simulation results showed that the framework was able to generate robust locomotion even in challenging situations. To explore the effectiveness of the proposed framework regarding the dynamics model and the controller, we developed a baseline framework based on LIPM and our proposed framework and conducted all the presented simulations in the previous section to compare the results. In the *Walking Around A Disk* scenario, we increased the complexity of the scenario by increasing the step's lengths from 0.08m to 0.25m. The results showed that the baseline framework can successfully complete the simulation by a maximum step length of 0.12m while the proposed framework is capable of completing the simulation by 0.22m which is 83% better than the baseline. In the *Omnidirectional Walking* scenario, although the current commands were challenging for the baseline, both frameworks were able to complete the simulation successfully. Therefore, we defined a scale factor to change the input commands and increase the complexity of the scenario consequently. The factor has been increased while the robot can not complete the simulation. The results showed that the scale factor can be increased up to 1.03 for the baseline and 1.16 for the proposed framework. For both frameworks, the last command ($X = 0.2$m, $Y = 0.05$m, $\alpha = 0.26$rad/s) was the most challenging command in the simulation and the robot mostly lost its stability. Therefore, we selected this part of the simulation to compare the performance of both frameworks. The results showed that the proposed framework is 13% better than the baseline. In the next simulation, to compare the maximum level of the withstanding, we performed the *Push Recovery* scenario using the baseline and $F = 100$N was the maximum withstanding level which is 50% less than the proposed framework. In the last simulation, we examine the capability of baseline for walking on uneven terrain. In the two first terrains (small and medium tiles), the robot could pass almost half of the terrain and once it put its feet on a narrow edge, it lost its stability and fall down. In the third terrain, it lost its stability immediately. Therefore, the baseline could not pass successfully the uneven terrains. A summary of the simulations results is given in the Table 6.2.

Table 6.2: Summary of the experiments results.

Simulation Scenario	**Baseline framework**	**Proposed framework**	**Improvement**
Walking Around A Disk	step length: 0.12m	step length: 0.22m	83%
Omnidirectional Walking	scale factor: 1.03	scale factor: 1.16	13%
Push Recovery	withstanding: 100N	withstanding: 150N	50%
Walking on Uneven Terrains	Failed	Success	100%

The simulation results validated the performance of the framework and showed that considering the dynamics of the torso and legs extremely improved the performance in terms of stability and speed. Unlike [19, 99] we have not used the current state of the system to adjust the planner parameters online, which improves the performance in terms of stability and speed. Also, unlike [99], we did not take into account the vertical motion of masses to keep the linearity of our dynamics model but taking into account these motions along with the arm motions can improve the performance.

6.9 Summary

In this chapter, we have developed a model-based walking framework to generate robust biped locomotion. The core of this framework is a dynamics model that abstracts the overall dynamics of a robot into three masses. In particular, this dynamics model and the ZMP concept were used to represent the overall dynamics model of a humanoid robot as a state-space system. Then, this state-space system was used to formulate the walking controller as a linear MPC which generates the control solution using an online optimization subject to a set of objectives and constraints. Later, we have presented a hierarchical planning approach which was composed of three main layers to generate walking reference trajectories. To examine the performance of the proposed planner, a path planning simulation scenario has been designed. Afterwards, according to the planned reference trajectories, a set of numerical simulations has been performed using MATLAB to examine the performance and robustness of the controller. Besides, the proposed framework has been deployed on a simulated COMAN humanoid robot to conduct a set of simulations using Gazebo simulator. The simulation results validated the performance and robustness of the proposed framework. Additionally, the simulation results confirmed that considering the dynamics of the torso and legs improved the performance in terms of stability and speed.

In the next chapter, we will develop a deep reinforcement learning module that combines with the frameworks proposed in the previous chapters to regulate the frameworks' parameters adaptively and to generate residuals to adjust the robot's target joint positions.

Chapter 7

Learning Residual Physics

Although humanoid robots are made to resemble humans, their stability is not yet comparable to ours. When facing external disturbances, humans efficiently and unconsciously combine a set of strategies to regain stability. This chapter deals with the problem of developing robust hybrid locomotion frameworks for biped robots. To this end, the Linear Inverted Pendulum (LIP) and Divergent Component of Motion (DCM) concepts will be used to formulate the biped locomotion and stabilization as an analytical control framework. On top of that, a neural network with symmetric partial data augmentation learns residuals to adjust the joint's positions, thus improving the robot's stability when facing external perturbations. Afterwards, we tackle the problem of developing a robust omnidirectional walking framework, which can generate versatile and agile locomotion on complex terrains. To do so, we develop a closed-loop CPG-ZMP walking engine that is combined with a reinforcement learning module. This module learns to regulate the walk engine's parameters adaptively and generates residuals to adjust the robot's target joint positions (residual physics). Additionally, we propose a proximal symmetry loss to increase the sample efficiency of the Proximal Policy Optimization (PPO) algorithm by leveraging model symmetries. The effectiveness of the proposed framework was demonstrated and evaluated across a set of challenging simulation scenarios. The robot was able to generalize what it learned in one scenario, by displaying human-like locomotion skills in unforeseen circumstances, even in the presence of noise and external pushes.

7.1 Introduction

Humanoid robots are extremely versatile and can be used in a wide range of applications. Nevertheless, robust locomotion is a challenging topic which still needs investigation. *Stability and safety* are essential requirements for a robot to act in a real environment. Humans combine a set of strategies (e.g. moving arms, ankles, hips, taking a step, etc.) to regain stability after facing an external disturbance. The question is: *despite the humanoid robots' versatility, why are they not as capable as us?*

To find an answer to this question, we started by investigating the stabilizer systems of humanoid robots. The core of these systems is typically an abstract or accurate dynamics model of the robot, according to which, a set of controllers is designed to constantly cancel the effects of perturbations, such as external pushes, uneven terrains, and collisions with obstacles. Abstract models are used more often due to their simplicity and computational efficiency but also because they are platform-independent and can be easily adapted to any type of humanoid robot.

The most widely used model in literature is the Linear Inverted Pendulum (LIP) which abstracts

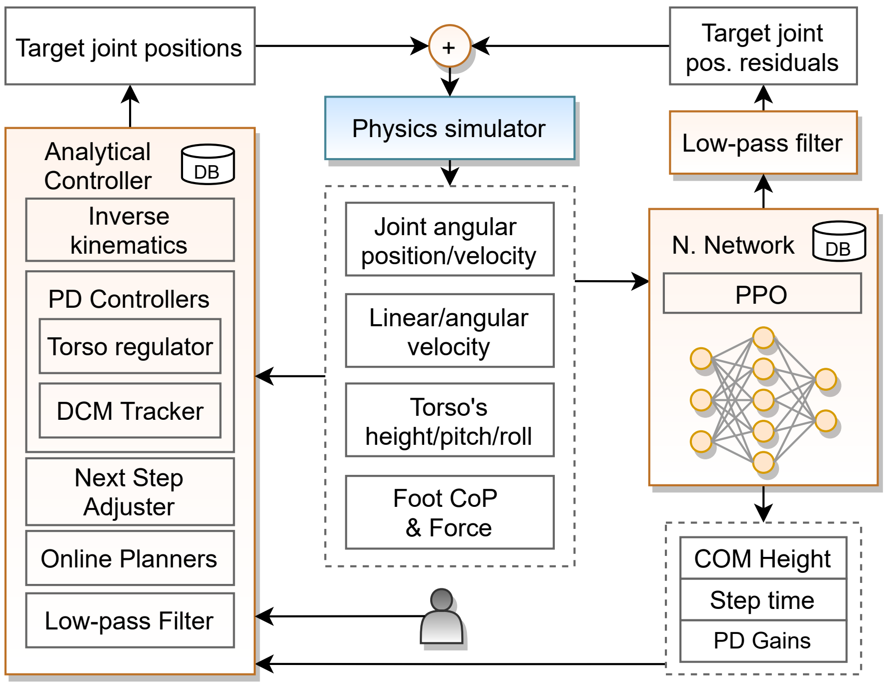


Figure 7.1: An overview of the proposed stabilizer system. The highlighted boxes represent the main modules and the white boxes are the exchanged information among them.

the overall dynamics of a robot as a single mass. It restricts the vertical movement of the mass to provide a linear model which yields a fast solution for real-time implementations.

This model has been investigated and extended for decades to design and analyze the balance controllers. Pratt et al. [37] introduced the Capture Point (CP) concept to improve walking stability. Basically, the CP uses the current state of the COM to determine a 2D point on the floor where the robot should step to regain its stability. Later, Takaneka et al. [49] proposed DCM concept that splits the LIP's dynamics into stable and unstable parts, such that controlling the unstable part is enough for keeping the stability. In [50], DCM has been extended to 3D and, based on that, the Enhanced Centroidal Moment Pivot point (eCMP) and also the Virtual Repellent Point (VRP) has been introduced, which could be used to encode the direction, magnitude and total forces of the external push. Using these concepts, several control approaches including classical feedback controllers [39, 109, 110], LQR-based methods [14, 111, 112, 113] and MPC [98, 99, 114] have been proposed to formulate the stabilizer system. All of them are trying to compensate the tracking error by using a combination of three strategies which are: manipulating the Ground Reaction Force (GRF) and modifying the position and the time of the next step. Recently, researchers are investigating about releasing the assumptions of LIP (e.g. vertical motion of the COM and its angular momentum) which causes dealing with nonlinearities [115, 116, 117]. The stability of humanoid robots has significantly improved but they are not stable and safe enough to be utilized in our daily-life environments.

To answer the question raised in the beginning of this chapter, we looked at how humans learn recovery strategies. We rely on past experiences to improve our methods. After a few years, we have a solid locomotion strategy, on top of which is easy to learn new things. Therefore, learning recovery strategies is just a matter of efficiently combining our locomotion gait with additional residual reactions. In this chapter, we firstly focus on designing a stabilizer system for humanoid robots which is based on a tight coupling between a DCM-based analytical controller and a machine learning module. An overview of this system is depicted in Figure 7.1. Specifically, we use LIPM to analytically formulate biped locomotion and recovery strategies, and combine it with a symmetry-enhanced optimization framework based on the Proximal Policy Optimization (PPO) [80] to learn residual physics — a concept coined by Zeng et al. [118]. The learned policy adjusts a set of parameters of the analytical controller and learns a set of model-free skills to regain stability.

Our approach is applied to the COMAN robot [107], which, like most humanoid models, has reflection symmetry in the sagittal plane. Human-like behaviors are hard to characterize. However, we highlight some aspects which are commonly associated with human gaits, such as motion fluidity, pattern repetition, and symmetry. First aspect is dependent on the mechanical part of the robot since a low-level controller can easily produce fluid movements. The repetition of a pattern can be guided by an analytical controller, working as a foundation to learn more complex skills. Finally, the notion of symmetry can be analyzed through two points of view. On the one hand, humans show a clear bias towards symmetry, often associating positive reactions with symmetry and negative reactions with asymmetry [119]. From this perspective, perfect symmetry follows the concept of classical beauty immortalized by Leonardo da Vinci's Vitruvian Man. On the other hand, real humans are not perfectly symmetrical, and, according to Handžić and Reed, unbalanced gait patterns can be perceived as unimpaired or normal, within reason [120]. Therefore, in the context of human-like behaviors, the symmetry of a model should be leveraged when developing biped locomotion, but not to the point where it becomes a hard constraint.

Leveraging model symmetries using domain knowledge is a widely used technique in machine learning approaches. Data augmentation is a straightforward method to achieve that goal, being popular in supervised learning at least since Baird et al. in 1992 [121], according to Chen et al. [122], and gaining momentum in reinforcement learning [123, 124, 125, 126]. We propose a learning framework where the data is only partially augmented, leveraging the symmetry to improve learning time and human-likeness without restricting asymmetric movements too much, thus widening the range of possible behaviors.

The remainder of this chapter is structured as follows: Section 7.2 describes preliminaries of learning on top of a walking engine. Section 7.3 provides an overview of related work. Section 7.4 focuses on the structure of the proposed analytical controller. Section 7.5 introduces a learning framework designed to learn residual physics on top of the analytical controller. In Section 7.6, a set of simulations will be designed and executed to train and test the proposed framework. In Section 7.7 ten different symmetry ratios will be tested to evaluate residual symmetry, drift and robustness of the framework. Afterwards, the CPG-ZMP walking engine presented in Chapter 2 will be augmented with learning residual physics module in Section 7.8 to develop an agile and versatile walking engine. The architecture of this framework will be presented in Section 7.9. We will address the problem of generating a walking gait in Section 7.10. Afterwards, a proximal symmetry loss function will be proposed in Section 7.11 to extend the PPO algorithm by leveraging model symmetries. Then, a set of simulations will be designed and executed to train and test the proposed framework in Section 7.12. Section 7.13 focuses on sample efficiency evaluation of the different symmetry loss and then the simulation results will be compared in Section 7.14. Finally a brief summary of this chapter will be presented in Section 7.15.

7.2 Preliminaries

The problem of learning on top of a walking engine can be described as a Markov Decision Process (MDP) – a tuple $\langle S, A, \Psi, p, r \rangle$, with a set of states S, a set of actions A, a set of possible state-actions pairs $\Psi \subseteq S \times A$, a transition function $p(s,a,s') : \Psi \times S \rightarrow [0,1]$, and a reward function

$r(s,a) : \Psi \rightarrow \mathbb{R}$.

Model reduction allows the exploitation of redundant or symmetric features. To this end, Ravindran and Barto [127] proposed a mathematical formalism to describe MDP homomorphisms — a transformation that groups equivalent states and actions. An MDP homomorphism h from $M = \langle S, A, \Psi, p, r \rangle$ to $M' = \langle S', A', \Psi', p', r' \rangle$ can be defined as a surjection $h : \Psi \rightarrow \Psi'$, which is itself defined by a tuple of surjections $\langle f, \{g_s | s \in S\} \rangle$. For $(s,a) \in \Psi$, the surjective function $h((s,a)) = (f(s), g_s(a))$, where $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$ satisfies:

$$
\begin{aligned}
& p'(f(s), g_s(a), f(s')) = t(s, a, [s']_{B_h|S}), && (7.1)\\
& \qquad \forall s, s' \in S, a \in A_s, \\
\text{with} \quad & t(s, a, [s']_{B_h|S}) = \sum_{s'' \in [s']_{B_h|S}} p(s, a, s''), \\
\text{and} \quad & r'(f(s), g_s(a)) = r(s,a), \quad \forall s \in S, a \in A_s, && (7.2)
\end{aligned}
$$

where B is a partition of M and, consequently, a partition of $\Psi \subseteq S \times A$ into equivalence classes; $B_h|S$ is a partition resultant from the projection of B onto S; $[s']_{B_h|S}$ denotes the block of partition $B_h|S$ to which state s' belongs; and $t : \Psi \times B|S \rightarrow [0,1]$ is the probability of transiting from state s to a state in $[s']_{B_h|S}$.

