Robust Bipedal Locomotion over Rough Terrain by extending ZMP-based Control

Robuste zweibeinige Fortbewegung über unebene Gelände durch Erweiterungen ZMP-basierter Regelungskonzepten
Master-Thesis von Anaïs Reynaud (Studiengang Elektrotechnik und Informationstechnik)
Tag der Einreichung:

1. Gutachten: Prof. Dr. Oskar Von Stryk, Fachgebiet Simulation, Systemoptimierung und Robotik
2. Gutachten: Prof. Dr.-Ing. Ulrich Konigorski, Fachgebiet Regelungstechnik und Mechatronik

TECHNISCHE UNIVERSITÄT DARMSTADT

sim rtm

Robust Bipedal Locomotion over Rough Terrain by extending ZMP-based Control
Robuste zweibeinige Fortbewegung über unebene Gelände durch Erweiterungen ZMP-basierter Regelungskonzepten

Vorgelegte Master-Thesis von Anaïs Reynaud (Studiengang Elektrotechnik und Informationstechnik)

1. Gutachten: Prof. Dr. Oskar Von Stryk, Fachgebiet Simulation, Systemoptimierung und Robotik
2. Gutachten: Prof. Dr.-Ing. Ulrich Konigorski, Fachgebiet Regelungstechnik und Mechatronik

Tag der Einreichung:

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Anaïs Reynaud, die vorliegende Master-Thesis / Bachelor-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden. Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

English translation for information purposes only:
Thesis Statement pursuant to § 22 paragraph 7 and § 23 paragraph 7 of APB TU Darmstadt I herewith formally declare that I, Anaïs Reynaud, have written the submitted thesis independently pursuant to § 22 paragraph 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form. I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once. In the submitted thesis the written copies and the electronic version for archiving are pursuant to § 23 paragraph 7 of APB identical in content.

Datum/Date:　　　　Unterschrift/Signature:

Abstract

Humanoid robotics is a very active and recent research field, that aims at developing robots which are suitable to interact with an environment designed for humans. One of the biggest challenges is the generation and control of a stable dynamic biped locomotion on any terrain. In order to ensure the dynamic stability, the Zero-Moment Point (ZMP) criterion has been widely used. Based on it, the ZMP-Preview Control proposed by Kajita et al. has become the most used walking approach for bipedal robots. Although the basic method focuses on generating a stable horizontal motion for walking on flat ground, various extensions have then been proposed for uneven terrain scenarios which are discussed in this work.
A basic ZMP-Preview Control limited to walking on flat ground is already implemented on the humanoid robot "Johnny #5", which is used as experimental platform in this work. Hence, this thesis investigates how to extend this existing software to enable walking motions on uneven terrain. The application of a unique method using Virtual Slopes shows first promising results and was successfully tested in simulation on uneven terrain such as stairs. On the real robot, a robust ZMP Balance Control is additionally required. For this reason, the already existing approach was re-designed by combining state of the art balance controllers. In order to evaluate and tune the controller performance, visualization tools are provided by the newly implemented software stack as well.

Keywords: humanoid robot, biped locomotion, zero-moment point, preview control, virtual slope, balance control.

Zusammenfassung

Humanoide Robotik ist ein sehr aktives und aktuelles Forschungsgebiet, das darauf abzielt, Roboter zu entwickeln, die mit einer für Menschen gedachten Umgebung interagieren können. Eine der größten Herausforderungen ist die Steuerung und Regelung einer stabilen dynamischen, zweibeinigen Fortbewegung auf beliebigem Gelände. Um die dynamische Stabilität zu gewährleisten, ist das *Zero-Moment Point (ZMP)* Kriterium weit verbreitet. Basierend darauf hat sich die von Kajita vorgeschlagene *ZMP-Preview Control* Methode zum meistgenutzten Ansatz für zweibeinige Roboter entwickelt. Obwohl sich die Grundmethode auf die Erzeugung einer stabilen horizontalen Bewegung für das Gehen auf flachem Boden konzentriert, wurden verschiedene Erweiterungen für unebene Geländeszenarien vorgeschlagen, die in dieser Arbeit diskutiert werden.
Ein einfaches *ZMP-Preview Control*, das sich auf das Gehen auf flachem Boden beschränkt, ist bereits auf dem humanoiden Roboter "Johnny #5" implementiert, der in dieser Arbeit als experimentelle Plattform dient. Deshalb wird in dieser Thesis untersucht, wie die bestehende Software erweitert werden kann, um eine Gehbewegung auf unebenem Gelände zu ermöglichen. Die Anwendung einer einzigartigen Methode mit virtuellen Steigungen zeigt erste vielversprechende Ergebnisse und wurde erfolgreich in der Simulation auf unebenem Gelände wie Treppen getestet. Am realen Roboter ist zusätzlich eine robuste *ZMP* Balance Regelung erforderlich. Aus diesem Grund wurde die bereits bestehende Implementierung durch die Kombination modernster Bilanzregler neu konzipiert. Um die Leistung der Regelung zu bewerten und einzustellen, werden auch Visualisierungstools durch den neu implementierten Software-Stack bereitgestellt.

Contents

1 Introduction

The Kobe earthquake in 1995, one of the worst in Japan's history in terms of victims and damage, has been the trigger for the development of rescue robotics [1]. The difficulty and inefficiency of the disaster response back then pushed roboticists to design new systems to support the rescue teams when the environment is too hazardous for humans. Sixteen years later the nuclear catastrophe of Fukushima once again urged roboticists to redouble their efforts to build robust and performing rescue robots.

In order to promote the research in this field numerous competitions have been organized after these incidents. In 2001 the yearly competition RoboCup, which was originally composed of robot soccer leagues only, was supplemented with a rescue league for mobile robots. The Defense Advanced Research Projects Agency (DARPA) organized a competition named the DARPA Robotics Challenge (DRC) in several phases between 2013 and 2015 for humanoid rescue robots. Capabilities of rescue robots that are tested in such competitions are for instance:

- cartography and inspection of the damaged environment
- victims localization
- communication with victims through robot microphone and speaker
- clearing the way (move obstructions)
- cleanup (for instance removal of radioactive element)

The robots are either fully autonomous or remote-controlled by an operator from a safe distance or the robots are even capable of doing tasks autonomously.

Rescue robots can take different shapes depending on the task they are meant for and can be combined in disaster response. The SIM lab[1] of the TU Darmstadt[2] participates to such competitions with three kinds of robots: mobile robots with tracks, drones and humanoid robots. The third type is of interest in this thesis.

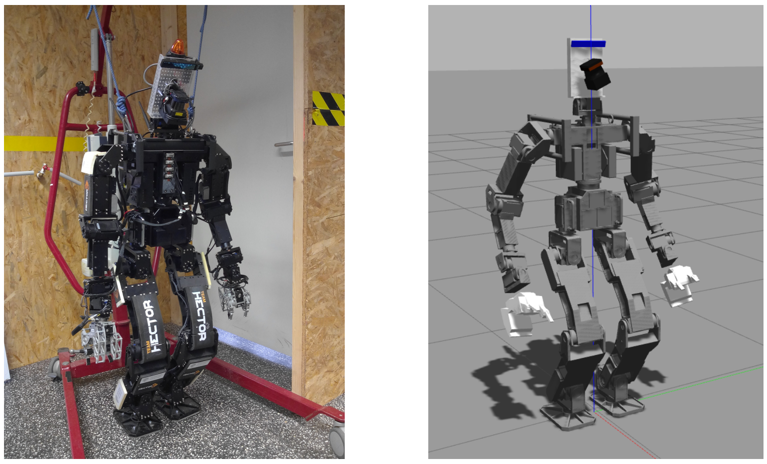
1.1 Motivation

Designing human-like robots permits them to better interact with an environment that was built for humans. They are for instance more suitable to open doors than drones or to climb up stairs than mobile robots. On the other hand one of the biggest challenges in humanoid robotics is undoubtedly the robustness of the legged locomotion. Such platforms are generally much more complex to control than mobile robots or drones because of their natural instability, the large number of actuators and the high height of the Center of Mass (CoM). Moreover in a catastrophic scenario humanoid robots should be capable of traversing difficult terrain such as uneven damaged floors with debris, introducing more challenges for legged locomotion control.

The SIM lab disposes of a humanoid robot of the company Robotis, which is shown in Figure 1.1 and described more in detail in Section 1.2. Before the beginning of this thesis it could only walk on a perfectly even floor. Hence, the focus of this thesis is to improve the locomotion such that the robot

[1] Lab of the Institut for Simulation, Optimization and Robotics
[2] Technical University of Darmstadt

is capable of walking up and down stairs and slopes (during straight forward motion). These are the first types of terrain that should be mastered before trying on even more complex terrain. In order to achieve this goal the generation of a whole-body motion taking pre-planned footsteps into account and its feedback control will be studied here. The planning of the desired footsteps based on the perception of the environment is not part of this thesis.



(a) Real robot **(b)** Simulation (Gazebo)

Figure 1.1: Johnny #5, the humanoid robot of the SIM lab

1.2 THORMANG3

THORMANG3 is a humanoid robot platform designed and sold by the South Korean company Robotis, whereby THOR is the acronym for Tactical Hazardous Operations Robot [2]. The SIM lab possesses the third version of this platform, which was named Johnny #5. The initial state of the hardware and software before the start of this thesis is described in Section 1.2.1 and Section 1.2.2 respectively.

In the rest of this thesis the term THORMANG3 will be used for referring to the robot platform in general and the name Johnny #5 when especially considering the robot of the SIM lab with all individual changes that were made.

1.2.1 Hardware Description

Johnny #5 has 33 active Degrees of Freedom (DoFs). For the locomotion only the actuators of the legs will be considered (12 DoFs). The 1-DoF hands of Robotis were replaced by 2-DoFs hands from Virginia Tech University for better manipulation [3]. The actuators are Dynamixel Pro servomotors with integrated encoders and position control.

The SIM lab placed two PCs on the sides of the robot (one dedicated for motion tasks and one for perception tasks) and a router on its back for wireless communication. This way an operator can send commands to the robot and receive feedback information from the computers of the latter.

An overview of the used sensors is illustrated in Figure 1.2. The Force/Torque (F/T) sensors in the wrists are used for manipulation tasks, while the F/T sensors in the ankles and the Inertial Measurement Unit (IMU) enable to control the locomotion of the robot. Moreover, a camera and a rotating scanning laser have been added in order to capture data about the environment for the perception software. All the hardware components are listed in Table 1.1.

Table 1.1: Specifications of Johnny #5 [4]

Weight	48.5 kg
Height	1.53 m
Active DoF	Head: 2 DoFs Arm: 7 DoFs (x2) Hand: 2 DoFs (x2) Waist: 1 DoF Leg including hip: 6 DoFs (x2)
Controllers (PCs)	Quanmax KEEX-8100 with Intel® Core™ i7-4800MQ processor (x2)
Sensors	Camera: Intel® RealSense™ R200 Scanning Laser: Hokuyo UTM-30LX-EW F/T Sensors Wrists: ATI Mini45 SI-290-10 (x2) F/T Sensors Ankles: ATI Mini 58 SI-1400-60 (x2) IMU: Microstrain 3DM-GX3-45
Actuators	DYNAMIXEL PRO H-Series
Batteries	LiPo 11000 mA 6S 22.2 V (x2)

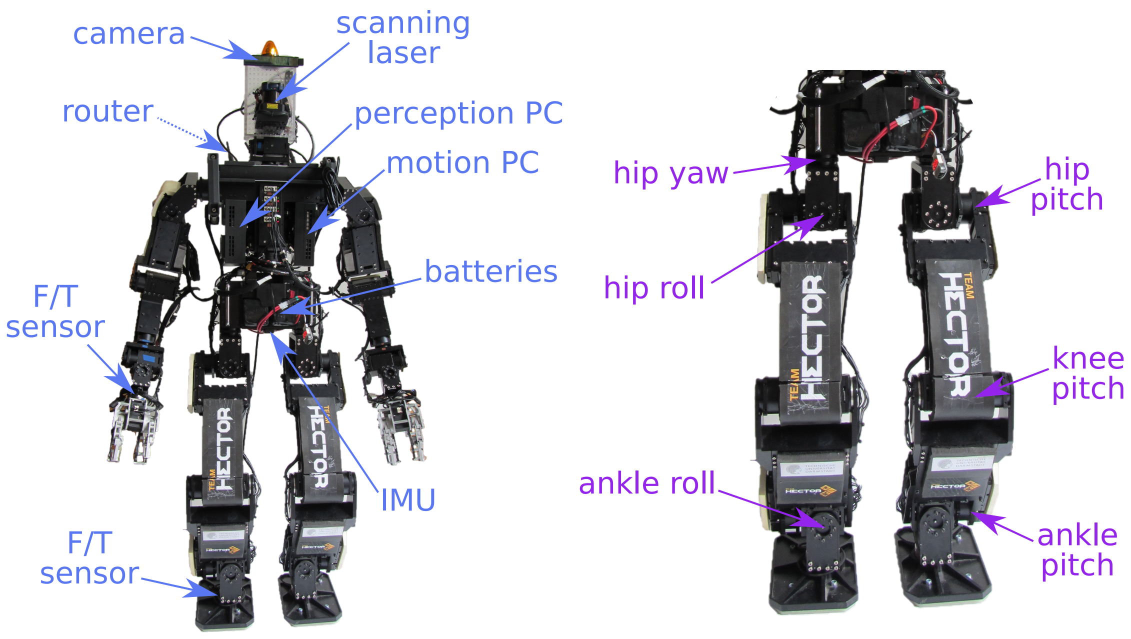


Figure 1.2: Hardware of Johnny #5

1.2.2 Existing Software

The software of Johnny #5 runs on Ubuntu 16.04 system with ROS (Robot Operating System)[1]. The robot is simulated in Gazebo and its perception of the environment can be visualized using RViz[2]. The software is developed by the SIM lab and parts of it are provided by Robotis, especially the code for the robot to walk on flat ground, however without any detailed information on how it works. Although this code takes as input a plan of 3D footsteps, the robot could not walk on uneven terrain.