MDP symmetries constitute a specialization of the described framework, where f and $g_s, s \in S$ are bijective functions and, consequently, the homomorphism $h = \langle f, \{g_s | s \in S\} \rangle$ from M to M' is an isomorphism. Additionally, since symmetries can be characterized as MDP isomorphisms from and to the same MDP, they are automorphisms, which simplifies the homomorphism conditions (7.1) and (7.2):

$$
\begin{aligned}
& p(f(s), g_s(a), f(s')) = p(s, a, s'), \quad \forall s, s' \in S, a \in A_s, && (7.3)\\
\text{and} \quad & r(f(s), g_s(a)) = r(s,a), \quad \forall s \in S, a \in A_s. && (7.4)
\end{aligned}
$$

7.3 Related Work

Several approaches for developing biped locomotion have been proposed that can be categorized into three major categories. In the remainder of this section, these categories will be introduced and some recent works in each category will be briefly reviewed. This section starts by presenting specific techniques to leverage symmetries in MDPs, followed by a review of some recent works concerning the development of hybrid walking frameworks.

7.3.1 Symmetries in MDPs

Following the overview about symmetries in MDPs presented in Section 7.2, we delve now into specific techniques, as well as their strengths and weaknesses. In the context of temporal symmetry, time inversion requires the application to be characterized as a conservative system with no loss of mechanical energy (e.g. due to friction), such as a frictionless pendulum which Agostini and Celaya [123] control through RL. Since this assumption is unrealistic in real-world systems, this approach is not desirable. Regarding spatial symmetry, the contributions can be majorly subdivided into:

Relabeling states and actions

Surovik et al. [128] combine this technique with reflection transformations to swap frame-dependent values, such as lateral rotation, to effectively reduce the state volume of a tensegrity robot. However, this same idea can be applied to biped models in an intuitive way. If, instead of left and right, we think of "stance" and "non-stance" leg, we can relabel the physical legs every half cycle to obtain a symmetric controller [129, 130]. Peng et al. [131] apply this relabeling method at fixed intervals of 0.5s, forcing the gait cycle to have a fixed period. This idea is simple to implement but it restricts the gait dynamics, hindering fast recoveries from unexpected external forces. In all cases, the policy is constrained by some analytical symmetry switch, whether it is based on time or behavioral patterns.

Data augmentation

In symmetry oriented solutions, data augmentation can be used with model-based [132] or model-free RL algorithms, although the scope of this work is limited to the latter alternative. Examples of successful applications of this technique include a real dual-armed humanoid robot that moves objects [133], the walking gait of several humanoid models [124] and a quadruped with more than one plane of symmetry [134], among others [123, 135]. This approach is computationally inefficient for large sample batches.

Symmetric Network

Concerning humanoid models, Abdolhosseini et al. [124] introduced a symmetric network architecture which forces perfect symmetry. This method guarantees that if states and actions are symmetrically normalized, the behavior has no asymmetrical bias. However, as the authors acknowledge, neutral states are inescapable, i.e., a symmetrical policy $\pi(s) = g_s(\pi(f(s))), \forall s \in S$ applied to a symmetrical state $s = f(s)$ cannot yield asymmetrical states unless the environment introduces its own bias. Pol et al. [136] generalize this approach to additional problems by introducing MDP homomorphic networks, which can be automatically constructed by stacking equivariant layers through a numerical algorithm. The equivariance constraints (under a group of reflections or rotations) are applied to the policy and value networks. This family of solutions lacks the ability to control the symmetry enforcement at runtime and the ability to symmetrize asymmetrical models.

Symmetry Loss function

Mahajan and Tulabandhula [137] proposed an automated symmetry detection process and a method of incorporating that knowledge in the Q-learning algorithm [138]. A soft constraint was presented as an additional loss for the Q function approximator, $\hat{\mathbb{E}}_t[(Q_\theta(f(s_t), g_s(a_t)) - Q_\theta(s,a))^2]$, where θ is the parameter vector of Q, and the expectation $\hat{\mathbb{E}}_t$ indicates the empirical average over a batch of samples. Yu et al. [139] transpose this idea to policy gradients by fusing a new curriculum learning method with an homologous loss function, $\sum_i ||(\pi_\theta(s_i) - g_s(\pi_\theta(f(s_i)))||^2$. The authors present very good results when both approaches are used simultaneously. However, without the learning curriculum, the symmetry loss has no significant impact in the learned policy.

7.3.2 Analytical Approaches

The basic idea behind the approaches in this category is using an abstract or accurate dynamics model of the robot and designing a set of controllers based on some criteria to minimize the tracking error.

Griffin et al. [60] proposed a walking stabilizer system based on adjusting the step time and position. They proposed a simple step time adjustment algorithm to speed up the swing leg for setting the foot down quickly to recover from an error. Additionally, they developed a quadratic program to combine the abilities of a center of pressure (COP) feedback controller and a step adjustment algorithm. This combination provides an optimized reference trajectory based on footstep adjustments which takes into account the COP feedback control. The performance of their approaches has been validated using simulations and real robot experiments.

Jeong et al. [140] developed a walking controller based on online optimization of three push recovery strategies: ankle, hip, and stepping strategies. They used Linear Inverted Pendulum Plus Flywheel (LIPFM) as their dynamics model. This model is an enhanced version of LIPM that considers the momentum around the COM. They combined all the strategies, including the step time adjustment, based on a set of weighting factors, using an optimization program. The performance of their approach was shown by a set of simulations, including abstract and accurate model dynamics. They argued that their approach is practical enough to be deployed on real hardware and validated that with real robot experiments.

Stephane Caron [115] used a variable-height inverted pendulum (VHIP) to design a biped stabilizer based on linear feedback. They argued that VHIP is simple to implement and also more tractable in comparison with the 3D version of DCM because of its linear dynamics. Using this abstract model, they introduced a new push recovery strategy that utilizes the vertical motion of the COM to recover from severe perturbations. They set up an experiment using a real HRP-4 humanoid robot to show the performance of their method. They showed that the behavior of their stabilizer is similar to LIPM once the Zero Momentum Point (ZMP) is inside the support polygon, and the height variations were enabled when the ZMP reached the support polygon's edges.

Khadiv et al. [141] argued that adjusting the time and position of the next step is sufficient to improve the locomotion robustness. They proposed a simple and optimal walking pattern generator that was able to select the next step location and timing at every control cycle. The performance of their approach has been validated using a simulated humanoid robot with passive ankles through different simulation scenarios, including push recovery and foot slippage.

7.3.3 Machine Learning Approaches

The approaches in this category are designed to learn a feasible policy through interaction with the environment. Nowadays, Deep Reinforcement Learning (DRL) has shown its capability by solving complex locomotion and manipulation tasks, which are generally composed of high-dimensional continuous observation and action spaces [73, 142].

Data augmentation in reinforcement learning is widely used to improve optimization performance but, in this work, we restrict the scope to symmetry-oriented solutions. In the context of supervised learning, it is worth mentioning that RL can also be used as a means to generate data augmentation policies [143]. Agostini et al. [123] differentiate between spatial and temporal symmetry, where the former considers spacial transformations such as reflection and rotation, and the latter considers time transformations, such as time inversion. Since our model is not a conservative system, the latter is not applicable, as time inversion symmetry requires no energy loss (e.g. due to friction). One of the proposed solutions to exploit spatial symmetries on an inverted pendulum is to generate symmetric data from actual samples and feed that information to the optimization algorithm. In literature, this process is used to control distinct models performing numerous tasks, which include teaching a real dual-armed humanoid robot how to move objects [125], incorporate symmetry in the walking gait of various humanoid models [124] and a quadruped [134] (which has more than one plane of symmetry), and even reducing the complexity of game of GO through a dihedral group of 8 reflections and rotations [126].

7.3.4 Hybrid Approaches

The approaches in this category are focused on combining the potential of both aforementioned categories. To do so, learning algorithms are used on top of physics-based controllers to predict the controllers' parameters and to learn residual physics, which can lead to impressively accurate behaviors [118, 144].

Yang et al. [144] designed a hierarchical framework based on DRL to ensure the stability of a humanoid robot by learning motor skills. Their framework is composed of two independent layers, the high-level layer generating a set of joint angles and the low-level layer translating those angles to joint torques using a set of PD controllers. In their approach, the reward function was composed of six distinct terms that were mostly related to the traditional push recovery strategies, and it was obtained by adding all terms together with different weights. They showed the performance of their approach using a set of simulations with a simulated NASA Valkyrie robot. The results demonstrated that the robot learned how to regain its stability while facing external disturbances.

Tsounis et al. [145] combined model-based motion planning and DRL to realize terrain-aware locomotion for quadruped robots. They decomposed locomotion into main parts and independently trained them using model-free DRL to plan and execute foothold and base motions. They demonstrated the performance of the approach using a suite of challenging terrains, including uneven terrain, gaps, and stepping-stones. The simulation results showed that the simulated robot overcame all the challenges successfully.

Koryakovskiy et al. [76] argued that learning from scratch approaches are not applicable on real robots due to involve many trials with exploratory actions. To circumvent this problem, they combined RL with a model-based control using two different approaches to compensate model-plant mismatches. The first approach consisted of learning a compensatory control action that minimizes the same performance measure as a nominal controller. The second approach was designed to learn a compensatory signal from the difference between an actual transition and a predicted transition by the internal model (MPML). Using a set of simulations, they showed that MPML is not only twice as fast, but it also exhibits no deviating behaviors. Moreover, they tested MPML using a real robot named Leo in a squat scenario. The results showed that MPML successfully realizes squatting by learning the compensation signal.

Ahn et al. [146] formulated humanoid locomotion as a novel Markov Decision Process (MDP) for safe and data-efficient learning, aided by a dynamic balancing model. Specifically, they proposed a structured footstep planner by combining a time-to-velocity-reversal planner based on LIPM with a neural network and a safety mechanism. In their structure, the planner generates achievable sub-optimal guidance, and then the neural network is used to maximize the long-term reward. The safety controller takes into account the capturability concept to facilitate safe exploration. To evaluate the performance of their method, they carried out a series of simulations using two different simulated robots. The results showed that their method can generate various types of walking, including turning and walking on irregular terrains robustly.

Tran et al. [94] proposed a perturbation rejection based on reinforcement learning on top of a CPG walking generator. Specifically, they combined a self-organizing map with a Q-learning algorithm to automatically detect and select appropriate reactions to regain stability during the swing phase. The effectiveness of their approach has been validated by simulations and real robot experiments using a

NAO robot. The results showed that their approach was only able to reject perturbations when the robot was performing stable repetitive movements before being pushed.

Wang et al. [147] developed a framework for adaptive walking of humanoid robots based on Matsuoka's CPG [148]. Their framework was composed of three main stages including feature processing, neural network (NN) learning, and signal filtering. Particularly, the first stage transforms Matsuoka's CPG outputs into a normalized limit cycle which will be combined with sensory feedback using a three-layer NN to generate a set of rhythmic signals for walking. In the last stage, they used a first-order low-pass filter to avoid discontinuity in the output of the NN. The performance of their method has been validated through a set of simulations and experiments using a NAO robot. The results showed that their framework was able to generate adaptive walking on fixed and inclined terrains based on sensory feedback.

Van der Noot et al. [149] developed a bio-inspired controller for biped locomotion. Their controller utilizes virtual muscles driven by reflexes and a central pattern generator to generate a human-like and energy efficient locomotion. In their structure, the controller generates torque references for all the joints based on biped state. Additionally, the user can provide high-level commands as linear or quadratic functions. They deployed their framework on a simulated COMAN and showed a robust and human-like walking.

It should be noted that the hybrid approach has been applied to many types of robots, from quadrupeds [150] to snakes [151], to generate bio-inspired and robust locomotion for traversing complex environments. According to the aforementioned works, we believe that using machine learning on top of analytical approaches is the key to open doors for humanoid robots to step out of laboratories. In the remainder of this chapter, we will describe the design of a framework composed of an analytical controller and a DRL algorithm, used to learn residual physics.

7.4 Analytical Controller

In this section, the structure of our analytical controller will be presented. The overall architecture of this controller is depicted in Figure 7.2. The core of this controller is based on the LIPM and DCM concepts. The proposed controller is composed of two main modules: `Online Planners` and `PD Controllers`. `Online Planners` is responsible for generating the reference trajectories according to the stride's parameters provided by the user, the robot's state, and the controllers' output. `PD Controllers` is responsible for regulating the upper body orientation and tracking the planned trajectories to generate closed-loop locomotion. The corresponding target joint positions will be generated using the `Inverse Kinematics`, which takes into account kinematic feasibility constraints. The target joint positions will be fed into the `Physics Simulator`, which is responsible for simulating the interaction of the robot with the environment and producing sensory information, as well as the global position and orientation of the robot in the simulated environment. In the rest of this section, the details of `PD Controllers` and `Online Planners` will be explained.

7.4.1 Regulating the Upper Body Orientation

The upper body of a humanoid is generally composed of several joints. While the robot is walking, their motions and vibrations generate angular momentum around the COM. LIP-based models do not take this momentum into account and consider that the Ground Reaction Force (GRF) always passes through the COM. To cancel the effects of this momentum, we designed a PD controller based on the

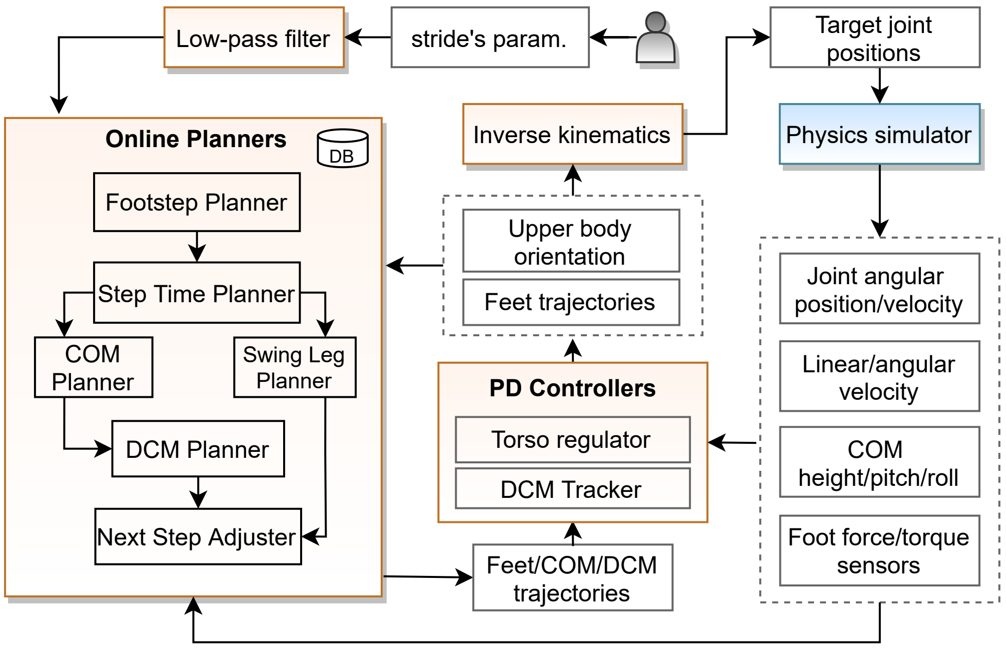


Figure 7.2: Overview of the proposed analytical controller. The online planners module generates a set of reference trajectories according to the input command and the states of the system constantly. The PD controllers module is responsible for tracking the generated trajectories. The highlighted boxes represent the main modules and the white boxes are the exchanged information among them.

inertial sensor values that are mounted on the robot's torso:

$$\dot{\Phi} - \dot{\Phi}_d = -K_\Phi(\Phi - \Phi_d) \qquad , \tag{7.5}$$

where $\Phi = [\Phi_{roll} \quad \Phi_{pitch}]^\top$ is the angle of the torso, $\dot{\Phi}$ represents the angular velocity of the torso, Φ_d denotes the desired state of the torso and K_Φ is the controller gain.

7.4.2 DCM Tracker

According to LIPM model, the dynamics model of a humanoid robot can be represented using the following differential equation:

$$\ddot{c} = \omega^2(c - p) \qquad , \tag{7.6}$$

where $c = [c_x \quad c_y]^\top$ is the position of the COM, $p = [p_x \quad p_y]^\top$ represents the position of the ZMP, $\omega = \sqrt{\frac{g}{c_z}}$ is the natural frequency of the pendulum, being g the gravity constant and c_z the height of the COM. The DCM for this model is defined as:

$$\zeta = c + \frac{\dot{c}}{\omega} \qquad , \tag{7.7}$$

where $\zeta = [\zeta_x \quad \zeta_y]^\top$ is the DCM and $\dot{c}$ is the velocity of the COM. By taking the time derivative of this equation and substituting (7.6) into the result, LIPM dynamics can be represented by a linear state space system as follows:

$$\frac{d}{dt}\begin{bmatrix} c \\ \zeta \end{bmatrix} = \begin{bmatrix} -\omega & \omega \\ 0 & \omega \end{bmatrix}\begin{bmatrix} c \\ \zeta \end{bmatrix} + \begin{bmatrix} 0 \\ -\omega \end{bmatrix} p \quad . \tag{7.8}$$

This system shows that the COM is always converging to the DCM, and controlling the DCM is enough to develop a stable locomotion. Consequently, the DCM tracker can be formulated as follows:

$$\dot{\zeta} - \dot{\zeta}_d = -K_\zeta(\zeta - \zeta_d) \qquad , \tag{7.9}$$

where K_ζ represents the controller gains, ζ_d and $\dot{\zeta}_d$ are the desired DCM and its time derivative, which are generated by the DCM planner (see Figure 7.2). It should be mentioned that the controllers' gains have been tuned by an expert.

7.4.3 Online Planners

As shown in Figure 7.2, the `Online Planners` is composed of a set of sub-planners which are solved separately and connected together hierarchically to reduce the complexity of the planning process. The planning process is started by generating a set of footsteps ($f_i = [f_{i_x} \quad f_{i_y}]^\top \quad i \in \mathbb{N}$) according to the input stride's parameters and the current feet configuration. Then, the step time planner assigns a set of timestamps to the generated footstep according to the stride duration. Afterwards, to have a smooth trajectory during lifting and landing of the swing foot, a cubic spline is used to generate the swing leg trajectory based on the generated footsteps and a predefined swing height. Accordingly, the COM planner generates the COM trajectory by solving (7.6) as a boundary value problem based on the generated footsteps:

$$c = f_i + \frac{(f_i - c_f)\sinh(\omega(t - t_0)) + (c_0 - f_i)\sinh(\omega(t - t_f))}{\sinh(\omega(t_0 - t_f))} \tag{7.10}$$

where t_0, t_f, c_0 and c_f are the times and corresponding positions of the COM at the beginning and end of a step, respectively. c_f is assumed to be between the current support foot and the next support foot ($\frac{f_i + f_{i+1}}{2}$). After generating the COM trajectory, the DCM trajectory can be obtained by substituting (7.10) and its time derivative into (7.7). This trajectory will be fed into `PD Controllers` to generate closed-loop locomotion.

In some situations, such as when the robot is being pushed severely, the DCM tracker cannot track the reference because of the controllers' output saturation. In such conditions, humans adjust the next step time and location, in addition to the COM's height. Equation (7.7) is an ordinary differential equation. Due to the observability of DCM at each control cycle, the position of the next step can be determined by solving this equation as an initial value problem:

$$f_{i+1} = f_i + (\zeta_t - f_i)e^{\omega(T-t)} \qquad , \tag{7.11}$$

where f_i and f_{i+1} are the current and next support foot positions, t and T denote the time and stride duration, respectively. It should be noted that adjusting the next step time as well as the height of the COM are challenging because of the nonlinearities. In the next section, we explain how these parameters can be adjusted through a machine learning approach.

7.5 Learning Residual Physics

Although the presented analytical controller framework is able to keep the stability of the robot and generate stable locomotion, it does not generalize well to unforeseen circumstances. In this section, we introduce a learning framework designed to learn residual physics on top of the analytical controller. The objective is to regulate control parameters such as the COM height and step length, but also learn model-free skills by adjusting some of the robot's joint positions.

7.5.1 Formal structure

The learning framework extends the PPO algorithm with symmetric data augmentation based on static domain knowledge. Like most humanoid models, the COMAN robot has reflection symmetry in the sagittal plane. This knowledge can be leveraged to reduce the learning time and guide the optimization algorithm in creating a human-like behavior.

7.5.2 Data augmentation

In this chapter, the formulated problem is optimized using PPO [80], an actor-critic algorithm that uses a clipping function to constrain the policy update directly inside the objective function, thus preventing it from being too greedy. After performing a grid search, the batch size was set to 8192 samples and the learning rate to 3e−4 (using a linear scheduler). For each episode, an MDP trajectory j is characterized by a sequence of states, actions and rewards such that $j = \{S_0, A_0, R_0, S_1, A_1, R_1, ...\}$. Each trajectory is used to produce a set of samples $k = \{\{S_0, A_0, Ad_0, V_0\}, \{S_1, A_1, Ad_1, V_1\}, ...\}$, where V_i is obtained from the λ-return as defined by Sutton and Barto [152], and serves as value target for the update function; and Ad_i is the generalized advantage estimate [87]. Our proposal is to partially augment data by copying and transforming only a fraction of the acquired samples. Different augmentation ratios are tested in Section 7.7. As an example, consider the addition of symmetrical samples with a ratio of 50%. Following the condition established in (7.4), each batch of samples is artificially built as $\{W_1, W_2, u(W_2), W_3, W_4, u(W_4), ...\}$ where $u(W_i) = \{f(S_i), g_s(A_i), Ad_i, V_i\}$. The observations' normalization is continuously updated by calculating the mean and standard deviation of each observation. However, both of these metrics are shared among the two symmetric groups to ensure that no asymmetrical bias is introduced.

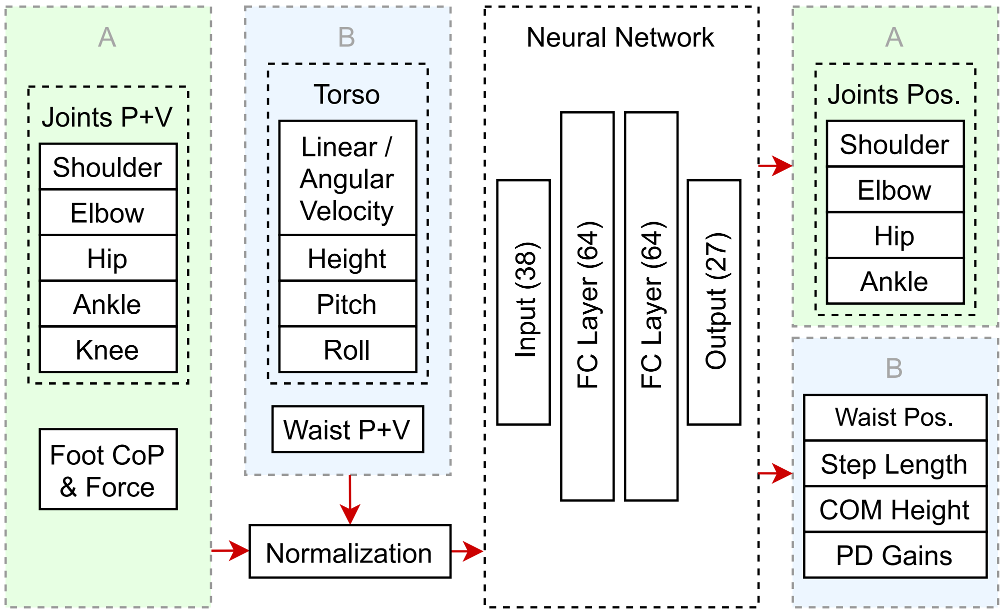


Figure 7.3: Network architecture, system space parameters and symmetry transformation groups used for data augmentation: reflection symmetry transformation (A) and no transformation or inversion (B).

7.5.3 Network Architecture

The network architecture as well as the system space and action space parameters are represented in Figure 7.3. The observations comprise the positions of 6 joints: shoulder, hip and waist with 3 degrees of freedom (DOF), ankle with 2 DOF, knee and elbow with 1 DOF. Considering that all joints are mirrored except the waist, there are 23 joint variables. Additional observations include the foot center of pressure (relative to its center, in x and y) and respective force magnitude, the torso's linear and angular velocity, height, pitch, and roll; totalling 38 state space variables. This information is fed to a neural network composed of two hidden layers of 64 neurons, producing 27 continuous values that control joints' actuators, two high-level parameters (the step length and COM height) and two PD gain vectors (K_Φ from (7.5) and K_ζ from (7.9)). The high-level parameters regulate the analytical controller by providing the duration of each step and the COM height, needed to generate the leg joint trajectories. The joint positions are controlled with residuals, which are added to the precomputed trajectories.

Both the state space and action space parameters are grouped in two categories, according to the symmetry transformations used for data augmentation. Category A includes duplicated observations that are mirrored, considering the reflection symmetry in the sagittal plane. Category B includes unique observations that may remain unchanged or suffer an inversion transformation. As an example, the torso's height and pitch are not transformed when generating symmetric samples, but the roll angle is inverted. The linear and angular velocities are common is some axes, and inverted in others to obtain their symmetric counterparts.

7.5.4 Reward function

The reward function tries to convey one fundamental goal — balance — while motivating cyclic movement patterns. The balance goal seeks to keep the robot on its feet in all situations. The subgoal of performing cyclic movement patterns has the purpose of improving the human-like aspect of the behavior, according to the characterization given in Section 7.1. Specifically, it tries to reduce the neural network's influence (NNI) when there is no need to intervene. Both of these notions can be expressed through the following reward:

$$R = 1 - \overbrace{\frac{1}{J}\sum_{i}^{J}\frac{|\delta_i|}{S_i}}^{NNI}, \tag{7.12}$$

where δ_i is the residual applied to the position of joint i, J is the number of joints, and S_i is the residual saturation value. It is important to note that the NNI component's objective is not to reduce the energy consumption or range of motion, since it is only applied to the residuals and not the hybrid controller's output.

7.6 Simulation Scenarios

To validate the performance of the proposed framework, a set of two learning scenarios and one test scenario has been designed. The goal of this structure is to prepare the physical robot to handle real world adverse conditions. We use the COMAN robot in PyBullet [153] – an environment based on the open source Bullet Physics Engine. The simulated robot is 95cm tall, weighs 31kg, and has 23 joints (6 per leg, 4 per arm and 3 between the hip and the torso).

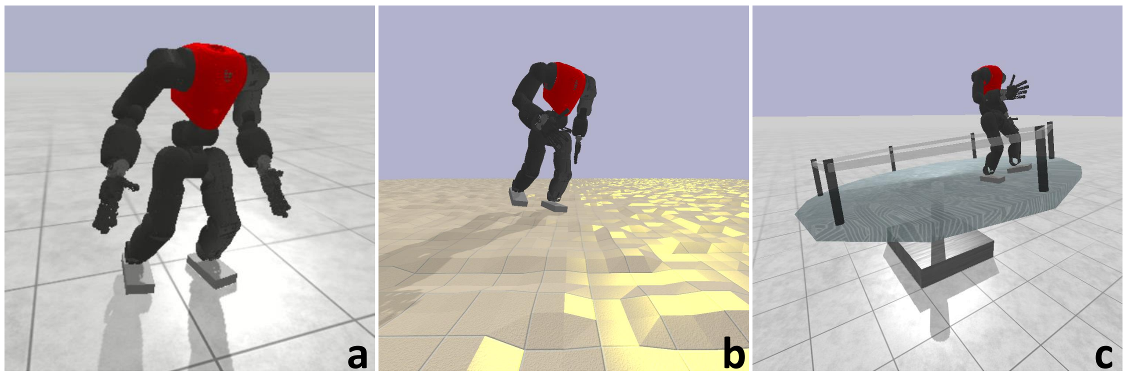


Figure 7.4: Simulation scenarios. The robot learns how to recover from external forces in a flat surface (**a**) and in an uneven terrain with perturbations that can reach 2cm (**b**). It then faces unseen circumstances when it is placed on a tilting platform that moves erratically (**c**).

7.6.1 Scenario L1

The first learning scenario, is composed of a flat platform (see Figure 7.4 (a)), where the robot is initially placed in a neutral pose. It then starts to walk in place, while being pushed by an external force at random intervals, between 2.5 and 3.0 seconds. The force is applied for 25ms and ranges from 500N to 850N. Its point of application is fixed at the torso's center and its direction is determined randomly in a 2D plane parallel to the ground. The robot's objective is to avoid falling. The episode ends when the robot's height drops below 0.35m or any of its body parts (except the feet) touches the ground.

7.6.2 Scenario L2

The second learning scenario, is an extension of the first one, where the flat surface is replaced by an uneven terrain with perturbations that can reach 0.02m, as depicted in Figure 7.4 (b). The external force dynamics are the same as in scenario `L1`.