The original goal of the thesis of Reimold [5] was to use the locomotion tools of the library "Drake" (instead of Robotis code) in order to improve the walking of Johnny #5. This library is developed by the Computer Science and Artificial Intelligence Lab (CSAIL) of the MIT[3] and tested by them on Atlas, the humanoid robot platform by Boston Dynamics. Not only the unsuccessful results on Johnny #5 back then but also other reasons described in Appendix A led to the decision not to continue in this direction. This is mainly because Atlas is torque-controlled, while THORMANG3 is position-controlled at joint level. Reimold then used Missura's Capture Step Framework [6] instead of Drake. Since it was developed for robots walking on flat soccer fields, Johnny #5 could still not walk on uneven ground.

After ruling out these two approaches the first stage of this thesis was investigating how Robotis code works and searching for state of the art methods for robust locomotion of position-controlled robots. It turned out that Robotis implemented the well spread method of ZMP[4]-Preview Control which is described in Section 2.2. Plenty of papers were written on extensions of this method for locomotion on uneven terrain. For these reasons this thesis focuses on improving the code written by Robotis.

1.3 Thesis Outline

Fundamentals of bipedal walking are explained in Section 2 along with state of the art walking methods based on ZMP control. Section 3 and Section 4 respectively present the two main parts of the walking software:

- the Gait Pattern Generator (GPG) which generates stable desired trajectories (feedforward)
- the Balance Control (BC) whose aim is to make the robot follow these trajectories to keep balance (feedback control)

As a parallel project the complete walking software is reimplemented as a library to improve code readability, ease further developments and enable the use on other robots. This new structure is described in Section 5. After developing the software its parameters have to be tuned for the considered robot (Johnny #5) as explained in Section 6. Section 7 presents the results of this thesis through walking experiments and finally Section 8 concludes with a sum up of the achievements and suggestions for further improvements.

[1] `http://wiki.ros.org`.
[2] `http://wiki.ros.org/rviz`
[3] Massachusetts Institute of Technology
[4] Zero-Moment Point (cf. Section 2.1 for a definition)

2 State of the Art of ZMP-based Control Concepts

In order to enable a robot to walk, reference trajectories are generated and then control theory is used to make the robot follow these trajectories [7]. The main goal by the generation and tracking of these trajectories is the stability of the gait.

In a torque-controlled robot the torque commands for the actuators of the joints can be optimized based on planned footsteps and a whole-body model, which can be expressed as the matrix equation

$$\boldsymbol{M}(\boldsymbol{q}) \cdot \ddot{\boldsymbol{q}} + \boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \boldsymbol{S}^T \boldsymbol{\tau} + \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{\lambda} \tag{2.1}$$

where vector $\boldsymbol{q}$ contains the positions of the robot base and of the joints, $\boldsymbol{\tau}$ the efforts in the joints and $\boldsymbol{\lambda}$ the contact forces. $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ the matrix of the Coriolis and gravitational terms, $\boldsymbol{S}$ the joint selection matrix and $\boldsymbol{J}$ the contact Jacobian [8, 9].

On the other hand for a position-controlled robot a simplified model is usually used, which does not consider the torques in the joints, since they cannot be directly controlled. Often, the model generates trajectories of the CoM and the feet, which are then converted into joint trajectories using Inverse Kinematics (IK). In order to generate these trajectories a dynamic stability criterion is required, whereby the Zero-Moment Point (ZMP) is commonly used. This criterion is presented in Section 2.1 in detail. Many methods have been investigated, however Section 2.2 focuses on the well spread method of Kajita et al. [10]: the ZMP Preview Control (ZMP-PC) which is based on the well-known Linear Inverted Pendulum Model (LIPM). This method was already implemented by Robotis for THORMANG3, however some improvements are required to enable the robot to walk on a complex terrain. Section 2.3 and Section 2.4 present state-of-the-art extensions (at the GPG and the BC level, respectively) to improve the robustness of the locomotion on uneven terrain with or without prior knowledge about it. In our case the terrain is known thanks to the sensors of the head, but not perfectly (an error in the centimeter range is assumed), so methods without this knowledge can be a great reference to deal with such discrepancies and improve the robustness of the walking.

2.1 Fundamental Definitions for Bipedal Locomotion using the ZMP

Commonly, a frame is placed at a fixed position in the humanoid robot upper body with the x axis pointing forwards and the z axis upwards. The locomotion is then described in the three following planes of the 3D space [11]:

- the frontal plane, corresponding to the y-z plane
- the sagittal plane, corresponding to the x-z plane
- the transverse plane, corresponding to the x-y plane

which can be seen in Figure 2.2 and Figure 2.3. During a walking gait the robot alternates between Single-Support Phases (SSPs) where only one foot is touching the ground and Double-Support Phases (DSPs) where both feet are on the ground [12]. The enclosing convexe region of the contact points between the feet and the terrain is named the support polygon and is illustrated in Figure 2.2.

As mentioned above and illustrated in Figure 2.1, walking control algorithms are classically implemented in two blocks:

- a Gait Pattern Generator (GPG) that generates ideal reference trajectories
- a Balance Control (BC) that makes the robot keep its balance and follow these trajectories.

The first block takes different but similar names in the literature. Here, the same name as in [13] is used.

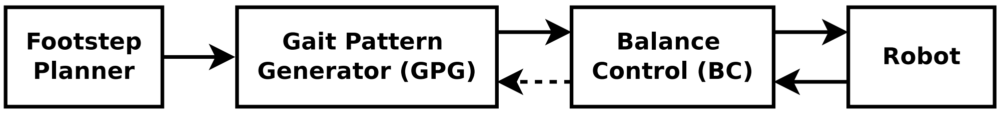


Figure 2.1: Classical structure of walking algorithms

The input of the GPG is a queue of the steps represented by parameters such as foot positions, step duration and upper body motion. Its outputs are generated trajectories of the feet, the ZMP and the upper body. The BC modifies these trajectories according to the feedback of the sensors. In some papers like [14, 15] the BC also gives feedback about the actual state of the robot to the GPG to improve the trajectories for longterm stability, while the BC deals with instant stability [7].

When the robot is standing, a good stability criterion is the projection of the Center of Mass (CoM) on the ground. If it is situated inside the support polygon, the robot is in a stable standing pose. Otherwise, the robot is falling over. This criterion can however not be used when the robot is walking, since the CoM can be located outside the support polygon during a stable walk.

Therefore in practice, the ZMP is used as dynamic stability criterion [16]. Considering the stance foot during a SSP, the total horizontal forces of the ground reaction as well as the vertical moment are compensated by the friction. The horizontal moments can only be compensated by the position of the total reaction force inside the support polygon, i.e. by the position of the Center of Pressure (CoP). If they cannot be compensated, the CoP reaches one edge of the support polygon, the horizontal torques are not zero any more at the CoP and the robot tips over. Thus, a necessary and sufficient criterion for the robot to keep balance, is that the horizontal moments at the CoP are null (cf. detailed proof in [16]). This point is named the Zero-Moment Point (ZMP). When the robot is standing still, this criterion is equivalent to the projected CoM, since it corresponds to the CoP. The goal of the BC is then to keep the ZMP inside the support polygon within safety margins to its edges.

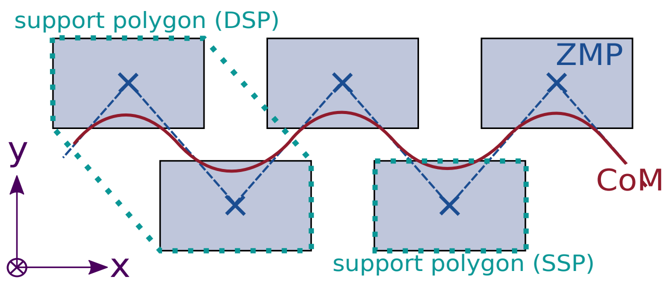
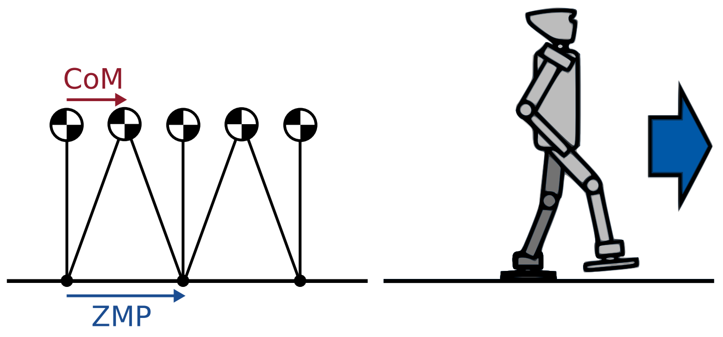
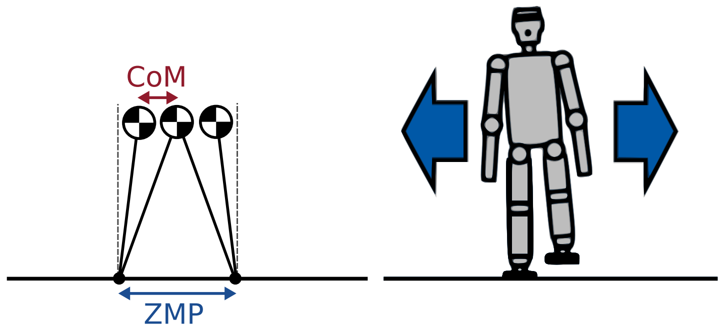


Figure 2.2: Example of a standard footstep plan with desired CoM and ZMP trajectories in the transverse plane

In the GPG, a reference trajectory of the CoM is derived from a reference trajectory of the ZMP. In Figure 2.2 a typical trajectory of the CoM of a humanoid robot during a walking gait is illustrated in the transverse plane. In this example, the ZMP reference is situated at the middle of the desired foot position with a maximal margin to the edges of the support polygon. In order to generate the CoM trajectory, a commonly used model is the Linear Inverted Pendulum Model (LIPM). The two ends of the pendulum corresponds to the ZMP and the CoM as illustrated in Figure 2.3. Based on this model, Kajita et al. [10] proposed the method of ZMP Preview Control (ZMP-PC) which is explained in the next section. It can be categorized into a more general concept called Model Predictive Control (MPC) [7].



(a) Sagittal plane



(b) Frontal plane

Figure 2.3: 3D-Linear Inverted Pendulum Model (LIPM) [image credits Reimold [5]]

2.2 ZMP-Preview Control

The ZMP Preview Control (ZMP-PC) method was first proposed by Kajita et al. [10]. It is based on the equations of the Linear Inverted Pendulum Model (LIPM) which describe the CoM and ZMP relative motion in the transverse plane:

$$\begin{cases} p_x = x - \frac{z-p_z}{\ddot{z}+g} \cdot \ddot{x} \\ p_y = y - \frac{z-p_z}{\ddot{z}+g} \cdot \ddot{y} \end{cases} \tag{2.2}$$

where (p_x, p_y, p_z) are the coordinates of the ZMP, (x, y, z) the coordinates of the CoM and g the gravity acceleration. Assuming the terrain is horizontal and flat and the CoM is moving at a constant height H above the ground ($p_z = 0$ and $z = H$), these equations become linear:

$$\begin{cases} p_x = x - \frac{H}{g} \cdot \ddot{x} \\ p_y = y - \frac{H}{g} \cdot \ddot{y} \end{cases} \tag{2.3}$$

These linear equations are equivalent to a cart-table model in each (decoupled) direction as illustrated in Figure 2.4. Indeed, the equation in x direction of this model is

$$\tau_{zmp} = mg(x - p_x) - m\ddot{x}H = 0, \tag{2.4}$$

where τ_{zmp} is the torque at the ZMP in the y direction. This leads to the first equation of the system (2.3). The same equation applies in the y direction. Thus, only the equations in the x direction will be considered in the rest of this section. From equations (2.3), the state equations of the cart-table model can be derived as

$$\begin{cases} \frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot u_x \\ p_x = \begin{bmatrix} 1 & 0 & -\frac{H}{g} \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \end{cases} \tag{2.5}$$

For a numerical implementation the system has to be discretized with a sample period T_s. Thus, the equations become

$$\begin{cases} \boldsymbol{x_{k+1}} = \boldsymbol{A} \cdot \boldsymbol{x}_k + \boldsymbol{B} \cdot u_{x,k} \\ p_{x,k+1} = \boldsymbol{C} \cdot \boldsymbol{x}_k \end{cases} \tag{2.6}$$

with

$$\boldsymbol{x_k} = \begin{bmatrix} x(kT_s) \\ \dot{x}(kT_s) \\ \ddot{x}(kT_s) \end{bmatrix}, \quad u_k = u_x(kT_s), \quad p_{x,k} = p_x(kT_s),$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \frac{T_s^3}{6} \\ \frac{T_s^2}{2} \\ T_s \end{bmatrix} \quad \text{and} \quad \boldsymbol{C} = \begin{bmatrix} 1 & 0 & -\frac{H}{g} \end{bmatrix}. \tag{2.7}$$

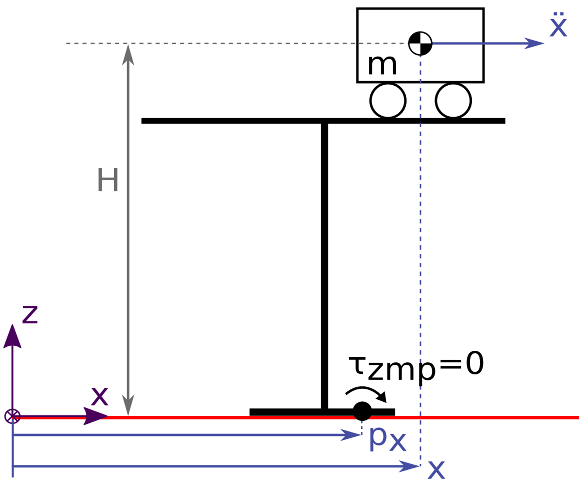


Figure 2.4: The cart-table model in 2D

In this system of equations the ZMP position is the output and the CoM state is the state vector. However, in the GPG the reference trajectory of the ZMP is determined by the desired feet positions from the Footstep Planner and a CoM trajectory is meant to be generated from this ZMP reference. Therefore, Kajita et al. [10] proposed to add a ZMP Tracking Control: A CoM position is generated by controlling the error between the ZMP reference and the ZMP position generated by the state equations. The controller was designed based on the following observation: When taking a step, the CoM is supposed to move forwards before the swinging foot touches the ground, i.e. before the next desired ZMP position is reached (cf. Figure 2.7). Therefore, the controller should take the future ZMP reference positions into account. This method was named ZMP Preview Control (ZMP-PC) because of this particularity and is illustrated in Figure 2.5. It allows the CoM to have a smooth trajectory between the feet even if the ZMP reference is discontinuous.

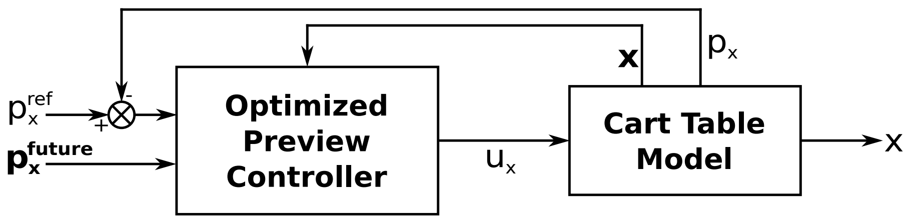


Figure 2.5: Block scheme of the ZMP Preview Control

The input u_x of the cart-table model is determined using a linear quadratic optimization with the performance index

$$J = \sum_{i=k}^{\infty} Q_e e_{x,i}^2 + \boldsymbol{\Delta x_i}^T \boldsymbol{Q_x} \boldsymbol{\Delta x_i} + R \Delta u_{x,i}^2 \tag{2.8}$$

where

$$e_{x,i} = p_{x,i} - p_{x,i}^{ref} \tag{2.9}$$

$$\boldsymbol{\Delta x_i} = \boldsymbol{x_i} - \boldsymbol{x_{i-1}} \tag{2.10}$$

$$\Delta u_{x,i} = u_{x,i} - u_{x,i-1} \tag{2.11}$$

The optimal input minimizing this performance index is

$$u_{x,k} = \underbrace{-G_i \sum_{j=0}^{k} e_{x,j}}_{\substack{\text{integral action}\\ \text{on tracking error}}} \quad \underbrace{-\boldsymbol{G_x x_k}}_{\text{state feedback}} \quad \underbrace{-\sum_{j=1}^{N} G_p(j) p_{x,k+j}^{ref}}_{\text{preview action}} \tag{2.12}$$

The gains G_i, $\boldsymbol{G_x}$ and $G_p(j)$ are derived from the parameters Q_e, $\boldsymbol{Q_x}$, R and H. The equations to calculate these gains can be found in [17] and are based on the discrete-time algebraic Ricatti equation (DARE). $T_{preview} = NT_s$ is the preview horizon during which the ZMP reference has to be known. The preview horizon is set to 1.6 s since for a bigger time, the preview gain G_p is negligible as shown in Figure 2.6 [10].

In Figure 2.7 the CoM generated by the ZMP-PC with the same gains as in Figure 2.6 is illustrated, including the ZMP position (p_x, p_y) generated by the cart model. The ZMP reference is fixed at the middle of the planned footsteps and the objective is fulfilled since the error between the generated ZMP and the ZMP reference remains very low.

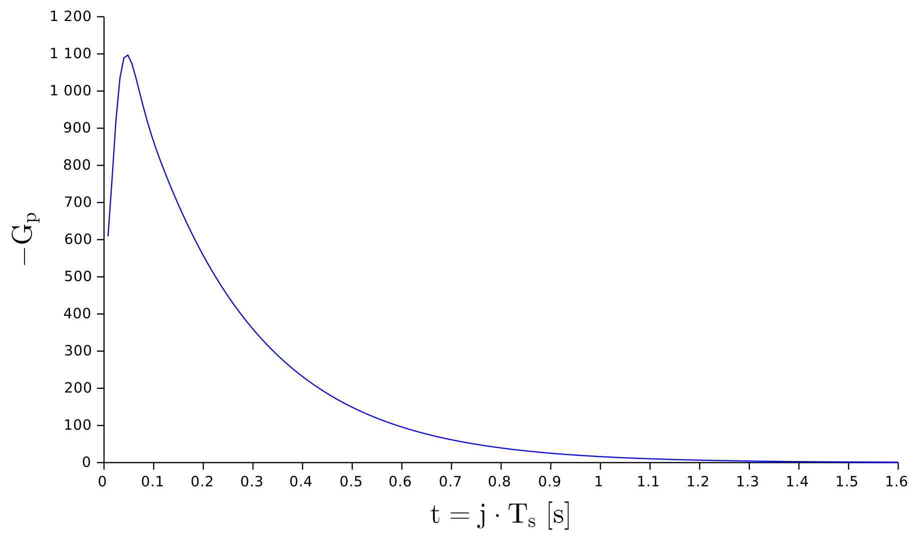


Figure 2.6: Gain G_p of the preview action for $H = 0.5\ m$, $T_s = 8\ ms$, $T_{preview} = NT = 1.6\ s$, $Q_e = 1$, $\boldsymbol{Q_x} = \boldsymbol{0}$ and $R = 1 \cdot 10^{-6}$

Wieber [18] showed that this simple model leads to very good results: The generated trajectory leads to a stable walk and the error on the generated ZMP position compared to a whole-body model is less than 2 cm. Moreover, this imprecision can be easily overcome by having safety margins of at least 2 cm inside the support polygon for the control of the ZMP.

Although other ZMP based methods exist (for instance analytical methods), the choice was made to use the ZMP-PC in this thesis because it is widely used and a lot of papers has been published on the subject, proposing extensions for improved robustness (cf. Section 2.3 and Section 2.4). Moreover, this method has already been implemented by Robotis for THORMANG3 and has already shown good results on flat ground. Another advantage of this method is that any gait is possible (i.e. the desired footsteps can be freely chosen) in contrary to methods which only allow periodic fixed gait patterns.

However, some drawbacks are that this method works on flat ground only and that it is not possible to change the ZMP reference trajectory during the preview horizon of 1.6 s [11]. These restrictions can be overcome using extensions that are presented in the next section. The removal of the second restriction can be really useful since it for instance makes it possible to overcome stronger perturbations by quickly change a desired footstep position according to the sensory feedback.

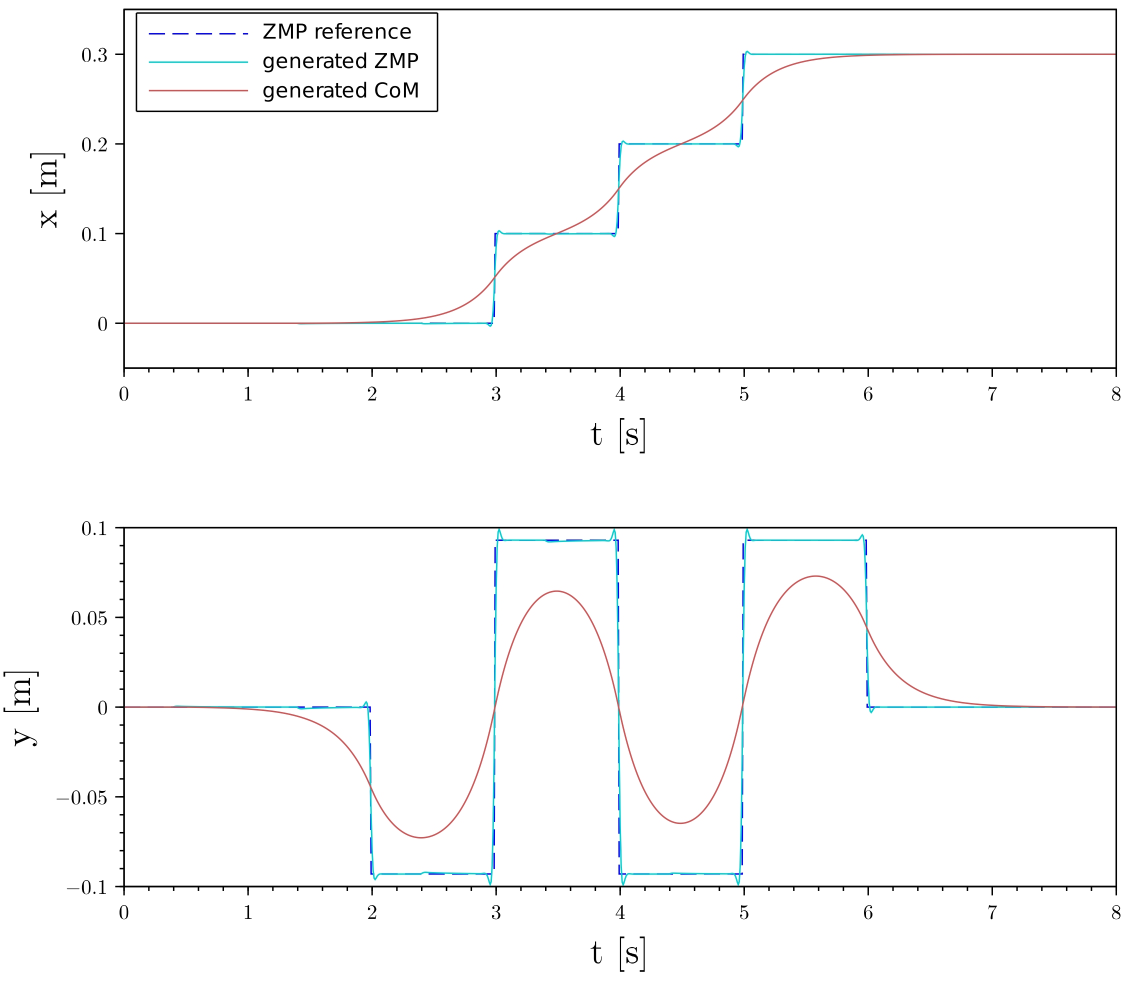


Figure 2.7: Trajectories of ZMP and CoM generated by ZMP Preview Control taking a ZMP reference trajectory as input

2.3 Extensions of the ZMP-Preview Control for Uneven Terrain

In the paper of Kajita et al. [10] presenting the ZMP Preview Control (ZMP-PC), the assumption was made that the ground is horizontal and that the CoM is moving in a parallel plane. However, the authors also tested the algorithm on stairs in the simulation but did not explain how they adapted it. Later, some researchers have proposed extensions of the method of ZMP-PC to explicitly consider the unevenness of the terrain (cf. Section 2.3.1). Moreover, some work has been made to improve the precision and robustness of the model (cf. Section 2.3.2).

2.3.1 Adapting the ZMP-Preview Control Method to Uneven Terrain

Virtual Slope method

Sato et al. [19] pointed out that the LIPM is confronted to what they named a *ZMP equation problem*. Indeed, the equations (2.2) are generally not linear on uneven terrain. However on a slope, if the CoM is moving in a plane parallel to the slope ($p_z = k \cdot p_x$ and $z = k \cdot x + H$ where k is the slope coefficient), the same linear equations (2.3) as on flat ground are obtained. On the other hand, on stairs with the CoM moving in a plane ($p_z = 0$ and $z = k \cdot x + H$) the equations remains non linear:

$$\begin{cases} p_x \frac{k\ddot{x}+g}{g} = x - \frac{H}{g} \cdot \ddot{x} \\ p_y = y - \frac{H}{g} \cdot \ddot{y} \end{cases} \tag{2.13}$$

Moreover during a DSP on stairs, the ZMP is not defined because the two feet do not lay on the same plane. The authors named it the *ZMP definition problem*. These two problems can be overcome by using the proposed method of Virtual Slope (VS).