7.6.3 Scenario T1

The test scenario, was designed to evaluate the generalization capabilities of the hybrid controller in unexpected circumstances. It is characterized by a tilting cylindrical platform (see Figure 7.4 (c)), which is supported by two actuators that rotate on the x and y axes, and range between -15deg and 15deg. The setpoints of each actuator is given by adding a random component $r \in [-8, 8]$deg to a correcting component $c = 0.35 \times P$, where P is the position of the robot in the opposite axis to the actuator. The goal of the latter component is to keep the robot on top of the platform by encouraging it to move to the center. The episode starts in a neutral state with the robot walking in place, and it ends when the robot falls, as in the previous scenarios.

7.7 Experiments

Five different symmetry ratios were tested per learning scenario, totalling ten different configurations. The symmetry ratios were 0 (no data augmentation), 1/8 (1 symmetrical sample is generated per 8 acquired samples), 1/4, 1/2 and 1/1 (full symmetry). For each configuration, five models were trained. Figure 7.5 depicts the learning curves for the best model in each configuration. The results

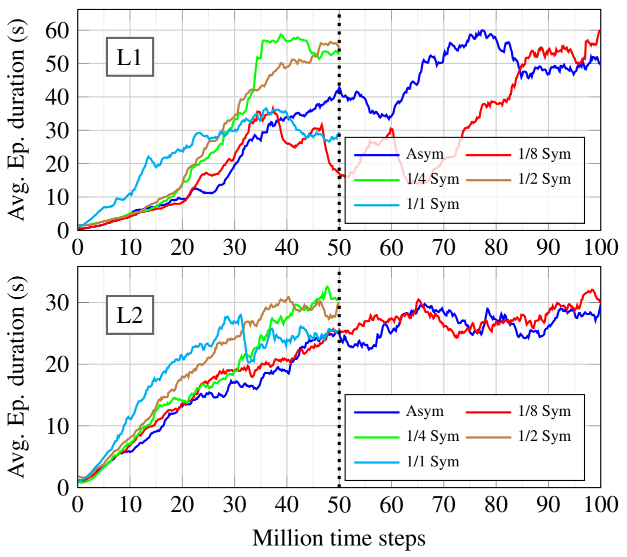


Figure 7.5: Learning curves for the best models trained in scenario `L1` (above) and `L2` (below), under different symmetry configurations.

Table 7.1: Average results per learning configuration

Learning configuration	Episode duration (s)			N. Network Influence	M. Sym. Index
	L1	L2	T1		
Baseline	3.47	1.51	1.87	-	-
L1 Asym	104.5	5.1	4.8	0.072	1.42
L1 1/2 Sym	202.2	4.6	4.8	0.055	1.19
L2 Asym	321.9	34.2	27.8	0.165	1.23
L2 1/2 Sym	193.7	43.5	21.0	0.127	0.99

are grouped according to the training scenario (`L1` above and `L2` below). Most optimizations ran for 50M time steps. However, the asymmetric and 1/8 symmetry configurations needed 100M time steps to reach a plateau. For the configurations that included data augmentation, the best performing ratios were 1/4 and 1/2, with very similar results. In a subjective visual evaluation, the 1/2 ratio model seems to be marginally better in producing a human-like behavior. For the remainder of this section, we will compare in greater detail the asymmetric version with the 1/2 symmetric version. Videos of the results are available at `https://youtu.be/Lk9Wjha7RHw`.

It is important to note that the average episode duration reported by these learning curves results from a stochastic policy with a non-negligible random component. To better assess the optimized models, they were tested in each scenario (including `T1` — the only test scenario) for 1000 episodes using the deterministic policy. Moreover, considering the 50M time steps target, and to be fair with every approach, only the evolution until 50M time steps was considered in these tests. Table 7.1 compares the average performance of four models against the baseline. The first four columns indicate, in this order, the episode duration, in seconds, in scenario `L1`, `L2` and `T1`; and the neural network influence. The last column will be analyzed later in this section.

The baseline version (without residuals) is not able to handle the strong external forces applied in scenario `L1`, falling on average after 3.47s, which is typically after the first push is applied. On `L2`, it falls almost immediately due to the floor perturbations, an outcome which is also seen in `T1`. All four learned models are a great improvement over the baseline. As expected, the last two models that learned on `L2` were able to generalize successfully when tested on `L1` or `T1`, and, on the opposite side, the models that learned on `L1` did not perform well in unforeseen circumstances.

However, some results were not expected. During training, the symmetrically-enhanced models performed better than their counterparts, but while testing in distinct scenarios, the asymmetrical models generalized better. Another interesting result is that the asymmetrical `L1` model performed worse in its own scenario (104.5*s*) than the asymmetrical `L2` model (321.9*s*).

To better understand this outcome, we need to analyze the neural network influence column, whose metric is explained in (7.12). Since `L2` and `L2 Sym` are the most challenging scenarios, the robot learned to sacrifice its immediate reward by applying larger residuals in order to survive for a longer period. Naturally, this is a trade-off between cyclic movement patterns and raw performance. Moreover, learning an asymmetrical behavior can arguably be considered more challenging, which, in this case, has led to a higher network influence. So, in essence, that may explain why it generalizes better than its symmetrical counterpart.

7.7.1 Symmetry analysis

The analytical controller produces symmetrical trajectories upon which the neural network residuals are applied. To evaluate the residuals symmetry, we built upon the concept of Symmetry Index (SI) proposed by Robinson et al. [154]. The original method compares the kinematic properties of each lower limb. To address the issues caused by abstracting the kinematic properties of each joint, we propose the Mirrored Symmetry Index (MSI):

$$\text{MSI} = \frac{\|\delta_t - \delta'_t\|_1}{0.5 \times (\|\delta_t\|_1 + \|\delta'_t\|_1)}, \tag{7.13}$$

where $\delta_t = [\delta^t_1, ..., \delta^t_n]$ is the vector of residuals applied to each joint during time step t, $\|\cdot\|_1$ is the ℓ1-norm, and δ'_t is the vector of residuals applied to the symmetric set of joints if the current state was also symmetrically transformed, i.e., $\delta'_t \sim \pi(\cdot|f(S_t))$, where π is a stochastic policy. Instead of evaluating an average kinematic feature, the MSI computes a symmetry index at each instant, which can then be averaged for a full trajectory to obtain a global symmetry assessment.

As seen in Table 7.1, the models which were learned using the data augmentation method obtained a lower MSI value, when compared to the other two models. The results do not show a large reduction, which can be explained by the analytical controller's role in regulating the trajectory symmetry, and the relaxed data augmentation restriction imposed on the network.

To assess the notion of symmetry in a practical scenario, the models trained on `L2` and `L2 Sym` were submitted to a test where an external force with a constantly increasing norm is radially applied to the robot in a given direction. When the robot is no longer able to recover consistently (more than 50% of the trials), the maximum force is registered and another direction is tested. The result can be seen in Figure 7.6 on the flat terrain (solid red line) and uneven terrain (dotted blue line). In both cases, the robot is able to better withstand forces that are applied to the front (around 0 deg). On one side, the symmetrically-enhanced version presents a more balanced result, which can be visually perceived. On the other side, the asymmetrical model can withstand larger forces in a specific range around 300 deg. This difference consists of a trade-off between symmetry and raw performance.

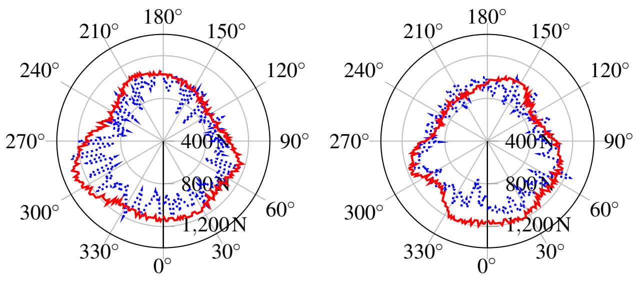


Figure 7.6: Maximum radially applied external force from which the robot can consistently recover as a function of the direction of application, where zero degrees corresponds to the front of the robot. On the left is shown the model which learned on `L2 Sym` and on the right `L2`. The force was applied both in the flat terrain (solid red line) and the uneven terrain (dotted blue line).

7.7.2 Drift

Another way of objectively assessing the influence of symmetry in the learned behaviors is to compare the resulting models in terms of position drift while walking in place. This experiment was performed for 500 seconds, in scenario `L1`, without any external force being applied to the robot. Initially, the robot is placed at $(0,0)$ and it is instructed to walk in place. Figure 7.7 shows the path described by the baseline (green), solely composed by the analytical controller; the `L1 1/2 Sym` model (red); and the `L1 Asym` model (blue). A video demonstration is available at `https://youtu.be/Lk9Wjha7RHw`. All models perform some form of drift over time. However, due to its asymmetry bias, the `L1 Asym` model had the worst results. Sampling the position at every 5 seconds, the total traveled distance was 18.09m for the `L1 Asym` model, 2.60m for the `L1 1/2 model`, and 3.51m for the baseline. Although a high-level controller could compensate for the model bias and eliminate the drift, lowering the need for regulation is always desirable.

7.7.3 Noise robustness

Finally, we present a noise robustness analysis, which is a matter of significant concern on real applications. To test this, the state variables are multiplied by a random factor that follows a uniform distribution $z \sim \mathscr{U}(1.0, N)$ where N ranges from 1.0 to 1.4, i.e., 0% to 40% of maximum noise. Figure 7.8 shows the average impact of this artificial perturbation on the average episode duration, on the uneven terrain scenario, while being pushed by an external force with the same dynamics as described

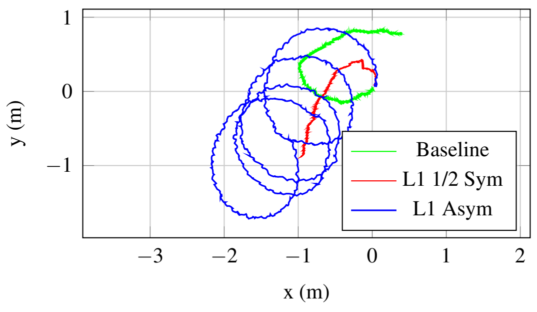


Figure 7.7: Drift comparison, while walking in place for 500s, for the baseline (red), the L1 Asym model (orange) and the L1 1/2 Sym model (yellow).

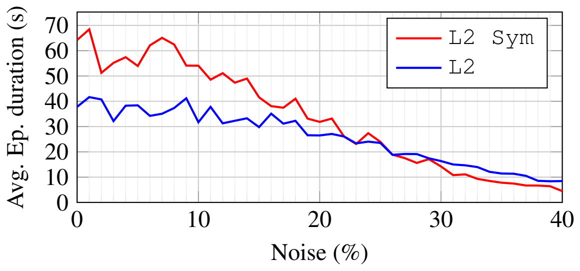


Figure 7.8: Average episode duration as a function of noise applied to the state observations for the symmetrical (red line) and asymmetrical (blue line) models learned and tested on the uneven terrain.

in Section 7.6.1, with a fixed interval of 3.5 seconds. Both the symmetrical and asymmetrical models can withstand a maximum noise of 20% without dropping below the 30s mark, which attests to the models' robustness in considerably noisy scenarios.

In the rest of this chapter, we propose a hybrid omnidirectional walking framework that is able to generate versatile and agile locomotion on complex terrains, even in the presence of noise and external pushes. Specifically, we will use the closed-loop CPG-ZMP walk engine presented in Chapter 2 and combine it with an extension of the PPO algorithm [80] that leverages model symmetries to learn model-free skills and tune the parameters of the walking engine adaptively. We apply this approach to the COMAN [155] humanoid robot in order to learn residual physics.

7.8 A CPG-Based Agile and Versatile Locomotion Using Proximal Symmetry Loss

One of the most recognizable features of humans and bilateral animals is their approximate symmetry with respect to the sagittal plane. Humanoid robots, as the name implies, share some fundamental biomechanical characteristics with humans, including symmetry, which is usually assumed to be exact during simulation. In general, model minimization is conceptually straightforward when dealing with symmetries. However, in practice, there are several techniques to leverage this redundancy. Most of them fall under spatial symmetry and, to a lesser extent, temporal symmetry. The latter concerns invariability under temporal transformation such as scaling or inversion. The former is itself divided into diverse areas.

First, relabeling states and actions allows the permutation of roles between elements of the same type. This is a low-level approach that can be implemented directly on the simulator and requires no changes to the optimization algorithm. Second, data augmentation is widely used in RL to improve sample efficiency and stability, being one of its main precursors to the experience replay method introduced by Lin [156]. In the scope of this work, data augmentation consists of artificially creating samples by applying a symmetric transformation to actually experienced samples. Symmetric networks comprise the third subdivision of spatial symmetry. These solutions enforce symmetry constraints directly on the policy, by modifying the network architecture. This approach is very robust in applications where symmetry is an intrinsic characteristic of optimal solutions. For example, given an optimal ball throwing technique with the right hand and assuming perfect model symmetry, we can automatically infer the optimal technique for the left hand by applying a reflection transformation. However, this equivalence cannot be assumed for a cyclic task, like biped walking, where the optimality of each step cannot be assessed individually. It is therefore unclear whether a perfectly symmetrical gait is preferable or not. Additionally, unbalanced gait patterns (within reason) can be

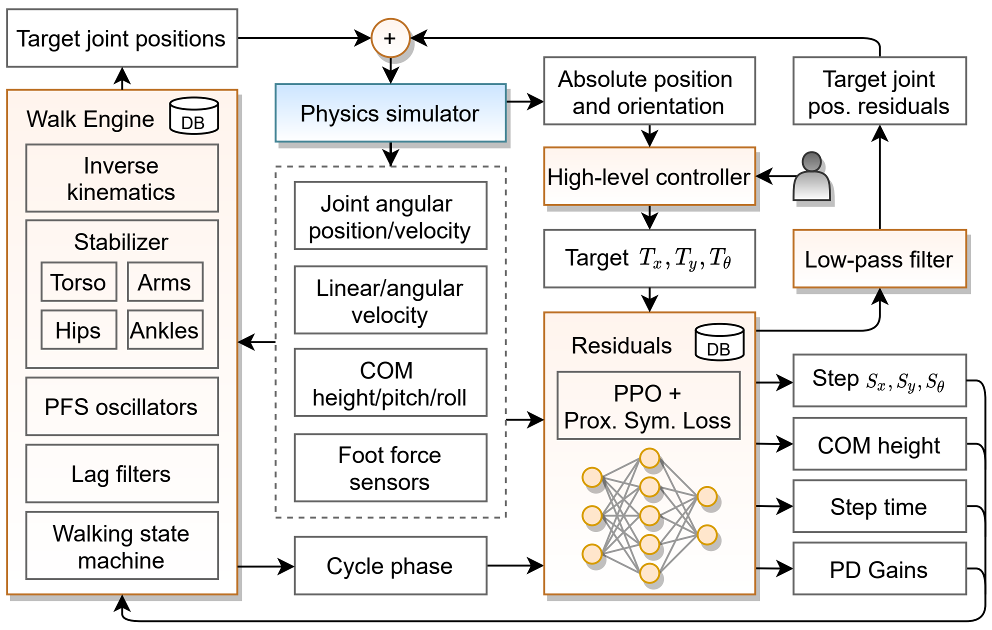


Figure 7.9: Overall architecture of the proposed CPG-based framework. The highlighted boxes represent the main modules and the white boxes are exchange information among them.

perceived as unimpaired or normal [120]. The fourth and final subdivision encompasses the symmetry loss function, which constitutes a flexible way of incorporating symmetry in an optimization problem. It allows more freedom at both design time and runtime than any of the other methods. However, it gives no guarantee of symmetry, like the relabeling technique or symmetric networks.