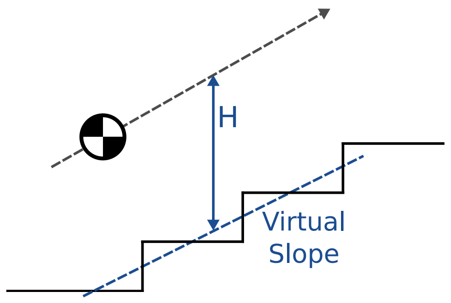


Figure 2.8: Virtual Slope method of Sato et al. [19] on stairs

As illustrated in Figure 2.8 the stairs are replaced by a VS which can be freely chosen but has to stay near the stairs. If the CoM moves in a plane parallel to it, the same linear equations as on a slope (i.e. as on flat ground) can be applied (equations (2.3)). For this VS, a virtual ZMP and a virtual support polygon are defined. In order to construct the latter, a line is drawn between each vertex of the real support polygon and the CoM. The boundaries of the virtual support polygon are made of the intersections of these lines and the virtual slope. The virtual ZMP is situated at the intersection between the virtual slope and the Zero-Moment Line (ZML). The latter is the line on which the total reaction moment is zero and is described more in detail in Section 4.2.5. Although the authors did not use ZMP-PC but an analytical polynomial solution, their VS method could be used in combination to ZMP-PC since it is based on the same equations of the LIPM.

Extended ZMP Method

Instead of looking for a method to enable the use of the linear equations (2.3) of the cart-table model for any uneven terrain, Sun et al. [20] proposed an extended cart-table model for solving the *ZMP equation problem* and an Extended Zero-Moment Point (EZMP) for solving the *ZMP definition problem*. In this extended model a vertical motion of the CoM is enabled by adding a third preview controller in the z direction based on the EZMP instead of the ZMP. The EZMP is the point of the Virtual Contact Plane (VCP) where the ground reaction moment is orthogonal to the VCP. The VCP is defined as the plane passing through the two feet positions in the actual DSP and the position of the next planned footstep.

For equations (2.2) of the LIPM to be linear the fraction before the CoM acceleration has to be constant:

$$\frac{z - p_z}{\ddot{z} + g} = c \tag{2.14}$$

This leads to the following linear system

$$\begin{cases} p_x = x - c\ddot{x} \\ p_y = y - c\ddot{y} \\ p_z^{aux} = p_z + cg = z - c\ddot{z} \end{cases} \tag{2.15}$$

where $\boldsymbol{p^{aux}} = (p_x, p_y, p_z^{aux})^T$ is called auxiliary EZMP and c is a parameter that is set to an arbitrary constant. Based on these equations, the authors implemented a “triple preview control”. The desired EZMP is fixed in the middle of the stance foot during SSP and interpolated linearly during DSP. The virtual support polygon is simply defined as the orthogonal projection of the real support polygon on the VCP.

Other Similar Methods

Shimmyo et al. [21, 22] used the method of the VS of Sato et al. on flat ground to enable a vertical motion of the CoM to obtain a more natural gait. They constructed a virtual plane with a sinusoidal profile in the sagittal plane and made the CoM moves parallel to this plane (i.e. at a constant height about it). The ZMP-PC is computed based on the virtual ZMP on this virtual plane.

Huang et al. [23] also proved that the equations of the LIPM are linear on a slope when the CoM is parallel to it by considering the LIPM equations in the frame of the slope. Unlike most of the papers which do not mention it at all, they especially considered the transition between the flat ground and the slope during the DSP and explained how to set the parameters to have a continuous transition. One disadvantage of their method is that they considered a fixed step distance for the whole gait. Moreover, the height H of the CoM above the slope cannot be freely chosen.

2.3.2 Improving the Robustness of the ZMP-Preview Control Method

Reducing the Modeling Errors

Some authors like Nishiwaki et al. [24] or Shimmyo et al. [25] decided to compute the ZMP-PC twice to reduce the errors due to the approximation of the LIPM. The first ZMP-PC is used to compute the CoM trajectory, the second one to compute an additional CoM compensation term taking a ZMP error as input. This error corresponds to the difference between the desired ZMP and the ZMP calculated using the Multi-Mass Model of the robot, whose input is the CoM trajectory calculated by the first ZMP-PC. A disadvantage of this method is that the preview horizon is twice as long. This means that the period during which the trajectory of the reference ZMP cannot be changed is twice as long as for the classical method of ZMP-PC.

Alternatively, some authors proposed to use a Three-Mass Model (instead of the LIPM) to reduce the ZMP error [13, 25]. The three masses correspond to the upper body and the two legs. As a drawback, since their heights are supposed constant, the swinging foot should not be raised too high during the gait.

Taking the Estimated Actual State of the Center of Mass into Account

In order to improve the robustness of the walking, some authors proposed to adapt the ZMP-PC such that the estimated actual state of the CoM is considered in the generation of desired CoM trajectory (as feedback correction [14] or initial condition [15]). In the quoted papers they achieved a stable walking under strong pushes on the side or on unknown slopes, respectively.

Other MPC Methods

Some authors proposed other MPC methods based on the same equations as the ZMP-PC to improve the performances of the generated trajectories. For instance, Wieber [14] pointed out that the limits of the support polygon are not taken into account in the ZMP-PC method. Hence, he proposed a Quadratic Program based on the same equations but with additional inequality constraints on the position of the ZMP. This leads the desired ZMP to have no fix position inside the sole.

2.4 Related Work on the Balance Control

In this section state-of-the-art control loops are presented that deal with making a position controlled humanoid robot keep its balance when walking.

Upper Body and Feet Orientation Control

In order to keep the upper body of the robot vertical, classical controllers are generally implemented (such as PD [26] or PI [27] controllers) which take the inclination of the upper body measured by an IMU as input and return an angle correction for specific joints that have an influence on this inclination (for instance the hip roll/pitch joints [26] or the ankle roll/pitch joints [27]).

In order to adapt the foot orientation to an uneven terrain, an adaptation control is often implemented, which takes the torques measured in the feet into account to compute an orientation correction (for instance as a low pass filter [27] or as a damping controller [28]).

Landing Control

The aim of a landing control is to reduce the impact of the swinging foot on the ground and maintain the contact with the ground after the landing. The input is the vertical force error and the output a correction of the foot height. In the literature different kind of controllers can be found: P controllers [26], damping controllers [15, 28] or controllers based on a mass-spring-damper system [19, 25, 27].

ZMP Control

Different methods have been proposed to control the position of the real ZMP while the robot is walking. The error between the ZMP reference and the estimated position of the real ZMP is used to compute a correction of the horizontal position [26, 27] or velocity [24] or acceleration [20] of the CoM using the same different kinds of controllers as for landing control.

Changing the ZMP Reference inside the Foot Sole

Instead of correcting the CoM state using a ZMP controller, other authors proposed to adapt the ZMP reference trajectory. In the original ZMP-PC method it is not possible to change the ZMP reference during preview horizon. In order to overcome this disadvantage, Kajita et al. [29] developed the concept of "auxiliary ZMP" which corresponds to the difference between the new desired ZMP and the planned ZMP reference. The input of the cart-table model is the sum of the output of the ZMP-PC controller based on the ZMP reference and of a correction term based on the auxiliary ZMP. The authors give an example of auxiliary ZMP which is composed of a PD controller on the upper body inclination and a P controller on the real ZMP error. This new method can deal with external forces or uneven terrain. This concept was taken over and extended by Nishiwaki et al. [30, 31]. Their robot was able to walk up on an unknown slope.

Modification of the Planned Footsteps

When dealing with very strong perturbations, the only solution to keep balance is to change the position of the next planned footsteps [11]. These new footsteps are commonly named Capture Steps. As already mentioned in Section 1.2.2, Missura [6] implemented a framework based on this concept for soccer robots and it was used on Johnny #5 in a former thesis [5]. In [32] the Quadratic Program of Wieber [14] was enhanced with additional parameters and constraints on the positions of the footsteps occurring during the preview horizon. The paper gives examples of simple conditions to avoid collisions, over-stretched legs and too high joint speeds. Nishiwaki et al. [30] also developed a method to change the next footstep position while using ZMP-PC. When it is not necessary to change the footstep position, they alternatively proposed to modify the duration of the current step.

Parameters Tuning

Most of the authors do not describe in details how they tuned the parameters of their software. Some use local models as in [28], but most of them certainly tune them manually by "trial and error" (for instance [27]). Alternatively, Fu et al. [26] proposed to determine the controller gains using Reinforcement Learning. However, the other parameters of the software were apparently set manually and the robot was walking very slowly (with a step duration of 6 s).

3 Design of the Gait Pattern Generator

The main goal of the GPG is to generate desired CoM and feet trajectories. As explained in Section 3.1, the CoM is actually approximated using the Center of Body (CoB). The reference frames used for the generation of the trajectories are described in Section 3.3. For the feet, Robotis implemented Swing Foot Trajectories that do not take obstacles or 3D terrain into account, so it was not possible for the robot to walk up stairs for instance. In this thesis, the existing Swing Foot Trajectory implementation is improved to add these features. The old and new methods are described in Section 3.4. Moreover in Robotis code, the CoB was moving in a horizontal plane using ZMP-PC. In the new implementation, a CoB vertical motion is enabled using an own Virtual Slope method based on Sato et al. work [19] (cf. Section 2.3.1). The generation of the former and new CoB trajectories is described in Section 3.5.

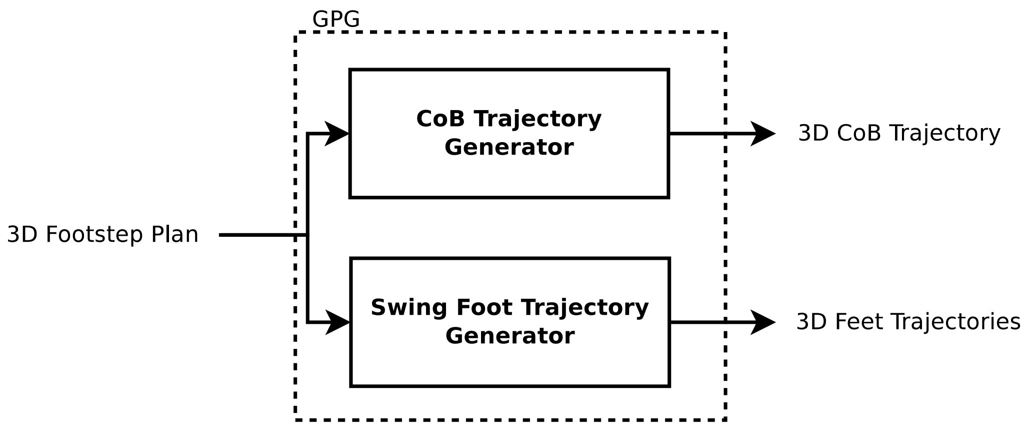


Figure 3.1: Internal structure of the GPG

3.1 Center of Body

In the ZMP-PC method of Kajita et al. [10] the CoM is assumed to be a fixed point relative to the upper body of the robot. This drastically simplifies the computation of the corresponding desired trajectories in joint space.

For THORMANG3, Robotis defined the Center of Body (CoB) as the middle point between the hip joints (on the hip pitch axis) as illustrated in Figure 3.2. The CoM is then approximated by the CoB position plus a constant offset in each spatial direction. In Section 6.3.2 and Section 6.3.3 it is explained how the horizontal and vertical offsets were determined. This assumption of a fix CoM seems acceptable, since only the legs are moving relatively to the upper body (the arms are immobile).

3.2 Initial Pose

Before a walking experiment, the robot is brought in a predefined initial pose illustrated in Figure 3.2, so the CoB is situated at $H_{cob} = 0.630\ m$ above the ground. This way the knees are bent enough to allow the robot to walk. In this pose the ankle roll joints are vertically aligned with the hip roll joints ($d_{legs} = 0.186\ m$).

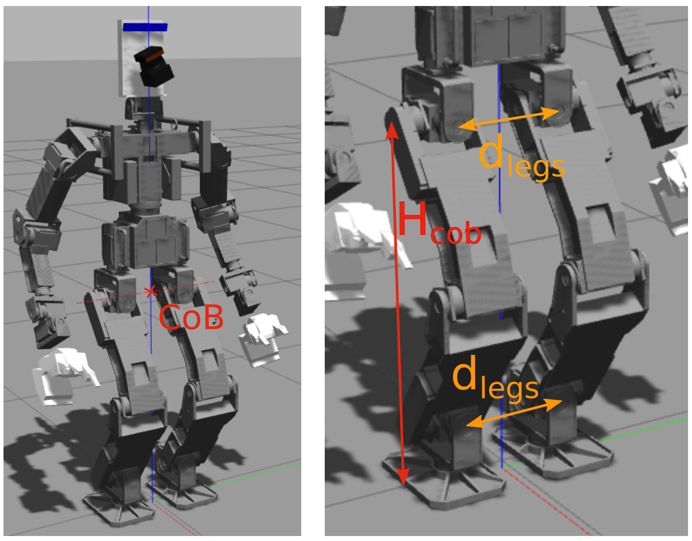


Figure 3.2: Initial pose and CoB position of THORMANG3 in Gazebo

3.3 Reference Frames

In this section, the reference frames for the generation of the desired trajectories of the CoB and the feet are described. These frames are purely mathematically constructed and correspond to the desired gait. They are neither related to the real motion of the robot (odometry) nor to the robot perception of the real world (cartography).

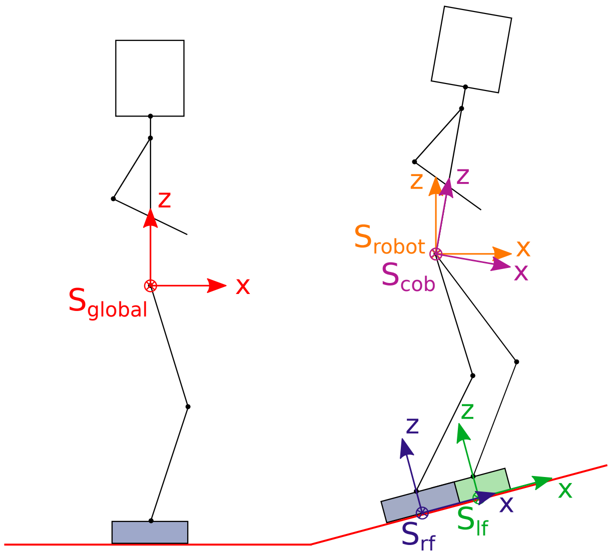


Figure 3.3: Reference frames of the GPG

The global frame S_{global} has its origin at the (desired) CoB position at the beginning of the motion (i.e. in the initial pose). The z axis is vertical and the x axis points in the walking direction as illustrated in Figure 3.3. Each time a new walking motion is started, S_{global} is reinitialized to the new starting position of the CoB.

The CoB frame S_{CoB} corresponds to the CoB (desired) position and orientation during the motion, while the robot frame S_{robot} corresponds to the CoB (desired) position and its yaw rotation only, i.e. its z axis remains vertical.

The origins of the right and left foot frames S_{rf} and S_{lf} are situated at the (desired) projection of the middle of the ankles (i.e. the intersection of the ankle roll and pitch joints) on the sole (or equivalently the projection of the middle of the F/T sensors). It does not lay in the middle of the sole as illustrated in Figure 3.4. The orientations of these frames correspond to the (desired) orientations of the feet during the motion, with the x axis pointing toward the foot front and the z axis pointing upwards (orthogonal to the sole).

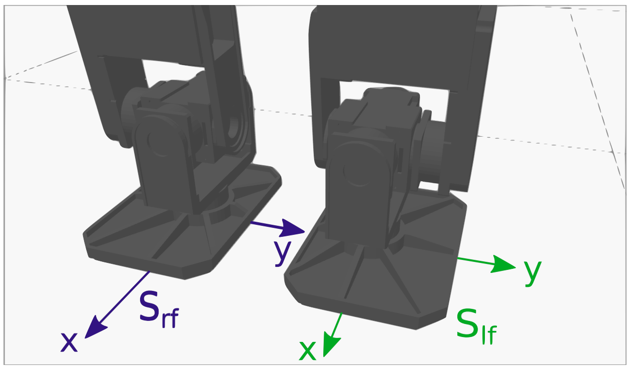


Figure 3.4: Reference frames of the feet

In the GPG the desired trajectories of the CoB and the feet (i.e of the frames S_{cob}, S_{rf} and S_{lf}) are generated relative to the frame S_{global} and then transformed into S_{robot} for the BC (cf. Section 4). The rotation of a frame relative to the global frame orientation is described using the roll-pitch-yaw convention (Euler ZYX angles) with the classical notation:

$$\begin{aligned} roll &\longleftrightarrow \phi_i \\ pitch &\longleftrightarrow \theta_i \\ yaw &\longleftrightarrow \psi_i \end{aligned}$$

where $i \in \{robot, cob, rf, lf\}$.

3.4 Swing Foot Trajectory Generator

The foot position is represented by the 3D pose of the foot frame relative to the global frame (cf. 3.3). The 3D Swing Foot Trajectory is generated by constructing C^2-smooth trajectories of the x, y, z, roll, pitch and yaw components of the foot pose. In the original code, the smoothness has been satisfied using a fifth polynomial function (cf. 3.4.1). In the new code, some additional requirements have been taken into account: smooth vertical landing, control of maximal altitude of foot and stepping over uneven terrain or obstacles without collision. These improvements are presented in 3.4.2. In order to demonstrate and compare the properties of the Swing Foot Trajectories, two case studies are considered: walking on flat ground and walking up stairs as illustrated in Figure 3.5. In both cases a straight forward swinging motion of the right foot is considered.

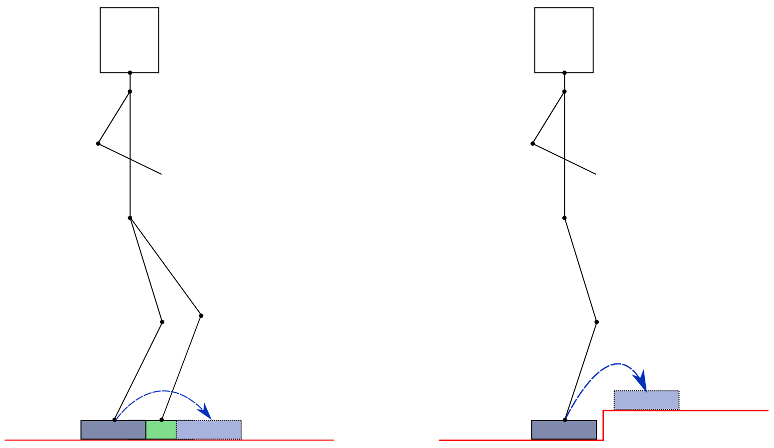

(a) step on flat ground **(b)** step on stairs

Figure 3.5: Case studies for the Swing Foot Trajectory

3.4.1 Robotis' Swing Foot Trajectory

The original code makes use of a fifth polynomial function

$$x(t) = a_0 \cdot t^5 + a_1 \cdot t^4 + a_2 \cdot t^3 + a_3 \cdot t^2 + a_4 \cdot t + a_5, \quad t \in [t_0, t_1] \tag{3.1}$$

with the six boundary conditions

$$\begin{gathered} x(t_0) = x_0, \quad \dot{x}(t_0) = \dot{x}_0, \quad \ddot{x}(t_0) = \ddot{x}_0, \\ x(t_1) = x_1, \quad \dot{x}(t_1) = \dot{x}_1 \quad \text{and} \quad \ddot{x}(t_1) = \ddot{x}_1. \end{gathered} \tag{3.2}$$

The six coefficients are computed by solving the resulting system of equations.

This function is used to generate the trajectory of each coordinate x, y and z of the frame of the swinging foot between the starting position at time t_{start} and the goal position at time t_{end} given by the footstep planner (in global frame). To ensure the continuity in velocity and acceleration, the boundary conditions of the first and second derivatives are set to zero. The resulting x, y and z ground trajectories on flat ground are illustrated in Figure 3.6. The roll, pitch and yaw trajectories of the foot are constructed the same way.

Since the velocity and acceleration boundary conditions are null, it can be mathematically proven that these fifth polynomial trajectories are always monotonic (cf. Lemma 1 and its proof in Appendix B), so the foot always stays between the start and the goal position and moves towards the latter. This is required for every component of the Swing Foot Trajectory except for the z component. Indeed, the foot should "swing" above the floor.

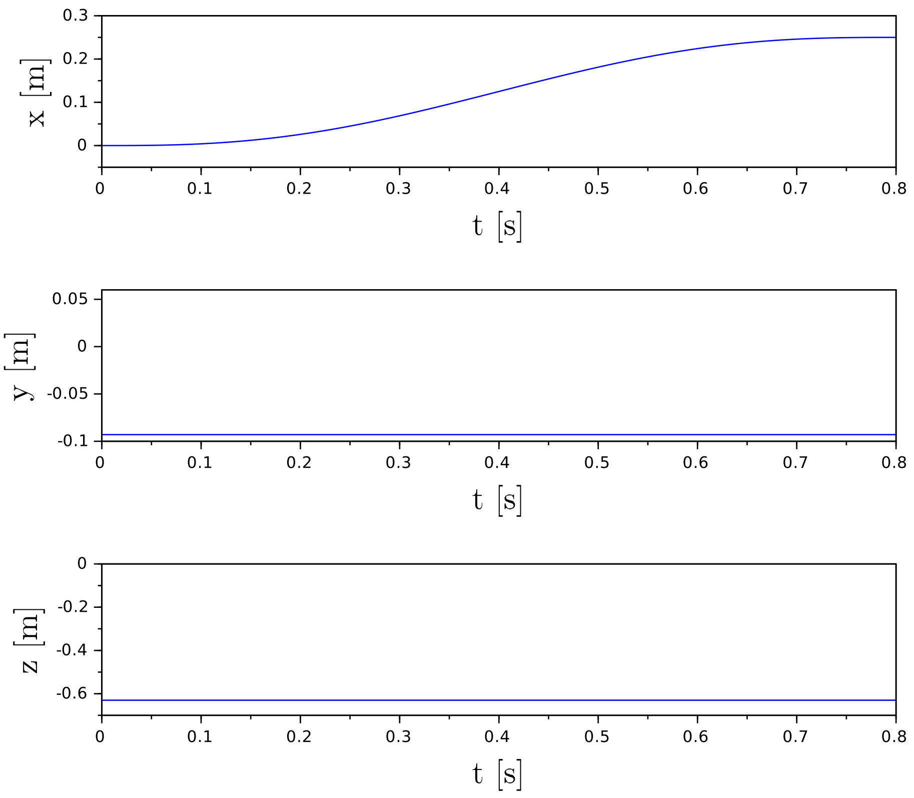


Figure 3.6: Robotis' ground Swing Foot Trajectory on flat ground (x, y and z components)

For this purpose a complementary trajectory in z-direction was added, which is composed of two fifth polynomial trajectories connected at the time t_{max} such that

$$t_{max} = \frac{t_{start} + t_{end}}{2}. \tag{3.3}$$

For the first function the boundary conditions are

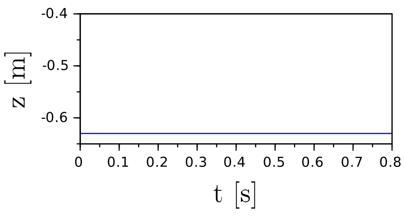
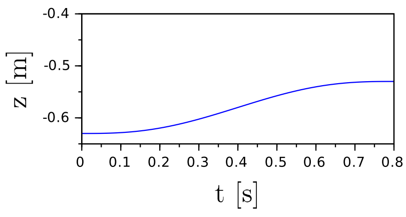
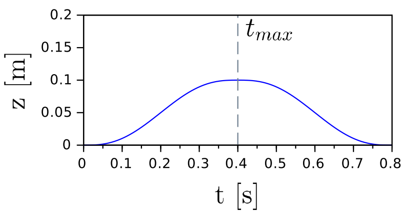
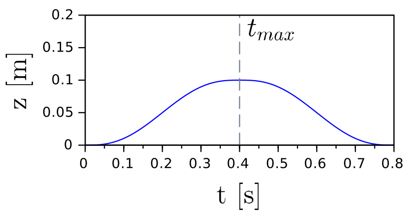
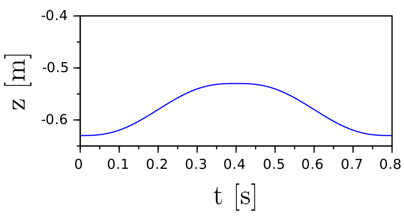
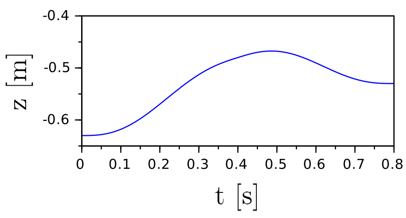
$$\begin{gathered} z(t_{start}) = 0, \quad \dot{z}(t_{start}) = 0, \quad \ddot{z}(t_{start}) = 0, \\ z(t_{max}) = \Delta Z_{foot}, \quad \dot{z}(t_{max}) = 0 \quad \text{and} \quad \ddot{z}(t_{max}) = 0 \end{gathered} \tag{3.4}$$

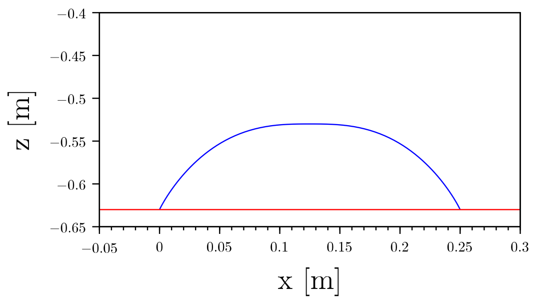
and for the second function

$$\begin{gathered} z(t_{max}) = \Delta Z_{foot}, \quad \dot{z}(t_{max}) = 0, \quad \ddot{z}(t_{max}) = 0, \\ z(t_{end}) = 0, \quad \dot{z}(t_{end}) = 0 \quad \text{and} \quad \ddot{z}(t_{end}) = 0. \end{gathered} \tag{3.5}$$

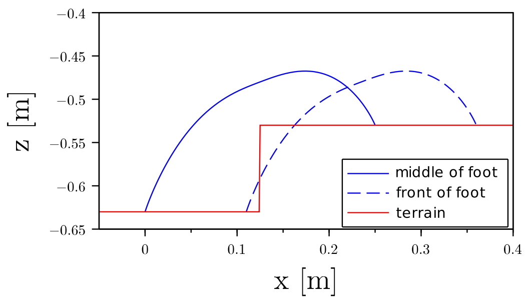
In the original code the amplitude ΔZ_{foot} of the swinging motion is set to 0.1 m. By adding this "swing complement" to the ground trajectory, the continuous trajectory in z direction illustrated in Table 3.1 is obtained. Figure 3.7 shows the resulting 3D swing foot trajectory in the x-z plane.