Concerning problems that do not necessarily benefit from perfect symmetry, symmetry loss is the most advantageous approach because it allows the user to define exactly how important symmetry is in the policy, at any given moment during learning. Moreover, it does not get stuck in neutral states (see Section 7.3.1) and it is more computationally efficient than data augmentation, especially when dealing with large sample batches. However, powerful policy optimization algorithms such as PPO [80] cannot take full advantage of existing solutions. To close this gap, we propose a new loss called Proximal Symmetry Loss.

In the remainder of this chapter, we present two contributions. First, we will use our CPG-ZMP walking engine presented in Chapter 2 to augment it with a reinforcement learning module that learns residual physics. Second, we will present the improvement of sample efficiency and natural-looking behaviors, by extending PPO with a novel symmetry loss.

7.9 Architecture

An overview of the proposed CPG-Based architecture is depicted in Figure 7.9. The highlighted boxes correspond to functional modules, while the white boxes represent data that is exchanged between them. This framework is composed of three main components: the `physics simulator`, the `walk engine` and the `residuals`. The employed simulator was PyBullet [153], which is based on the open-source Bullet Physics Engine. PyBullet is responsible for simulating the interaction of the COMAN robot with the environment and produce information about the robot's joints and sensors for the `walk engine` and `residuals`, as well as its position and orientation for the high-level controller. Based on this information, the high-level controller generates a new target position and orientation, according to the restrictions of the user and the current scenario, and outputs T_x, T_y and T_θ in the robot's local reference frame. Then, the `residuals` module uses those values along with the robot's observations and the cycle phase (which is initialized to zero), and generates parameters for the `walk engine`, including control commands for the step size S_x, S_y and step rotation S_θ, the target center of mass (COM) height, the duration of a step, and PD gains. At the same time, it also outputs target joint position residuals, which go through a low-pass filter.

Finally, the `walk engine` generates a set of target joint positions based on CPG and analytical control approaches according to the parameters dictated by the `residuals` module and the robot's observations. The target positions are added to the position residuals and fed to the physics simulator, which advances one time step. During the reinforcement learning stage, the `residuals` module is optimized using PPO and a proximal symmetry loss. In the testing phase, a static snapshot of the best policy is employed instead.

7.10 Walking Engine

In this section, the problem of generating a walking gait will be addressed by presenting a modular walking engine based on the combination of LIPM with CPG. In this walk engine, a state machine is designed to control the overall gait and to generate walking reference trajectories. These are obtained by arranging a set of Partial Fourier Series (PFS) oscillators and combining them with a set of controllers to stabilize the robot.

7.10.1 Walking State Machine

Due to the symmetric and periodic nature of humanoid locomotion, it can be represented by a state machine which is composed of five distinct states: *Idle*, *Init Single Support*, *Single Support*, *Init Double Support*, and *Double Support*. In this state machine, a timer is assigned to each state and, normally, a transition will occur once the timer is over. Moreover, in some situations, such as when the robot should immediately place its foot down, the state transition can be issued by specific controllers. In the *Idle* state, the robot is standing and waiting for a new walking command ($S = [s_x, s_y, s_z, s_\theta, s_t]^\top$) which contains length, width, height, rotation and duration of the next stride, respectively. The system state transits to *Init Single Support* once the new command is received. It should be noted that the new command is always passed through a first-order lag filter to ensure a smooth update. In this state, all the reference trajectories will be generated based on the filtered command. Then, the robot shifts its COM towards the first support foot to be ready for lifting its swing foot. Afterward, the system state will automatically transit to *Single Support* and the robot tracks the generated references to move its swing leg towards the next support foot. Then, in *Init Double Support*, the robot adjusts its feet to be completely connected to the floor. Finally, during *Double Support*, the robot shifts its COM towards the next support foot and, at the end of this state, the state will transition to *Init Single Support* in case another step is needed, otherwise, it will transit to *Idle*. It should be noted that the new command is applied just at the beginning of *Init Single Support*. To generate walking reference trajectories, a set of footsteps ($f_i = [f_{i_x} \quad f_{i_y}]^\top \quad i \in \mathbb{N}$) according to the input stride's parameters (S) and the current feet configuration will be generated. Then, the LIPM concept is used to model the overall dynamics of a humanoid robot using a differential equation:

$$\ddot{c} = \omega_0^2 (c - p), \tag{7.14}$$

where $c = [c_x \quad c_y]^\top$ denotes the position of the COM, $p = [p_x \quad p_y]^\top$ is the position of the ZMP and $\omega_0 = \sqrt{\frac{g}{c_z}}$ represents the natural frequency of the pendulum, where g and c_z are the gravity constant and the vertical position of the COM, respectively.

According to the LIPM, motion equations in sagittal and frontal planes are equal and independent. Additionally, since the walking gait is symmetric, the reference trajectories can be generated just for one side and then applied to both sides with a half-period phase shift. Accordingly, a set of PFS oscillators are arranged to generate walking reference trajectories based on LIPM in the sagittal plane. A PFS oscillator can be modeled as follows:

$$f(t,N,A,\phi,\beta) = \sum_{n=0}^{N} A_n \sin(n\beta t + \phi_n), \forall t \in \mathbb{R}, \tag{7.15}$$

where t denotes the time, N is the number of frequencies, β is the angular velocity, $A = [A_0, A_1, ...A_n]$ and $\phi = [\phi_0, \phi_1, ..., \phi_n]$ represent the amplitude and the phase of the n^{th} term, respectively. In our target framework, four PFS oscillators are used to generate feet trajectories (X,Y,Z,θ) in Cartesian space, and one is used to generate the arm's trajectory (A) in the x-axis direction. We considered that $c_x = f(t,N,B,\phi,\beta)$ and by plugging it into (7.14), the ZMP trajectory will be obtained:

$$p_x = \sum_{n=0}^{N} B_n (1 + (\frac{n\beta}{\omega_0})^2) \sin(n\beta t + \phi_n). \tag{7.16}$$

This equation is the core of our walk engine and shows that the COM and ZMP have the same frequencies and phase shift, but different amplitudes. This information gives us a clue to configure the oscillators and tune their parameters. The first parameter that should be configured is the number of frequencies, which generally are selected based on the oscillator's objective. For instance, the Z trajectory can be generated just by one term, but another term can be added to absorb the shock during the landing phase of swing foot. After determining the number of frequencies for each oscillator, (7.16) along with the planned foot steps are used to determine the remain parameters by solving nonlinear curve-fitting problems in least-squares sense. Figure 7.10 shows the outputs of the oscillators for an exemplary waking scenario ($v_x = 0.2$m/s, $v_y = 0.05$m/s and $v_\theta = 0.0$rad/s).

7.10.2 Walking Stabilizer

The presented walk engine generates a feed-forward walking regardless of the current state of the robot and the environment and it is not robust enough to cancel the effect of uncertainties (e.g. external disturbances). To deal with the uncertainties, we used the states of torso to design a set of PD controllers for stabilizing the robot. The state of torso is obtained using an inertial sensor values that are mounted on the robot's torso. These controllers stabilize the robot by adjusting the torso, the ankles, the arms, and the hip joints and can be formulated as follows:

$$\dot{\Phi} - \dot{\Phi}_d = -K_\Phi(\Phi - \Phi_d), \tag{7.17}$$

where $\Phi = [\Phi_{roll} \quad \Phi_{pitch}]^\top$ and $\dot{\Phi}$ are the orientation and the angular velocity of the torso, Φ_d represents the desired state and K_Φ is the controller gain that should be tuned by an expert.

7.11 Proximal Symmetry Loss

To understand the effect of the symmetric loss functions, we must analyze the most common implementation of PPO, where the policy and value networks do not share parameters. The PPO's objective L_t^{PPO} can then be expressed as:

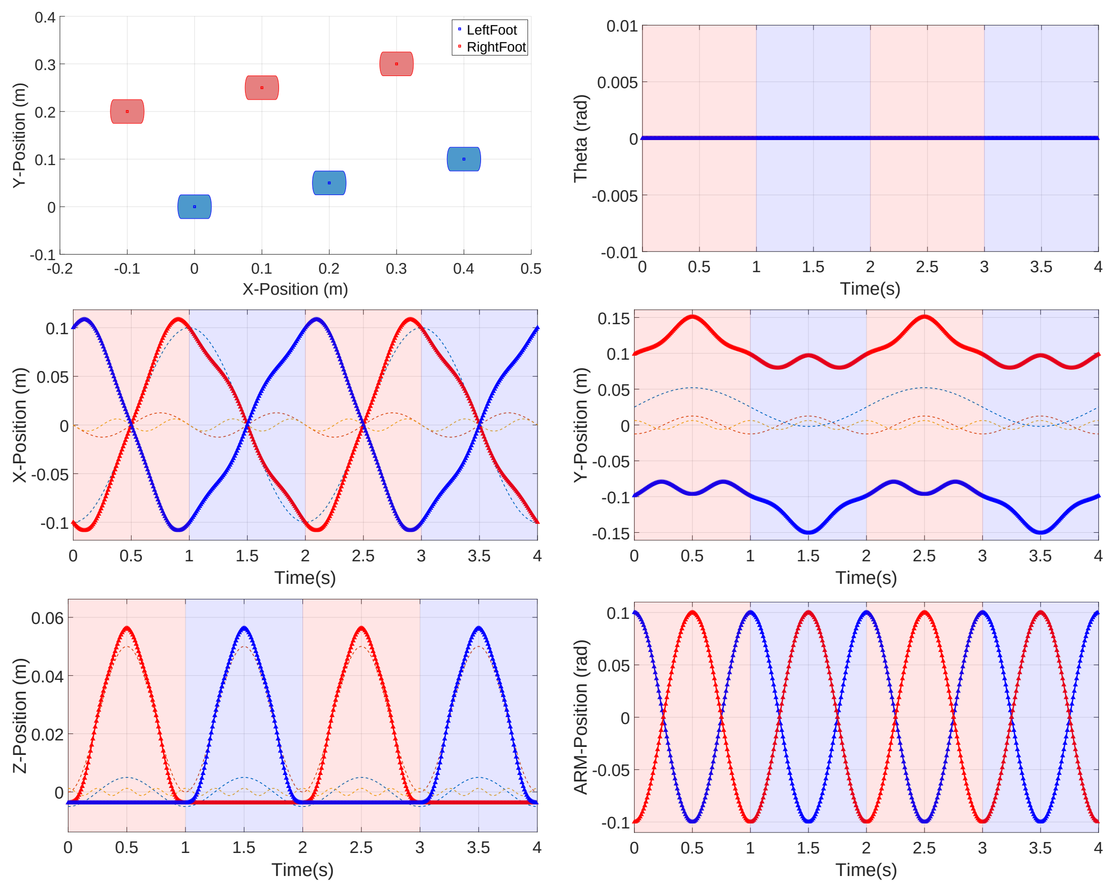


Figure 7.10: Exemplary trajectories for a four-steps walking $v_x = 0.2$m/s, $v_y = 0.05$m/s and $v_\theta = 0.0$rad/s. Dashed lines are the terms of each oscillator and thick red and blue lines represent the output of the oscillators, the red and blue phases representing right or left is moving, respectively.

$$L_t^{PPO}(\theta, \omega) = \hat{\mathbb{E}}_t \left[L_t^C(\theta) - L_t^{VF}(\omega) + cH(\theta) \right], \tag{7.18}$$

$$\text{with} \quad L_t^C(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t \,,\, \operatorname{sgn}(\hat{A}_t) + \varepsilon) \right], \tag{7.19}$$

$$\text{and} \quad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}, \tag{7.20}$$

where the stochastic policy π_θ is parameterized by θ and the value function by ω. $\pi_{\theta_{old}}$ is a copy of the policy before each update, $\hat{A}_t$ is the estimator of the advantage function, L_t^{VF} is a squared error loss to update the value function, ε is a clipping parameter, c is a coefficient and H is the policy's distribution entropy. Since the gradient $\nabla L_t^{PPO}(\theta, \omega)$ with respect to θ ignores $L_t^{VF}(\omega)$ and c can usually be zeroed without negative impact, it is enough to analyze the merge of (7.19) with an arbitrary symmetry loss L_t^S:

$$L_t^{PPO+S}(\theta) = \mathbb{E}_t \left[L_t^C(\theta) \right] - L_t^S(\theta). \tag{7.21}$$

Within L_t^C, the value of the ratio r_t before the update is 1. During the update, considering all the mini-batches and epochs before switching again to the sample acquisition stage, the ratio tends

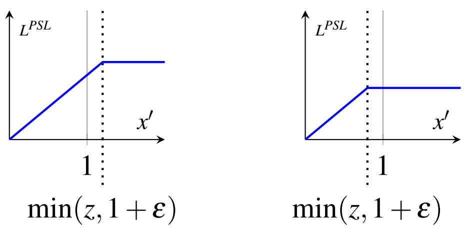


Figure 7.11: Plots for the proximal symmetry loss as a function of ratio x' for one term in one time step. The left plot indicates the behavior when $\min(z, 1+\varepsilon) = 1+\varepsilon$ or $z > 1$, and the right plot when $\min(z, 1+\varepsilon) = z$ and $z < 1$.

to remain near 1 or at least in the same order of magnitude. The advantage estimate $\hat{A}_t$, under the most popular implementations of PPO [157, 158], has a mean of 0, because each advantage estimates batch is z-score normalized. However, when computing the gradient, since we are considering only a mini-batch, the mean is not exactly zero. In practice, for the problem formalized in this work, L_t^C ranges from -1.8×10^{-2} to -1.0×10^{-2}. It is important to retain that this value has a low order of magnitude and is relatively stable. The mirror symmetry loss proposed by Yu et al. [139] $w \cdot \sum_i \|(\pi_\theta(s_i) - g_s(\pi_\theta(f(s_i)))\|^2$, where w is a weight factor, computes the square error between the action taken by the policy before and after the symmetric transformation. On the one hand, this means that, after crossing a certain asymmetry threshold, the loss dominates the policy gradient. On the other hand, if that threshold is crossed to the symmetric side, the symmetry loss loses influence in the policy update. Therefore, w does not dictate the weight of the symmetry loss in a consistent way, when computing the gradient, but rather the position of the symmetry threshold. To harmonize this effect, we propose the proximal symmetry loss:

$$L_t^{PSL}(\theta) = w \cdot \mathbb{E}_t\left[\,\min(x_t(\theta), 1+\varepsilon)\,\right] \tag{7.22}$$

$$\text{with} \quad x_t(\theta) = \frac{\min(\pi_\theta(a'_t \mid f(s_t)), \pi_{\theta_{old}}(a_t \mid s_t))}{\pi_{\theta_{old}}(a'_t \mid f(s_t))}, \tag{7.23}$$

$$\text{and} \qquad a'_t = g_s(\pi_{\theta_{old}}(s_t \mid \sigma = 0)), \tag{7.24}$$

where w is a weight factor and ε is a clipping parameter shared with L_t^{PPO}. $\pi_\theta(\cdot \mid s)$ and $\pi_{\theta_{old}}(\cdot \mid s)$ represent probability distributions during, and before the update, respectively. $\pi_{\theta_{old}}(s \mid \sigma = 0)$ denotes the deterministic action chosen by the policy for state s, before the update, if the standard deviation σ of the stochastic policy's distribution was zero. In other words, the resulting action is given by the mean of the policy's distribution. Please note that for the gradient computation, a'_t is a constant vector of actions. The only component that is not constant during the update is $\pi_\theta(a'_t \mid f(s_t))$.

Considering the ratio $x'_t = \frac{\pi_\theta(a'_t|f(s_t))}{\pi_{\theta_{old}}(a'_t|f(s_t))}$ and $z_t = \frac{\pi_\theta(a'_t|f(s_t))}{\pi_{\theta_{old}}(a'_t|f(s_t))}$, the proximal symmetry loss behaves as depicted in Figure 7.11 for one term in one time step. When $\min(z, 1+\varepsilon) = 1+\varepsilon$ or $z > 1$ (see left plot), the loss restricts the symmetry update analogously to L_t^C. In this way, the influence of the proposed loss is harmonized with PPO, and w carries an intuitive meaning. However, when $\min(z, 1+\varepsilon) = z$ and $z < 1$ (see right plot), the gradient is zero in the first update epoch. This means that the symmetry ratio is currently above what is desired, although the loss will not motivate its reduction. Within $x_t(\theta)$, the term $\pi_{\theta_{old}}(a_t \mid s_t)$ plays a similar role to the implicit symmetry threshold in the loss proposed by Yu et al. [139].

The difference is that the threshold is now dynamic and depends on the standard deviation σ of the stochastic policy π_θ. Without this restriction, the symmetry loss will motivate the reduction of σ to the point where exploration is halted, and numerical instability arises in the probability computations.

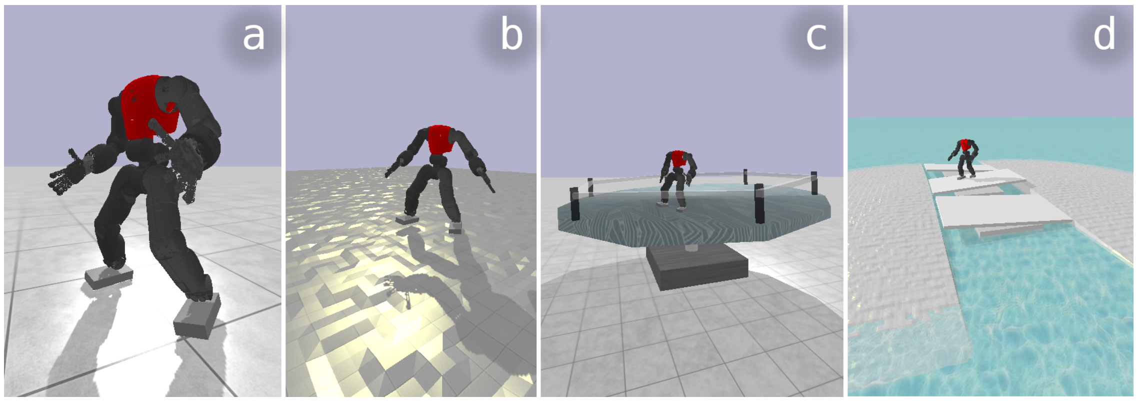


Figure 7.12: Simulation scenarios designed in PyBullet [153]: flat surface (**a**), uneven terrain (**b**), autonomous tilting platform (**c**), and island scenario (**d**) which combines the challenges of the previous ones. The robot was trained on scenario (**b**), and the others were used to test the framework in unforeseen circumstances.

This approach guarantees that the loss will not work against PPO, i.e., it will converge at a stable rate, dictated by w whether the policy has an initially strong asymmetrical bias or not. After the bias is reduced to a certain point, the loss will converge at the same pace as L_t^C, allowing exploration, and promoting algorithmic stability.

7.12 Evaluation Scenarios

Four scenarios were designed to test the walking capabilities of the COMAN humanoid robot in harsh conditions (see Figure 7.12). They are ordered by increasing difficulty: flat surface (a); uneven terrain with 2cm perturbations (b); tilting platform that rotates autonomously, following a random pattern with a 4.5-degree maximum tilt (c); two islands scenario that combines uneven terrain (1cm perturbations), a constant 3-degree slope, and unstable seesaws between both islands with a 4-degree maximum tilt (d). It should be noted that, in all the simulation scenarios presented in Section 7.6 were focused on assessing the performance while the robot is walking in place. In this section, we will be focusing on assessing the performance while the robot is walking.

The agent learned residual physics in scenario (b), without prior knowledge about the terrain. The other three scenarios were designed to assess the ability of the framework to generalize to new conditions and compare its raw performance against a baseline. The robot is initially placed in the center of the map, except in (d), where it is placed on the left island. Then, the high-level controller (see Figure 7.9) generates a new objective, composed of a 2D position (x, y) and a desired final orientation θ. The agent is rewarded when the robot reduces its distance to the objective:

$$R_t = k_1(d_{t-1} - d_t) + k_2(\alpha_{t-1} - \alpha_t), \tag{7.25}$$

where d_t is the linear distance between robot and objective, in meters, α_t is the angular distance, in radians, and $k_1 = 2$ and $k_2 = 3.5$ are proportional constants defined empirically. Each objective has a conquer time of 1.5s, i.e., the robot must be within a small area (less than 15cm from the target position) 1.5s for the high-level controller to generate a new objective. This approach intends to motivate a stable walking-in-place behavior. Otherwise, the robot is always rushing to the next objective, without mastering the subtler skills. An episode ends when the robot falls or after a maximum period of 1000s, although the latter restriction was not applied during the learning stage. To get a higher reward,

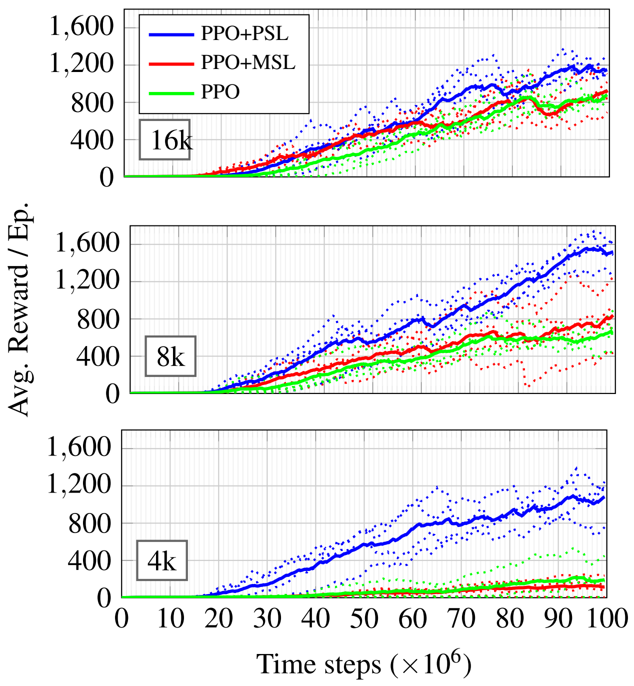


Figure 7.13: Training evolution comparison between PPO, the loss extension presented in this work (PPO+PSL), and the extension proposed by Yu et al. [139] (PPO+MSL). Each plot represents the evolution for 100M time steps, using 8 threads and a batch size per thread of 16k, 8k, and 4k, ordered from top to bottom. The dotted lines correspond to the average reward per episode for a single run. Each solid line is the average of 4 independent runs.

the agent must learn to maximize its speed, stability, and accuracy in order to efficiently conquer each objective.

In the testing stage, in addition to the terrain conditions, an external force was applied to the robot at fixed intervals of 4 seconds, in a random direction parallel to the ground, with a constant duration of 25ms. The applied force magnitude was initially set to 300N, and it was then increased to 400N. Moreover, a relative noise value was added to all observations, ranging from -10% to $+10\%$.

7.13 Sample Efficiency

To evaluate the sample efficiency of the different symmetry loss extensions, the reinforcement learning algorithm was executed with three different batch configurations, corresponding to the three plots in Figure 7.13. All optimizations ran for 100M time steps, using 8 threads. Analyzing the first plot, on the top, there are 12 dotted lines, which show the evolution of the average reward per episode for 12 optimizations, with a batch size per thread of 16k samples: 4 for PPO (green), 4 for our extension PPO+PSL (blue), and 4 for the symmetry mirror loss proposed by Yu et al. [139] (red). There are also 3 solid lines that follow the same color association and represent the mean of the corresponding 4 runs. The middle and bottom plots are equivalent to the top plot, except for the number of samples per batch.

This experiment confirms the intuitive notion that leveraging symmetry knowledge brings an added value to each sample. For 16k samples per batch, the performance of every approach is very similar, and above that size, it drops in the same proportion. It must be noted that the performance drops, not because of excessive samples, but because the number of updates is reduced for the same

Table 7.2: Results comparison for each simulated scenarios

	Force	Scenarios							
		(**a**)		(**b**)		(**c**)		(**d**)	
		t(s)	r/t	t(s)	r/t	t(s)	r/t	t(s)	r/t
Baseline	0N	158	0.46	1	-	2	-	2	-
					0.11		0.51		0.30
	300N	25	0.42	-	-	-	-	-	-
Best Model	0N	976	0.94	520	0.84	35	0.64	55	0.50
	300N	332	0.90	97	0.79	15	0.25	29	0.45
	400N	55	0.86	29	0.74	12	0.28	20	0.41

100M time steps. This constitutes a trade-off between the number of updates and information per update. However, after a while, adding samples to a batch yields no benefit. So, each approach has a certain optimal balance. For our learning scenario, the PPO+PSL approach works best with approximately 8k samples per batch. However, the optimal batch size for PPO and PPO+PSL seems to be around 16k samples. The difference is especially evident in the bottom plot, where each batch only has 4k samples. As the optimizations converge to a global optimum, the reward grows indefinitely, since the episodes have no time limitation during learning. Therefore, this analysis is only focused on sample efficiency, not on comparing the quality of the final solution after a plateau is reached.

7.14 Results Comparison

To evaluate the performance of the framework, the results were compared with a baseline, which is only composed of the walking engine presented in Section 7.10. Table 7.2 shows a comparison between the baseline and the best model developed by the proposed framework. Both models were tested in the four scenarios presented in Figure 7.12, using a force magnitude of 0N, 300N, and 400N. Regarding the baseline, the 400N line was omitted because the robot could not withstand a single push without losing its balance. The same logic can be applied to 300N for scenarios (**b**), (**c**) and (**d**). There are two metrics per scenario: the average episode duration t, measured in seconds; and the average reward per second (obtained by dividing the episode's total reward by its duration). Each simulation configuration was executed for 1000 episodes.

As aforementioned, the simulation was limited to a maximum period of 1000s during the testing stage. Consequently, the average duration of an episode in Table 7.2 can be considered as an average progress per mileage. In scenario (a), the best model has average progress of 976‰, which characterizes the corresponding gait as very stable. Without this limitation, some simulation threads can run almost indefinitely. As expected, the trained model generalized better for scenario (a), than (c) or (d). However, after visually inspecting the results (see simulation videos at `https://youtu.be/mpGYdo_PIMo`), the robot displays very interesting and human-like skills in scenarios (c) and (d). It must be noted that the best model is built on top of the baseline walk engine, which falls almost immediately in scenarios other than (a) and has no prior knowledge of the terrain.

After adding random external forces, the best model can sometimes recover by stepping in the direction of the push, although it was not trained to do that. We conclude that learning to walk on uneven terrain has given the robot the ability to recover its balance by stepping in some direction, even if that results in getting farther from the objective. As an example, in scenario (b), where the robot learned, it can resist 400N pushes for 29 seconds, on average. This is equivalent to approximately 7 pushes per episode, on average.

7.15 Summary

In this chapter, we tackled the problem of developing a robust stabilizer system and an omnidirectional walking engine for biped locomotion. We proposed a framework based on a tight coupling between analytical control and deep reinforcement learning to combine the potential of both approaches. First, we used the LIPM and DCM concepts to develop an analytical control framework to generate robust locomotion. Then, we designed a learning framework that extends the PPO algorithm with symmetric partial data augmentation to learn residuals on top of the analytical approach. This hybrid approach aims at unlocking the full potential of the robot by exploiting the consistency of the analytical solution, the adaptability of neural networks to adjust the control parameters in unforeseen circumstances, and the model's symmetry, while not totally constraining the exploration of asymmetric reactions. The results attest to the models' robustness in considerably noisy environments. The symmetry-enhanced models were able to perform better in the scenarios where they learned but were not able to generalize as well in unforeseen circumstances. However, the difference is partially explained by the way the reward function's influence penalty is less restrictive in challenging conditions. In comparison with an asymmetric approach, our symmetry learning method was more sample efficient and produced a behavior with less position drift, lowering the need for regulation in a high-level controller.