Table 3.1: Construction stages of Robotis' Swing Foot Trajectory in z direction

	flat ground	stairs of 10 cm
ground trajectory		
+		
swing complement		
=		
final trajectory		



(a) flat ground



(b) stairs of 10 cm

Figure 3.7: Robotis' Swing Foot Trajectory in x-z plane

Limits

The amplitude of the swing complement ΔZ_{foot} is a constant parameter which is set manually in the original code, i.e. it does not depend on the height of the stair. The higher the stairs are, the nearer to the stairs the foot is swinging and the later the maximum of the trajectory is arriving. This is illustrated in Figure 3.8. Thus in this implementation, a specific value for the maximum of the z-trajectory and its temporal abscissa cannot be set. So it is not possible to easily adapt the Swing Foot Trajectory to an uneven terrain. In fact, the foot forefront collides with the stairs in Figure 3.7. Symmetrically, the foot is not rising enough above the stairs when stepping down.

Another limit of the original implementation is the absence of consideration of early and late landing. This can occur because of modeling errors of the robot or of the terrain. As explained by Taşkıran et al. in [27], if the foot lands earlier as planned, it will not reach the goal position in the forward direction and will push the CoB in the opposite direction, while the other foot will continue to push it forward. This can cause the robot to loose balance. Taşkıran et al. proposed to simply stop the motion of the foot in the forward direction. But this leads to an error in the foot position. Alternatively, Yi et al. [28] proposed to generate a Swing Foot Trajectory which is vertical during the landing phase, so it ensures the reaching of the desired horizontal position and only the vertical motion has to be stopped at landing detection (cf. Section 4.2). Moreover, in order to avoid a too strong impact on the floor by an early landing, Yi et al. limited the vertical velocity of the landing foot. These concepts have been considered in the design of the improved Swing Foot Trajectory (cf. Section 3.4.2).

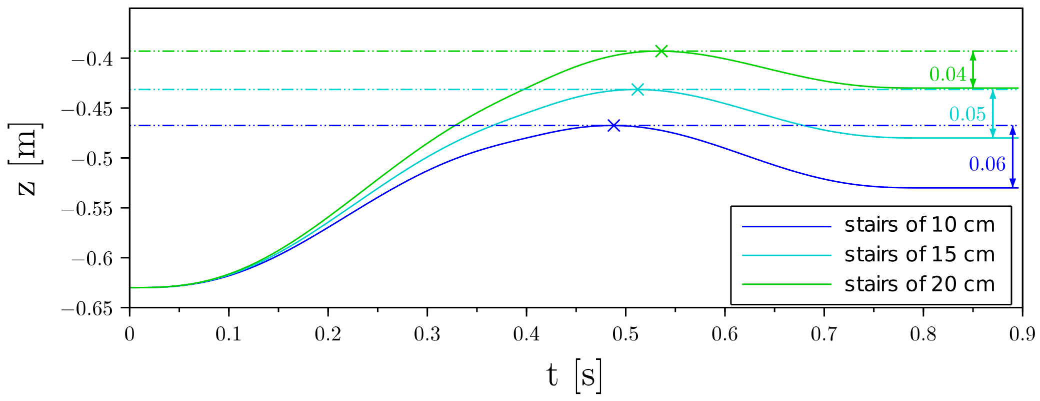


Figure 3.8: Variation of the maximum of Robotis' Swing Foot Trajectory in z direction for different stairs height

3.4.2 Improved Swing Foot Trajectory

To overcome the limits of the original trajectory (cf. Section 3.4.1), a new 3D Swing Foot Trajectory has been implemented with the following features:

- vertical landing with limited velocity
- parametrization of trajectory height
- collision avoidance using bypass points

All the constant parameters appearing in this section are listed with their values in Appendix E.

Vertical Landing

In order to achieve a vertical landing of the swing foot, obviously all horizontal components must reach their target first [28]. In addition, the angular components (roll, pitch and yaw) must line up with the terrain first, before starting to lower the swing foot. In this thesis the terrain is assumed to be known, thus the generation of such trajectories is possible. Like in Robotis' implementation, fifth polynomial functions are used for the x, y, roll, pitch and yaw trajectories. However, in this thesis they reach their target value at the approach time t_{appro} such that

$$t_{appro} = t_{end} - \Delta T_{appro}, \quad \Delta T_{appro} > 0. \tag{3.6}$$

The trajectories of the x, y and z components are plotted in Figure 3.9. The resulting trajectory in x-z plane on flat ground is very similar to the original trajectory except for the vertical landing part which is now vertical as shown in Figure 3.10.

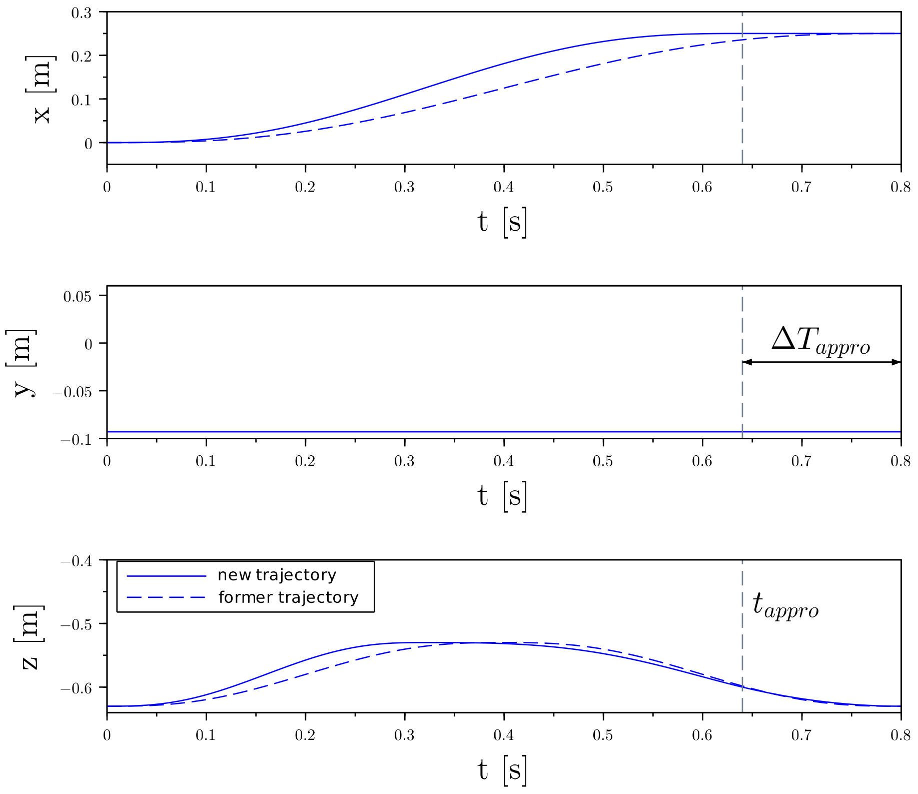


Figure 3.9: x, y and z components of new Swing Foot Trajectory with vertical landing on flat ground

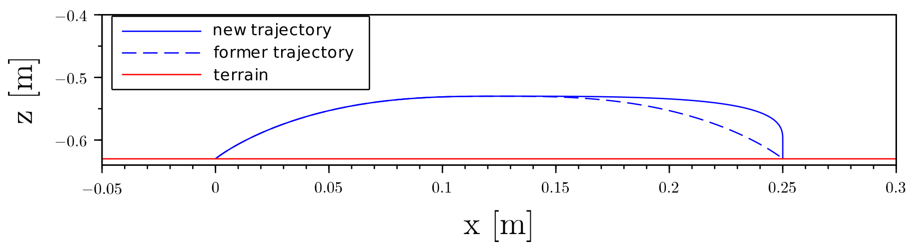
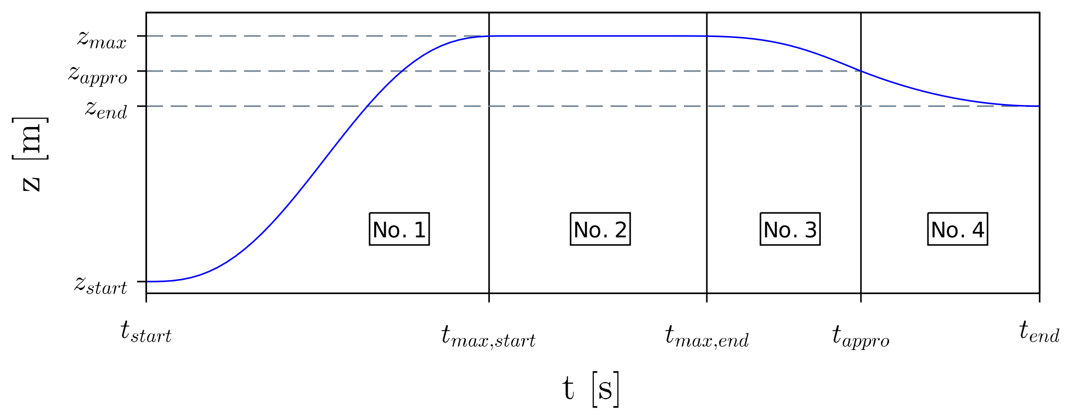
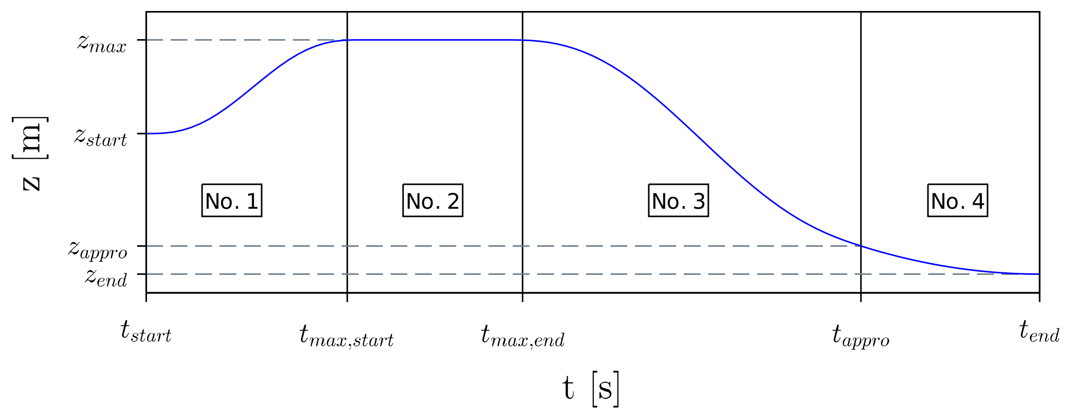


Figure 3.10: New Swing Foot Trajectory with vertical landing in x-z plane on flat ground

The new trajectory of the z component is constructed as a piecewise function whose different phases are illustrated in Figure 3.11. While Yi et al. [28] used only quadratic functions, Buschmann [11] implemented only fifth polynomial functions. In this thesis, both types are combined.



(a) step up



(b) step down

Figure 3.11: Vertical component of the new Swing Foot Trajectory on uneven ground

For the landing phase (No. 4) the quadratic function

$$z_{appro}(t) = \frac{\Delta Z_{appro}}{(\Delta T_{appro})^2} \cdot (t - t_{end})^2 + z_{end}, \quad t_{appro} \leq t \leq t_{end} \tag{3.7}$$

is implemented. The vertical approach distance ΔZ_{appro} is a positive parameter. The choice of this function was made by considering the following requirements for a smooth landing:

(i) convex polynomial function

(ii) decreasing for $t \in [t_{appro}, t_{end}]$

(iii) satisfying the four boundary conditions:

$$z(t_{appro}) = z_{appro}, \quad z(t_{end}) = z_{end}, \quad \dot{z}(t_{end}) = 0 \quad \text{and} \quad \ddot{z}(t_{end}) = 0 \tag{3.8}$$

$$\text{with} \quad z_{appro} = z_{end} + \Delta Z_{appro} \tag{3.9}$$

(iv) having the smallest possible derivative at time t_{appro}

It can be mathematically proven that all these requirements are satisfied by the implemented quadratic trajectory. Requirement (iv) is illustrated in Figure 3.12. Thus, the maximal vertical velocity V_{max} of the quadratic landing trajectory on the interval $[t_{appro}, t_{end}]$ is

$$V_{max} = \dot{z}(t_{appro}) = -\frac{2 \cdot \Delta Z_{appro}}{\Delta T_{appro}} \tag{3.10}$$

and can be adapted to the robot and the gait by tuning the approach parameters ΔZ_{appro} and ΔT_{appro}.