Moreover, we presented a hybrid walking framework by coupling a closed-loop CPG-ZMP walking engine with a DRL algorithm to combine the potential of both approaches. First, a feed-forward CPG-based walking engine was designed based on the LIPM and a set of PFS oscillators. Then, a walking stabilizer was formulated as a set of PD controllers, which were tuned by an expert to have a closed-loop CPG-ZMP walking engine. Additionally, we optimized a policy to adaptively update the walking engine parameters and to generate the steering commands, as well as residual joint position targets which complement the walking engine with model-free skills. This hybrid framework aims at generating robust, versatile, and agile omnidirectional walking gaits by exploring the full potential of the robot, taking advantage of the analytical solution's consistency and the flexibility of residual learning. Moreover, a proximal symmetry loss was proposed to increase the sample efficiency of PPO by leveraging MDP symmetries. To assess the performance of the overall framework, four simulation scenarios were designed. The robot was trained in only one of the scenarios. However, it was able to generalize its knowledge in unforeseen circumstances, displaying very interesting skills, which can be visualized at `https://youtu.be/mpGYdo_PIMo`.

Chapter 8

Conclusions and Future Research Directions

Versatility is a coveted feature when designing humanoid robots. Their shape allows them to be extremely resourceful in our daily-life environments without the necessity of adjusting to their surroundings. According to this distinctive property, humanoid robots have a wide range of applications from working in factories to helping elderly people. Despite a significant effort from the research community, their capabilities are still far from ours, particularly, in terms of speed, stability, and safety. People expect humanoid robots to walk robustly over any type of terrain and be able to recover from external perturbations.

In this thesis, we studied the topics of walking and push recovery and we tackled the problem of developing robust, agile, and versatile biped locomotion frameworks. To this end, we investigated a set of well-known dynamics models and we enhanced LIPPFM by releasing its constraints and compared it with some well-known dynamics models through a bunch of numerical simulations. Based on this model, we designed and developed several walking engines, composed of several layers that were connected hierarchically. This structure fades the complexity and increases the flexibility and we showed that, using this structure, the walking engines were platform-independent and could be easily deployed on other platforms just by few changes. Furthermore, we formulated the push recovery strategies as a set of controllers including classical and optimal controllers, and performed simulations as well as real robot experiments to validate their performances. Besides, due to the parametric nature of the proposed walking engines, we used optimization algorithms like GA and CREPS-CMA to find the optimum parameters of the frameworks in different scenarios. Moreover, we believed that using machine learning on top of analytical approaches is the key to open doors for humanoid robots to step out of laboratories. Therefore, we proposed modular frameworks to generate robust biped locomotion using a tight coupling between an analytical control approach and deep reinforcement learning. The core of this framework was a specific dynamics model which abstracts a humanoid's dynamics model into two masses. This dynamics model was used to design an adaptive reference trajectories planner and a controller which were fully parametric. Furthermore, a learning framework was augmented with this framework to learn how to improve the stability of the robot by moving the arms and changing the height of the robot's COM. Additionally, we used LIPM and DCM concepts to formulate the biped locomotion and stabilizer as an analytical control framework. On top of that, a neural network with symmetric partial data augmentation learned residuals to adjust the joint's positions, thus improving the robot's stability when facing external perturbations. According to these results, we tackled the problem of developing a robust omnidirectional walking framework, which was able to generate a versatile and agile locomotion on complex terrains. To this end, we proposed a proximal symmetry loss to increase the sample efficiency of the PPO algorithm by leveraging model symmetries. The effectiveness of the proposed framework was demonstrated and evaluated across a set of challenging simulation scenarios. We showed that the robot was able to generalize what it learned in one scenario, by displaying human-like locomotion skills in unforeseen circumstances, even in the presence of noise and external pushes.

8.1 Future Research Directions

Despite the promising results presented in this thesis, we believe that future autonomous, cognitive robots need to be able to automatically generate models based on information which is extracted from the data streams accessible to the robot. Moreover, it may be desired to overcome the limitations of using deep reinforcement learning in real robot applications. To this end, coupling MPC and DRL algorithms to combine the potential of both approaches can be a good starting point.

Another idea worth exploring is developing locomotion frameworks based on safe policy learning for continuous control approaches. In these approaches, robots interact with the environment through near-safe policies which means during the training and testing, the agents execute safe policies.

Bibliography

[1] S. Faraji and A. J. Ijspeert, “3LP: A linear 3D-walking model including torso and swing dynamics,” *the international journal of robotics research*, vol. 36, no. 4, pp. 436–455, 2017.

[2] M. Missura and S. Behnke, “Gradient-driven online learning of bipedal push recovery,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 387–392.

[3] X. Xinjilefu, S. Feng, and C. G. Atkeson, “Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 67–73.

[4] M. Kelly and A. Ruina, “Non-linear robust control for inverted-pendulum 2D walking,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4353–4358.

[5] S. Kajita and K. Tan, “Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum model,” in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1405–1411.

[6] M. Vukobratovic, A. Frank, and D. Juricic, “On the stability of biped locomotion,” *Biomedical Engineering, IEEE Transactions on*, no. 1, pp. 25–36, 1970.

[7] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, “Biped walking pattern generation by using preview control of Zero-Moment point,” in *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1620–1626.

[8] S. M. Kasaei, N. Lau, and A. Pereira, “A reliable hierarchical omnidirectional walking engine for a bipedal robot by using the enhanced LIP plus flywheel,” in *Human-centric Robotics-Proceedings of The 20th International Conference Clawar 2017*. World Scientific, 2017, p. 399.

[9] B. W. Bestbury, “*R*-matrices and the magic square,” *J. Phys. A*, vol. 36, no. 7, pp. 1947–1959, 2003.

[10] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi, “Biped walking stabilization based on linear inverted pendulum tracking,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4489–4496.

[11] S. Shimmyo, T. Sato, and K. Ohnishi, “Biped walking pattern generation by using preview control based on three-mass model,” *IEEE transactions on industrial electronics*, vol. 60, no. 11, pp. 5137–5147, 2013.

[12] Y.-J. Ryoo, “Walking engine using zmp criterion and feedback control for child-sized humanoid robot,” *International Journal of Humanoid Robotics*, vol. 13, no. 04, p. 1650021, 2016.

[13] R. Tedrake, S. Kuindersma, R. Deits, and K. Miura, “A closed-form solution for real-time ZMP gait generation and feedback stabilization,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 936–940.

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

[15] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, “Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 1028–1035.

[16] K. H. Koch, K. Mombaur, and P. Soueres, “Optimization-based walking generation for humanoid robot,” *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 498–504, 2012.

[17] S. Mason, L. Righetti, and S. Schaal, “Full dynamics LQR control of a humanoid robot: An experimental study on balancing and squatting,” in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 374–379.

[18] S. Hong, Y. Oh, D. Kim, S. Ra, and B.-J. You, “Walking pattern generation method with feedforward and feedback control for humanoid robots,” in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*. IEEE, 2009, pp. 263–268.

[19] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, “Online walking motion generation with automatic footstep placement,” *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[20] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, “The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation,” in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 239–246.

[21] S. M. Kasaei, D. Simões, N. Lau, and A. Pereira, “A hybrid ZMP-CPG based walk engine for biped robots,” in *Iberian Robotics conference*. Springer, 2017, pp. 743–755.

[22] H. Picado, M. Gestal, N. Lau, L. Reis, and A. Tomé, “Automatic generation of biped walk behavior using genetic algorithms,” *Bio-inspired systems: Computational and ambient intelligence*, pp. 805–812, 2009.

[23] S. Asta and S. Sariel-Talay, “Nature-inspired optimization for biped robot locomotion and gait planning,” *Applications of Evolutionary Computation*, pp. 434–443, 2011.

[24] J. Or, “A hybrid CPG–ZMP control system for stable walking of a simulated flexible spine humanoid robot,” *Neural Networks*, vol. 23, no. 3, pp. 452–460, 2010.

[25] A. Massah, A. Zamani, Y. Salehinia, M. A. Sh, and M. Teshnehlab, “A hybrid controller based on CPG and ZMP for biped locomotion,” *Journal of Mechanical science and technology*, vol. 27, no. 11, pp. 3473–3486, 2013.

[26] S. Gay, J. Santos-Victor, and A. Ijspeert, “Learning robot gait stability using neural networks as sensory feedback function for central pattern generators,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ieee, 2013, pp. 194–201.

[27] M. Missura and S. Behnke, “Omnidirectional capture steps for bipedal walking,” in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE, 2013, pp. 14–20.

[28] ——, “Self-stable omnidirectional walking with compliant joints,” in *Proceedings of 8th Workshop on Humanoid Soccer Robots, IEEE Int. Conf. on Humanoid Robots, Atlanta, USA*, 2013.

[29] S.-J. Yi, B.-T. Zhang, D. Hong, and D. D. Lee, “Online learning of a full body push recovery controller for omnidirectional walking,” in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 1–6.

[30] K.-T. Song and C.-H. Hsieh, “CPG-based control design for bipedal walking on unknown slope surfaces,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5109–5114.

[31] C. Liu, D. Wang, and Q. Chen, “Central pattern generator inspired control for adaptive walking of biped robots,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1206–1215, 2013.

[32] J. Cristiano, D. Puig, and M. Garcıa, “Locomotion control of biped robots on uneven terrain through a feedback CPG network,” in *XIV Workshop of Physical Agents*, 2013, pp. 1–6.

[33] J. Nassour, P. Hénaff, F. Benouezdou, and G. Cheng, “Multi-layered multi-pattern CPG for adaptive locomotion of humanoid robots,” *Biological cybernetics*, vol. 108, no. 3, pp. 291–303, 2014.

[34] J. Liu and O. Urbann, “Bipedal walking with dynamic balance that involves three-dimensional upper body motion,” *Robotics and Autonomous Systems*, vol. 77, pp. 39–54, 2016.

[35] S. Behnke, “Online trajectory generation for omnidirectional biped walking,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 1597–1603.

[36] S. M. Kasaei, S. H. Kasaei, E. Shahri, A. Ahmadi, N. Lau, and A. Pereira, “How to select a suitable action against strong pushes in adult-size humanoid robot: Learning from past experiences,” in *Autonomous Robot Systems and Competitions (ICARSC), 2016 International Conference on*. IEEE, 2016, pp. 80–86.

[37] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, “Capture point: A step toward humanoid push recovery,” in *2006 6th IEEE-RAS international conference on humanoid robots*. IEEE, 2006, pp. 200–207.

[38] B. Stephens, "Humanoid push recovery," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*. IEEE, 2007, pp. 589–595.

[39] T. Komura, H. Leung, S. Kudoh, and J. Kuffner, "A feedback controller for biped humanoids that can counteract large perturbations during gait," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 1989–1995.

[40] A. Abdolmaleki, N. Shafii, L. P. Reis, N. Lau, J. Peters, and G. Neumann, "Omnidirectional walking with a compliant inverted pendulum model." in *IBERAMIA*, 2014, pp. 481–493.

[41] K. Erbatur, A. Okazaki, K. Obiya, T. Takahashi, and A. Kawamura, "A study on the zero moment point measurement for biped walking robots," in *Advanced Motion Control, 2002. 7th International Workshop on*. IEEE, 2002, pp. 431–436.

[42] T. Hemker, H. Sakamoto, M. Stelzer, and O. von Stryk, "Hardware-in-the-loop optimization of the walking speed of a humanoid robot," in *Proc. of CLAWAR*, 2006, p. 20.

[43] P. A. Guertin, "The mammalian central pattern generator for locomotion," *Brain research reviews*, vol. 62, no. 1, pp. 45–56, 2009.

[44] A. Abdolmaleki, D. Simoes, N. Lau, L. P. Reis, and G. Neumann, "Contextual relative entropy policy search with covariance matrix adaptation," in *Autonomous Robot Systems and Competitions (ICARSC), 2016 IEEE International Conference on*. IEEE, May 2016, pp. 94–99.

[45] R. Hamming, "Lanczos' σ factors and the σ factors in the general case 32.6 and 32.7," *Numerical Methods for Scientists and Engineers*, pp. 534–536, 1986.

[46] H. Hemami and C. Golliday Jr, "The inverted pendulum and biped stability," *Mathematical Biosciences*, vol. 34, no. 1-2, pp. 95–110, 1977.

[47] A. Albert and W. Gerth, "Analytic path planning algorithms for bipedal robots without a trunk," *Journal of Intelligent and Robotic Systems*, vol. 36, no. 2, pp. 109–127, 2003.

[48] S. M. Kasaei, N. Lau, A. Pereira, and E. Shahri, "A reliable model-based walking engine with push recovery capability," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, April 2017, pp. 122–127.

[49] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot-1st report: Walking gait pattern generation," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1084–1091.

[50] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, 2015.

[51] J. Englsberger, G. Mesesan, and C. Ott, "Smooth trajectory generation and push-recovery based on divergent component of motion," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4560–4567.

[52] T. Kamioka, H. Kaneko, T. Takenaka, and T. Yoshiike, "Simultaneous optimization of ZMP and footsteps based on the analytical solution of divergent component of motion," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1763–1770.

[53] P. Kryczka, P. Kormushev, N. G. Tsagarakis, and D. G. Caldwell, "Online regeneration of bipedal walking gait pattern optimizing footstep placement and timing," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3352–3357.

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

[55] M. Khadiv, S. Kleff, A. Herzog, S. A. A. Moosavian, S. Schaal, and L. Righetti, "Stepping stabilization using a combination of DCM tracking and step adjustment," in *Robotics and Mechatronics (ICROM), 2016 4th International Conference on*. IEEE, 2016, pp. 130–135.

[56] M. A. Hopkins, D. W. Hong, and A. Leonessa, "Humanoid locomotion on uneven terrain using the time-varying divergent component of motion," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 266–272.

[57] Y. Zhao and L. Sentis, "A three dimensional foot placement planner for locomotion in very rough terrains," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*. IEEE, 2012, pp. 726–733.

[58] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control using divergent component of motion," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 2600–2607.

[59] H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1121–1126.

[60] R. J. Griffin, G. Wiedebach, S. Bertrand, A. Leonessa, and J. Pratt, "Walking stabilization using step timing and location adjustment on the humanoid robot, atlas," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 667–673.

[61] M. Kasaei, N. Lau, and A. Pereira, "A robust biped locomotion based on Linear-Quadratic-Gaussian controller and divergent component of motion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 1429–1434.

[62] M. M. Kasaei, N. Lau, and A. Pereira, "A model-based biped walking controller based on divergent component of motion," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2019, pp. 1–6.

[63] J. Yamaguchi, S. G. Eiji, S. Inoue, and A. Takanishi, "Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 1. IEEE, 1999, pp. 368–374.

[64] O. Khatib, L. Sentis, and J.-H. Park, "A unified framework for whole-body humanoid robot control with multiple constraints and contacts," in *European Robotics Symposium 2008*. Springer, 2008, pp. 303–312.

[65] K. Ishihara, T. D. Itoh, and J. Morimoto, "Full-body optimal control toward versatile and agile behaviors in a humanoid robot," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 119–126, 2019.

[66] J. Shan, C. Junshi, and C. Jiapin, "Design of central pattern generator for humanoid robot walking based on multi-objective ga," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 1930–1935.

[67] J. Lee and K. Seo, "Generation of walking trajectory of humanoid robot using cpg," *Journal of Korean Institute of Intelligent Systems*, vol. 23, no. 4, pp. 360–365, 2013.

[68] J. Yu, M. Tan, J. Chen, and J. Zhang, "A survey on cpg-inspired control models and system implementation," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 3, pp. 441–456, 2013.

[69] G. Zhong, N. A. Shevtsova, I. A. Rybak, and R. M. Harris-Warrick, "Neuronal activity in the isolated mouse spinal cord during spontaneous deletions in fictive locomotion: insights into locomotor central pattern generator organization," *The Journal of physiology*, vol. 590, no. 19, pp. 4735–4759, 2012.

[70] E. Menelaou and D. L. McLean, "Hierarchical control of locomotion by distinct types of spinal v2a interneurons in zebrafish," *Nature communications*, vol. 10, no. 1, pp. 1–12, 2019.

[71] M. Kasaei, N. Lau, and A. Pereira, "A fast and stable omnidirectional walking engine for the NAO humanoid robot," in *Robot World Cup*. Springer, 2019, pp. 99–111.

[72] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[73] M. Abreu, L. P. Reis, and N. Lau, "Learning to run faster in a humanoid robot soccer environment through reinforcement learning," in *Robot World Cup*. Springer, 2019, pp. 3–15.

[74] P. MacAlpine, S. Barrett, D. Urieli, V. Vu, and P. Stone, "Design and optimization of an omnidirectional humanoid walk: A winning approach at the robocup 2011 3d simulation competition," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[75] B. He, Z. Wang, R. Shen, and S. Hu, "Real-time walking pattern generation for a biped robot with hybrid CPG–ZMP algorithm," *International Journal of Advanced Robotic Systems*, vol. 11, no. 10, p. 160, 2014.

[76] I. Koryakovskiy, M. Kudruss, H. Vallery, R. Babuška, and W. Caarls, "Model-plant mismatch compensation using reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2471–2477, 2018.

[77] A. Abdolmaleki, D. Simoes, N. Lau, L. P. Reis, and G. Neumann, "Contextual relative entropy policy search with covariance matrix adaptation," in *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2016, pp. 94–99.

[78] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.

[79] S. Kajita, M. Benallegue, R. Cisneros, T. Sakaguchi, M. Morisawa, K. G. Hiroshi, I. Kumagai, K. Kaneko, and F. Kanehiro, "Position-based lateral balance control for knee-stretched biped robot," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2019, pp. 17–24.

[80] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. 1707.06347, 2017.

[81] L. Carvalho Melo and M. R. Omena Albuquerque Máximo, "Learning humanoid robot running skills through proximal policy optimization," in *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, 2019, pp. 37–42.

[82] H. Teixeira, T. Silva, M. Abreu, and L. P. Reis, "Humanoid robot kick in motion ability for playing robotic soccer," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2020, pp. 34–39.

[83] D. C. Melo, M. R. O. A. Máximo, and A. M. da Cunha, "Push recovery strategies through deep reinforcement learning," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. IEEE, 2020, pp. 240–245.

[84] M. Abreu, N. Lau, A. Sousa, and L. P. Reis, "Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2019, pp. 1–8.

[85] A. F. V. Muzio, M. R. O. A. Maximo, and T. Yoneyama, "Deep reinforcement learning for humanoid robot dribbling," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. IEEE, 2020, pp. 246–251.

[86] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. 1412.6980, 2017.

[87] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *CoRR*, vol. 1506.02438, 2018.

[88] N. Shafii, L. P. Reis, and N. Lau, "Biped walking using coronal and sagittal movements based on truncated fourier series," in *RoboCup 2010: Robot Soccer World Cup XIV*, J. Ruiz-del Solar, E. Chown, and P. G. Plöger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 324–335.

[89] R. J. Griffin and A. Leonessa, "Model predictive control for dynamic footstep adjustment using the divergent component of motion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1763–1768.

[90] P. MacAlpine and P. Stone, "Ut austin villa: Robocup 2017 3D simulation league competition and technical challenges champions," in *Robot World Cup*. Springer, 2017, pp. 473–485.

[91] M. Kasaei, A. Ahmadi, N. Lau, and A. Pereira, "A robust model-based biped locomotion framework based on three-mass model: From planning to control," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2020, pp. 257–262.

[92] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[93] C. P. Santos, N. Alves, and J. C. Moreno, "Biped locomotion control through a biomimetic cpg-based controller," *Journal of Intelligent & Robotic Systems*, vol. 85, no. 1, pp. 47–70, 2017.

[94] D. H. Tran, F. Hamker, and J. Nassour, "A humanoid robot learns to recover perturbation during swinging motion," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[95] C.-H. Chang, H.-P. Huang, H.-K. Hsu, and C.-A. Cheng, "Humanoid robot push-recovery strategy based on cmp criterion and angular momentum regulation," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2015, pp. 761–766.

[96] C. Li, R. Lowe, and T. Ziemke, "Humanoids learning to walk: a natural cpg-actor-critic architecture," *Frontiers in neurorobotics*, vol. 7, p. 5, 2013.

[97] S. Kajita, O. Matsumoto, and M. Saigo, "Real-time 3D walking pattern generation for a biped robot with telescopic legs," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 2299–2306.

[98] A. Herdt, N. Perrin, and P.-B. Wieber, "Walking without thinking about it," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 190–195.

[99] C. Brasseur, A. Sherikov, C. Collette, D. Dimitrov, and P.-B. Wieber, "A robust linear MPC approach to online generation of 3D biped walking motion," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 595–601.

[100] R. Mirjalili, A. Yousefi-Korna, F. A. Shirazi, A. Nikkhah, F. Nazemi, and M. Khadiv, "A whole-body model predictive control scheme including external contact forces and COM height variations," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–6.

[101] R. C. Luo, K. C. Lee, and A. Spalanzani, "Humanoid robot walking pattern generation based on five-mass with angular momentum model," in *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2016, pp. 375–380.

[102] T. Sato, S. Sakaino, and K. Ohnishi, "Real-time walking trajectory generation method with three-mass models at constant body height for three-dimensional biped robots," *IEEE transactions on industrial electronics*, vol. 58, no. 2, pp. 376–383, 2010.

[103] J. Luo, Y. Su, L. Ruan, Y. Zhao, D. Kim, L. Sentis, and C. Fu, "Robust bipedal locomotion based on a hierarchical control structure." *Robotica*, vol. 37, no. 10, pp. 1750–1767, 2019.

[104] A. Hornung and M. Bennewitz, "Adaptive level-of-detail planning for efficient humanoid navigation," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 997–1002.

[105] R. J. Griffin, G. Wiedebach, S. McCrory, S. Bertrand, I. Lee, and J. Pratt, "Footstep planning for autonomous walking over rough terrain," *arXiv preprint arXiv:1907.08673*, 2019.

[106] A. Dasgupta and Y. Nakamura, "Making feasible walking motion of humanoid robots from human motion capture data," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1044–1049.

[107] E. Spyrakos-Papastavridis, G. A. Medrano-Cerda, N. G. Tsagarakis, J. S. Dai, and D. G. Caldwell, "A push recovery strategy for a passively compliant humanoid robot using decentralized LQR controllers," in *2013 IEEE International Conference on Mechatronics (ICM)*. IEEE, 2013, pp. 464–470.

[108] M. Vukobratović and B. Borovac, "Zero-moment point—thirty five years of its life," *International journal of humanoid robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[109] J. Englsberger and C. Ott, "Integration of vertical com motion and angular momentum in an extended capture point tracking controller for bipedal walking," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 183–189.

[110] M. Morisawa, N. Kita, S. Nakaoka, K. Kaneko, S. Kajita, and F. Kanehiro, "Biped locomotion control for uneven terrain with narrow support region," in *System Integration (SII), 2014 IEEE/SICE International Symposium on*. IEEE, 2014, pp. 34–39.

[111] S. Mason, N. Rotella, S. Schaal, and L. Righetti, "Balancing and walking using full dynamics LQR control with contact constraints," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 63–68.

[112] S. Faraji, H. Razavi, and A. J. Ijspeert, "Bipedal walking and push recovery with a stepping strategy based on time-projection control," *The International Journal of Robotics Research*, vol. 38, no. 5, pp. 587–611, 2019.

[113] M. Kasaei, N. Lau, and A. Pereira, "A robust biped locomotion based on Linear-Quadratic-Gaussian controller and divergent component of motion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1429–1434.

[114] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2006, pp. 137–142.

[115] S. Caron, "Biped stabilization by linear feedback of the variable-height inverted pendulum model," in *IEEE International Conference on Robotics and Automation*, May 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02289919

[116] S. Kajita, M. Benallegue, R. Cisneros, T. Sakaguchi, S. Nakaoka, M. Morisawa, K. G. Hiroshi, I. Kumagai, K. Kaneko, and F. Kanehiro, "Biped gait control based on spatially quantized dynamics," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 75–81.

[117] T. Seyde, A. Shrivastava, J. Englsberger, S. Bertrand, J. Pratt, and R. J. Griffin, "Inclusion of angular momentum during planning for capture point based walking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1791–1798.

[118] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, 2020.

[119] D. W. Evans, P. T. Orr, S. M. Lazar, D. Breton, J. Gerard, D. H. Ledbetter, K. Janosco, J. Dotts, and H. Batchelder, "Human preferences for symmetry: Subjective experience, cognitive conflict and cortical brain activity," *PLoS ONE*, vol. 7, no. 6, 2012.

[120] I. Handžić and K. B. Reed, "Perception of gait patterns that deviate from normal and symmetric biped locomotion," *Frontiers in psychology*, vol. 6, p. 199, 2015.

[121] H. S. Baird, H. Bunke, and K. Yamamoto, *Structured document image analysis*. Springer Science & Business Media, 2012.

[122] S. Chen, E. Dobriban, and J. H. Lee, "A group-theoretic framework for data augmentation," arXiv preprint arXiv:1907.10905, 2020.

[123] A. Agostini and E. Celaya, "Exploiting domain symmetries in reinforcement learning with continuous state and action spaces," in *2009 International Conference on Machine Learning and Applications*. IEEE, 2009, pp. 331–336.

[124] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. van de Panne, "On learning symmetric locomotion," in *Motion, Interaction and Games*, 2019, pp. 1–10.

[125] Y. Lin, J. Huang, M. Zimmer, Y. Guan, J. Rojas, and P. Weng, "Invariant transform experience replay: Data augmentation for deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6615–6622, 2020.

[126] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[127] B. Ravindran and A. G. Barto, "Symmetries and model minimization in markov decision processes," USA, Tech. Rep., 2001.

[128] D. Surovik, K. Wang, M. Vespignani, J. Bruce, and K. E. Bekris, "Adaptive tensegrity locomotion: Controlling a compliant icosahedron with symmetry-reduced reinforcement learning," *The International Journal of Robotics Research*, 2019.

[129] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 2020, pp. 317–329.

[130] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames, "Dynamic humanoid locomotion: A scalable formulation for hzd gait optimization," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 370–387, 2018.

[131] X. B. Peng, G. Berseth, K. Yin, and M. van de Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. on Graphics (Proc. SIGGRAPH 2017)*, vol. 36, no. 4, 2017.

[132] R. van Bree, "Data augmentation for regularizing learned world models in reinforcement learning," Master's thesis, University of Twente, 2021.

[133] Y. Lin, J. Huang, M. Zimmer, Y. Guan, J. Rojas, and P. Weng, "Invariant transform experience replay: Data augmentation for deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6615–6622, 2020.

[134] S. Mishra, A. Abdolmaleki, A. Guez, P. Trochim, and D. Precup, "Augmenting learning using symmetry in a biologically-inspired domain," arXiv preprint arXiv:1910.00528, 2019.

[135] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[136] E. van der Pol, D. E. Worrall, H. van Hoof, F. A. Oliehoek, and M. Welling, "MDP homomorphic networks: Group symmetries in reinforcement learning," in *Advances in Neural Information Processing Systems*, 2020.

[137] A. Mahajan and T. Tulabandhula, "Symmetry learning for function approximation in reinforcement learning," *arXiv preprint arXiv:1706.02999*, 2017.

[138] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[139] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.

[140] H. Jeong, I. Lee, J. Oh, K. K. Lee, and J.-H. Oh, "A robust walking controller based on online optimization of ankle, hip, and stepping strategies," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1367–1386, 2019.

[141] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Walking control based on step timing adaptation," *IEEE Transactions on Robotics*, 2020.

[142] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[143] A. J. Ratner, H. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré, "Learning to compose domain-specific transformations for data augmentation," in *Advances in neural information processing systems*, 2017, pp. 3236–3246.

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

[145] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[146] J. Ahn, J. Lee, and L. Sentis, "Data-efficient and safe learning for humanoid locomotion aided by a dynamic balancing model," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4376–4383, 2020.

[147] Y. Wang, X. Xue, and B. Chen, "Matsuoka's cpg with desired rhythmic signals for adaptive walking of humanoid robots," *IEEE transactions on cybernetics*, vol. 50, no. 2, pp. 613–626, 2018.

[148] K. Matsuoka, "Sustained oscillations generated by mutually inhibiting neurons with adaptation," *Biological cybernetics*, vol. 52, no. 6, pp. 367–376, 1985.

[149] N. Van der Noot, A. J. Ijspeert, and R. Ronsse, "Bio-inspired controller achieving forward speed modulation with a 3D bipedal walker," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 168–196, 2018.

[150] A. A. Saputra, A. J. Ijspeert, and N. Kubota, "A neural primitive model with sensorimotor coordination for dynamic quadruped locomotion with malfunction compensation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 3783–3788.

[151] X. Liu, R. Gasoto, Z. Jiang, C. Onal, and J. Fu, "Learning to locomote with artificial neural-network and cpg-based control in a soft snake robot," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7758–7765.

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

[153] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2020.

[154] R. Robinson, W. Herzog, and B. M. Nigg, "Use of force platform variables to quantify the effects of chiropractic manipulation on gait symmetry," *Journal of manipulative and physiological therapeutics*, vol. 10, no. 4, pp. 172–176, 1987.

[155] N. G. Tsagarakis, Z. Li, J. Saglia, and D. G. Caldwell, "The design of the lower body of the compliant humanoid robot "cCub"," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2035–2040.

[156] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[157] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.

[158] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.