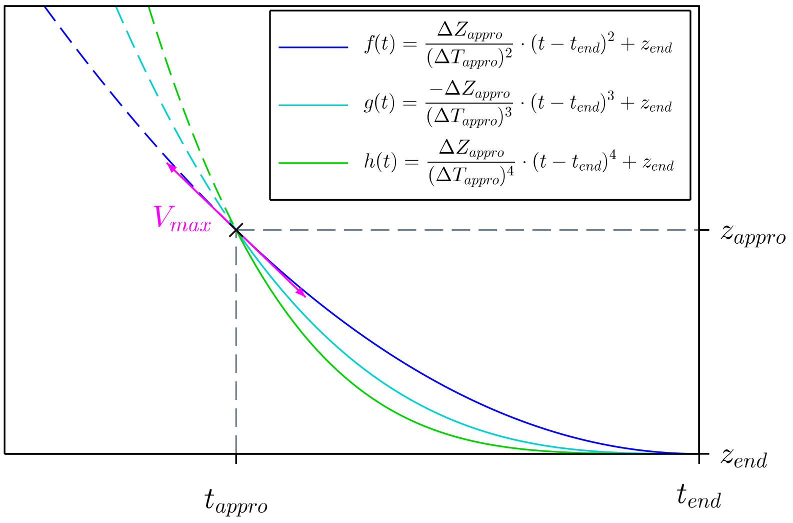


Figure 3.12: Comparison between polynomial trajectories of different orders verifying requirements (i), (ii) and (iii)

For the raising and decreasing phases of the trajectory (No. 1 and No. 3, cf. Figure 3.11), fifth polynomial functions were used in order to satisfy the boundary conditions in position, velocity and acceleration:

$$\begin{gathered} z(t_{start}) = 0, \quad \dot{z}(t_{start}) = 0, \quad \ddot{z}(t_{start}) = 0, \\ z(t_{max,start}) = z_{max}, \quad \dot{z}(t_{max,start}) = 0, \quad \ddot{z}(t_{max,start}) = 0, \\ z(t_{max,end}) = z_{max}, \quad \dot{z}(t_{max,end}) = 0, \quad \ddot{z}(t_{max,end}) = 0, \\ z(t_{appro}) = z_{appro}, \quad \dot{z}(t_{appro}) = V_{max} \quad \text{and} \quad \ddot{z}(t_{appro}) = A_{appro} \end{gathered} \tag{3.11}$$

where the approach acceleration A_{appro} is defined as

$$A_{appro} = \frac{2 \cdot \Delta Z_{appro}}{\Delta T_{appro}^2}. \tag{3.12}$$

Parametrization of Trajectory Height

The maximum of the vertical trajectory has to be limited by a parameter ΔZ_{max} in order to ensure kinematic feasibility. At the same time, the foot is supposed to raise at a sufficient height ($\geq \Delta Z_{min}$) above the ground in order to avoid undesirable contacts with the ground and to be compatible with the approach trajectory implementation. To sum up, the following constraints apply:

$$z_{max} - \min(z_{start}, z_{end}) \leq \Delta Z_{max} \tag{3.13a}$$

$$z_{max} - z_{start} \geq \Delta Z_{min} \tag{3.13b}$$

$$z_{max} - z_{end} \geq \Delta Z_{appro,min} \tag{3.13c}$$

In order to have a smooth decreasing trajectory during the phases No. 3 and No. 4, the minimal global approach distance $\Delta Z_{appro,min}$ above the goal position (i.e. not only during the vertical approach phase) is set as follows:

$$\Delta Z_{appro,min} = 2 \cdot \Delta Z_{appro}. \tag{3.14}$$

If during the trajectory generation one of the constraint is not satisfied, a ROS warn message is generated, a false boolean is returned and a null trajectory is generated. This can for instance occur if the difference between z_{start} and z_{end} is too high. For the nominal trajectory (i.e. without consideration of bypass points) the parameters are set as follows:

$$z_{max} = \max(z_{start} + \Delta Z_{min}, z_{end} + \Delta Z_{appro,min}) \tag{3.15a}$$

$$t_{max,start} = t_{max,end} = \frac{1}{2}(t_{start} + t_{appro}) \tag{3.15b}$$

Since the velocity and acceleration boundary conditions of phase No. 3 are not all equal to zero, the trajectory is not necessarily monotonic decreasing on $[t_{max,end}, t_{appro}]$. To remedy this, the parameter $t_{max,end}$ is increased until the velocity is negative on $[t_{max,end}, t_{appro}]$. Since directly checking the sign of the velocity on the whole interval is not feasible, equivalent criteria have been derived and are described in Appendix B. This method always leads to a solution (cf. Lemma 2 in Appendix B).

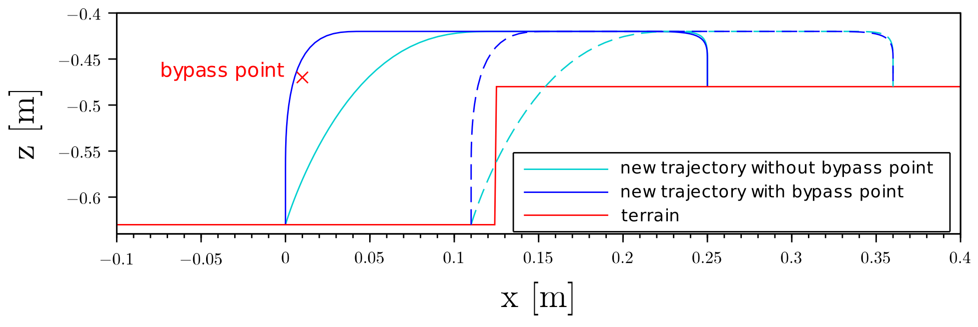
Bypass Points

In order to avoid collisions with known obstacles such as stairs, a list of bypass points can be passed as input of the Swing Foot Trajectory Generator. The bypass points are computed by the footstep planner using the knowledge of the terrain and taking safety margins around the obstacles into account, such that if the Swing Foot Trajectory (of the middle of the foot) passes above these bypass points in the x-z plane of the robot frame, the foot will not collide with the obstacle. The computation of the bypass points is not part of this thesis because it is the role of the Footstep Planner.

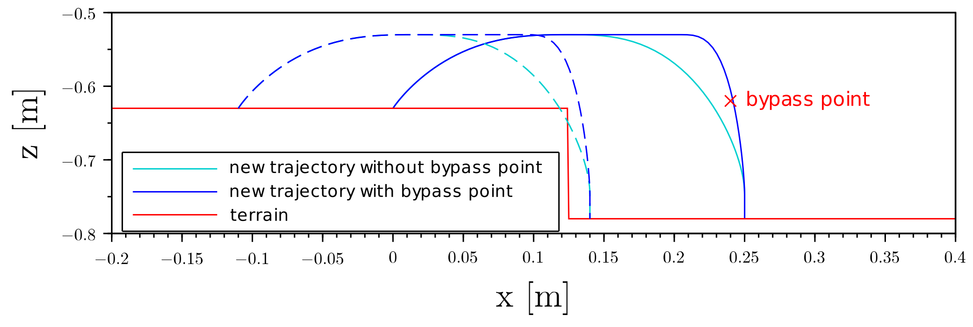
The generation of the Swing Foot Trajectory considering the given bypass points works as follows. First, the nominal trajectory explained above is generated. Then, for each bypass point the trajectory is gradually expended until it contains the considered bypass point. The detailed procedure is written as pseudo-code in Appendix C and an example of the generated trajectories on stairs is given in Figure 3.13.

This simple iterative approach has been chosen since it is a marginal issue of this thesis and it is still real time capable. Indeed, for a very high bypass point (25 *cm*) that is very near to the start step (distance of 5 *mm*), the update of the trajectory by considering bypass points costs about 0.07 *ms* additional execution time. For a very high bypass point (25 *cm*) that is very near to the goal step (distance of 5 *mm*), it sinks to 0.02 *ms*. So the consideration of one bypass point costs less than 1% of the cycle time (8 *ms*). The execution time is low because computing one 5th polynomial part of the Swing Foot Trajectory mainly consists in solving the system of the boundary equations (3.2), i.e. computing the inverse of 6x6-matrix and multiplying it with a 6x1-vector.

During the computation of the trajectory, the validity of each bypass point is checked. If a bypass point leads to an infeasible trajectory, a false boolean is returned and a null trajectory is generated as for an infeasible nominal trajectory.



(a) step up



(b) step down

Figure 3.13: New Swing Foot Trajectory on stairs of 15 *cm* considering bypass points. The dashed lines represent the front of the foot in (a) and the back of the foot in (b).

Further Improvements

By implementing this new Swing Foot Trajectory, the following goals have been achieved:

- vertical landing to improve balance and precision of motion
- collision avoidance for straight forward motion

Because the bypass points are only considered in the x (and z) direction of the global frame, the collision avoidance is only available for straight forward motions. This could be extending to all kind of motions by generating the Swing Foot Trajectory in the frame of the stance foot that is about to swing and then transforming it into the global frame. Moreover, the achieved collision avoidance could be improved by adapting the y, roll, pitch and yaw trajectories too, so the robot would be able to better pass a wider range of obstacles, as in [33] for instance.

3.5 Center of Body Trajectory Generator

The ZMP-PC of Kajita et al. [10] has already been implemented by Robotis for the generation of the CoM trajectory in the horizontal directions (cf. equations (2.3) in Section 2.2). The CoB then is computed using the horizontal offsets. The former and new methods to generate the relative vertical motion of the CoB (equivalently CoM) are described in Section 3.5.1 and Section 3.5.2. The desired orientation of the CoB is generated as explained in Section 3.5.3.

3.5.1 Robotis' Vertical Trajectory of the Center of Body

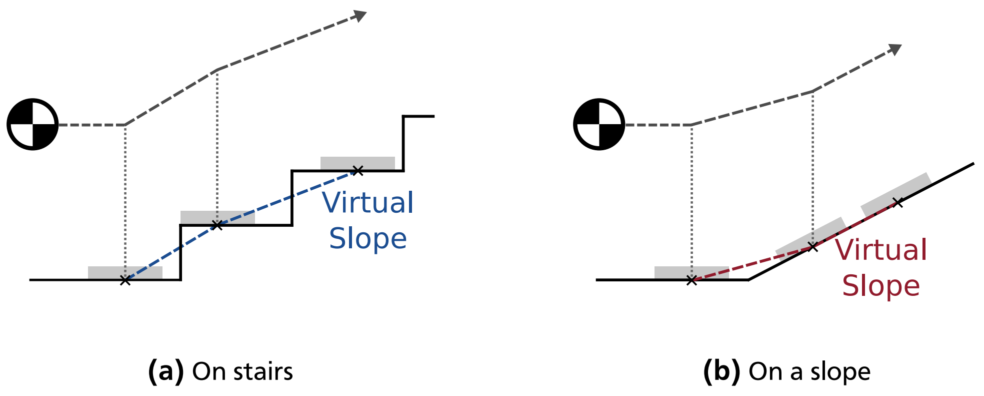
In Robotis implementation the CoB is moving at a fixed height relative to the global frame, without considering the height of the planned footstep. This leads to undesired effects. For instance by walking up stairs, the distance between the feet and the upper body was always becoming smaller (cf. Section 7.4).

To this constant height Robotis added what they called a "swap" motion, which was implemented exactly like the swing complement of the feet (cf. Table 3.1) but with an amplitude $\Delta Z_{cob} = 0.01\ m$. There was no explanation on its purpose, so it was tested by making the robot walk with and without it. The results are presented in Section 6.2.1.

3.5.2 New Vertical Motion of the Center of Body using Virtual Slopes

In order to enable the vertical motion of the CoB accordingly to the terrain, the VS of Sato et al. [19] is implemented (cf. Section 2.3.1) but between two ZMP references. This way, the virtual ZMP reference for the ZMP-PC exactly corresponds to the real ZMP reference, since it already lays on the VS. In [19] the virtual ZMP is placed inside the virtual support polygon which is computed based on the estimated future CoM movement (no details about how to do it in the paper). In the new proposed method these additional computations are dropped. Since the CoB has to move parallel to the VS and keep the same height above it, the linear height trajectory is changed when the CoB passes above the middle of the stance foot (SSP) in the walking direction as illustrated in Figure 3.14.

In [19] it was not mentioned when to "switch" between two Virtual Slopes. To our knowledge, in the literature only methods that switch in the DSP were proposed. For instance Yi et al. [28] did a similar CoM height adjustment but changed the trajectory during the DSP. In fact, because the robot has no knowledge of the terrain, the CoM vertical motion can only be determined after the landing of the swinging foot. At this moment, the local height of the terrain is estimated using sensory feedback and the plane in which the CoM should move is determined. Since the software of Johnny #5 knows the shape of the terrain, it is possible to already adapt the CoM motion (i.e. the CoB motion) in the SSP. Huang et al. [23] also placed the transition between the VS during the DSP. In their method, this results to the disadvantage that the CoM height above the feet changes (cf. Section 2.3). Sun et al. [20] also constructed each new VS during the DSP (which they name Virtual Contact Plane), however between the two actual ZMP references and the ZMP reference corresponding to the next footstep position.



(a) On stairs **(b)** On a slope

Figure 3.14: Virtual Slope between ZMP references

3.5.3 Trajectories of the Center of Body Orientation

If a different orientation of the upper body (i.e. of the CoB) is given by the Footstep Planner, the fifth polynomial function is used to generate the corresponding roll, pitch and yaw trajectories of the CoB. This is done similarly to the feet orientation trajectories. This has already been implemented by Robotis. The Footstep Planner transmits a different yaw angle when the robot is supposed to follow a curved trajectory. The roll and pitch angles are always set to zero.

Further Improvements

When walking on an uneven terrain, it could be an advantage to lean the upper body, for instance to the front on an upslope to reduce the torque in the knees and improve the stability. However, the desired roll and pitch angles should not be determined by the Footstep Planner, but by the GPG itself.

4 Design of the Balance Control

The BC gets the 3D trajectories of the feet and CoB generated by the GPG as inputs. These trajectories correspond to the desired positions and orientations of S_{cob}, S_{rf} and S_{lf} relative to S_{robot}. The Balance Controllers modify these trajectories depending on the feedback of the sensors. The corresponding joint trajectories are computed using IK and are tracked with a PD controller. Afterwards, the joint trajectories generated by the BC are sent to the servomotors. The described procedure is illustrated in Figure 4.1.

In Section 4.1 the balance controllers implemented by Robotis are presented, while Section 4.2 deals with the new balance controllers. All the controllers are discretized using backwards difference. The IK and PD Tracking of the joint trajectories implemented by Robotis were kept unchanged. Numerous safety features were added to the BC software to enable safe testing by avoiding hazardous motions of the robot (cf. Section 4.3).

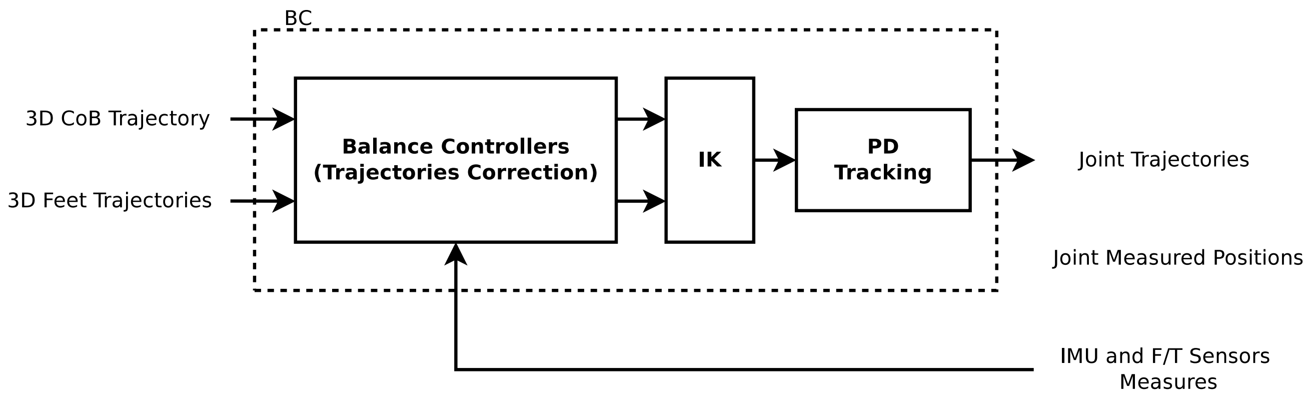


Figure 4.1: Internal structure of the BC

4.1 Robotis' Balance Control of the Feet

Robotis implemented a feet control based on the measures of the F/T sensors and the IMU (cf. Figure 4.3). The noise of these measures is filtered as explained in Section 4.1.1. The used IMU and F/T controls are presented in Section 4.1.2 and Section 4.1.3 respectively. Weaknesses of Robotis' Balance Controllers explaining why the robot sometimes falls over are expounded in Section 4.1.5.

4.1.1 Low-Pass Filters

Robotis implemented digital low-pass filters, which are kept unchanged in the new walking software. In the continuous-time domain, the equation of the filter is

$$x(t) = y(t) + \frac{1}{\omega_c} \cdot \frac{dy}{dt} \tag{4.1}$$

where $x(t)$ is the input signal of the filter, y(t) the output signal and ω_c the cut-off pulsation of the filter.

This equation is converted into the discrete-time domain using backwards difference, which leads to the implemented relation

$$y_i = \alpha x_i + (1-\alpha) y_{i-1} \tag{4.2}$$

where

$$\alpha = \frac{2\pi f_c T_s}{1 + 2\pi f_c T_s} \quad \text{and} \quad f_c = \frac{\omega_c}{2\pi}. \tag{4.3}$$

and T_s is the sampling time. The cut off frequencies f_c chosen by Robotis for the IMU and the F/T sensors are listed in Table E.4.

Remark

For the sake of simplicity all variables with a superscripted "meas" (e.g. x^{meas}) in this thesis correspond to the filtered values.

4.1.2 Robotis' IMU Control

The IMU control ensures that the upper body remains vertical by changing the feet orientation around their own frame. Two decoupled control loops were built, one for the roll control and one for the pitch control. Since they are identically implemented, in the following only the roll control loop is considered. It corresponds to a PD controller on the error of the roll angle, i.e. in continuous-time domain:

$$\Delta\phi_{imu} = -(k_p \cdot e_\phi + k_d \cdot \dot{e}_\phi), \quad e_\phi = \phi^{des} - \phi^{meas} \tag{4.4}$$

where $\Delta\phi_{imu}$ is the roll angle correction for each foot relative to the corresponding foot frame, ϕ^{des} the desired, ϕ^{meas} the measured roll angle of the upper body and k_p and k_d the positive proportional and derivative gains of the controller. The PD controller is preceded by a minus sign because the feet have to rotate in the same direction as the measured inclination of the upper body. Indeed, this will make the upper body rotate in the opposite direction. Since the IMU not only measures angles (ϕ^{meas}) but also angular velocities (v_ϕ^{meas}), no calculation of the derivative of ϕ^{meas} (i.e. $\dot{\phi}^{meas}$) is needed and the PD controller is implemented as two P controllers:

$$\Delta\phi_{imu} = -(\underbrace{k_{p1} \cdot (\phi^{des} - \phi^{meas})}_{\Delta\phi_{imu,1}} + \underbrace{k_{p2} \cdot (v_\phi^{des} - v_\phi^{meas})}_{\Delta\phi_{imu,2}}) \tag{4.5}$$

where the desired roll angle ϕ^{des} and velocity v_ϕ^{des} are set to zero.

Because of these roll and pitch corrections, the feet are generally not coplanar during the DSP. This is probably why Robotis added an additional correction of the feet height $\Delta z_{imu,i}$ computed as follows:

$$\Delta z_{imu,rf} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \boldsymbol{R}(y, \Delta\theta_{imu}) \cdot \boldsymbol{R}(x, \Delta\phi_{imu}) \cdot \begin{bmatrix} x^{des}_{mid\to rf} \\ y^{des}_{mid\to rf} \\ 0 \end{bmatrix} \tag{4.6}$$

where

$$x^{des}_{mid\to rf} = \frac{1}{2}(x^{des}_{robot\to rf} - x^{des}_{robot\to lf}) \tag{4.7}$$
$$y^{des}_{mid\to rf} = \frac{1}{2}(y^{des}_{robot\to rf} - y^{des}_{robot\to lf}) \tag{4.8}$$

and $\boldsymbol{R}(x, \alpha)$ is the standard notation of a 3D rotation matrix describing a rotation of an angle α around the axis x. For the other foot, the correction then is

$$\Delta z_{imu,lf} = -\Delta z_{imu,rf}. \tag{4.9}$$

Later, by designing a new IMU control it has been established that this correction is incomplete to keep the feet coplanar. This is explained in detail in Section 4.2.2.

4.1.3 Robotis' Force/Torque Control

The F/T measures are rotated to correspond to the robot frame orientation. Robotis made the assumption that the measures are made at the origin of the foot frame (under the sole). This assumption is discussed in Section 4.2.5.

The walking is divided into nine Balancing Phases (cf. Section 4.2.1) which are used to generate the trajectory of the desired vertical forces $F^{des}_{z,i}$ ($i \in \{rf, lf\}$). Their absolute value is set to half of the weight of the robot when standing in the initial pose (cf. Section 3.2). During a SSP it is set to the total weight for the stance foot and to zero for the swinging foot. For a DSPs between two SSPs, the desired force is interpolated with a sigmoid function between the values in both SSPs. Since the z axis is directed upwards, $F^{des}_{z,i}$ is negative. The desired torques are all zero and the desired horizontal forces are set to

$$F^{des}_{x,i} = -\frac{1}{2} m_{total} \ddot{x}^{des}_{com} \tag{4.10}$$

where m_{total} is the total mass of the robot and $\ddot{x}^{des}_{com}$ is the acceleration of the CoM in x direction generated by the ZMP-PC. The same applies in y direction. Based on these desired values, the PD controllers listed below were implemented:

$$(F^{des}_{x,i} - F^{meas}_{x,i}) \xrightarrow{\text{PD}} \Delta x_{ft,i}$$
$$(F^{des}_{y,i} - F^{meas}_{y,i}) \xrightarrow{\text{PD}} \Delta y_{ft,i}$$
$$(F^{des}_{z,i} - F^{meas}_{z,i}) \xrightarrow{\text{PD}} \Delta z_{ft,i}$$
$$(\tau^{des}_{x,i} - \tau^{meas}_{x,i}) \xrightarrow{\text{PD}} \Delta \phi_{ft,i}$$
$$(\tau^{des}_{y,i} - \tau^{meas}_{y,i}) \xrightarrow{\text{PD}} \Delta \theta_{ft,i}$$

4.1.4 Robotis' Controllers Summary

The total correction of the orientation and the position of the feet computed by Robotis' BC is

$$\begin{aligned}
\Delta\phi_i &= \Delta\phi_{imu} + \Delta\phi_{ft,i} \\
\Delta\theta_i &= \Delta\theta_{imu} + \Delta\theta_{ft,i} \\
\Delta x_i &= \Delta x_{ft,i} \\
\Delta y_i &= \Delta y_{ft,i} \\
\Delta z_i &= \Delta z_{imu,i} + \Delta z_{ft,i}
\end{aligned} \tag{4.11}$$

where $i \in \{rf, lf\}$. The controllers corresponding to these correction variables are listed in Appendix E.

4.1.5 Limits

The most frequent reason why the robot fell over was the following. The assumption is made that the robot is walking and that the current phase is a SSP on the right foot. When the robot begins to slightly tip over the outer edge of the stance foot, then both feet increase their roll angle (IMU control) and the left foot moves in the upper z direction because of the additional z correction of the IMU control. This leads to a bigger distance between the left foot and the ground (cf. Figure 4.2) and makes this foot touch the ground much later than planned. In Appendix F Figure F.2, snapshots from a walking experiment, during which this phenomenon appeared, are collected. When the robot begins to tilt back to the left and the swinging foot finally touches the ground, the CoB reaches a high velocity and this causes the robot to lean even more to the left than it did to the right. At this moment the robot either falls over or tilts back to the right with an even bigger velocity and falls over on the other side. Moreover, in such a situation the late landing of the swinging foot sometimes occurs after the beginning of the swing phase for the other foot that is still on the ground. This causes the swinging foot to brutally enter in contact with the ground and destabilize the robot even more.

Because of this lack of robustness of the IMU control for large roll angles of the upper body, an additional control loop was added whose aim is to maintain the ZMP inside the support polygon (cf. Section 4.2.6).

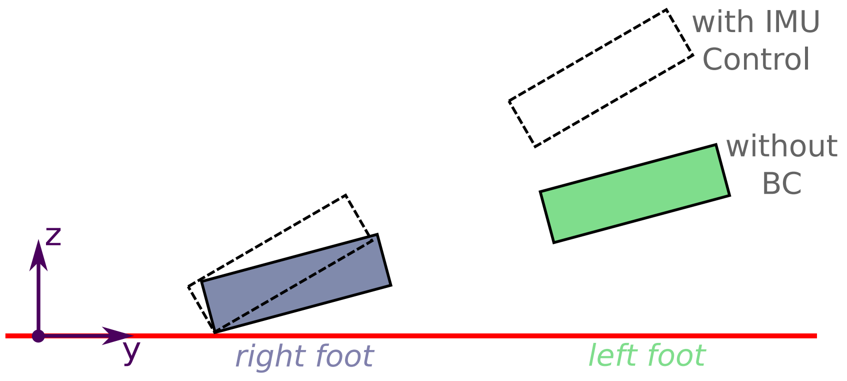


Figure 4.2: Feet positions at the end of a SSP when the robot is tipping over (comparison with IMU control/without any Balance Controllers)

In the new BC the IMU control was slightly changed (cf. Section 4.2.2), new controllers were implemented in the F/T control loop (cf. Section 4.2.3) and a ZMP control was added (cf. Section 4.2.6). The modifications of the global structure are illustrated in Figure 4.3. For the ZMP control, the position of the real ZMP (respectively of the virtual ZMP on the virtual slope) needs to be estimated as explained in Section 4.2.5.

Instead of having all controllers active all the time like in Robotis implementation, the walking gait is divided in new phases (cf. Section 4.2.1) during which only specific controllers are activated as in [26].

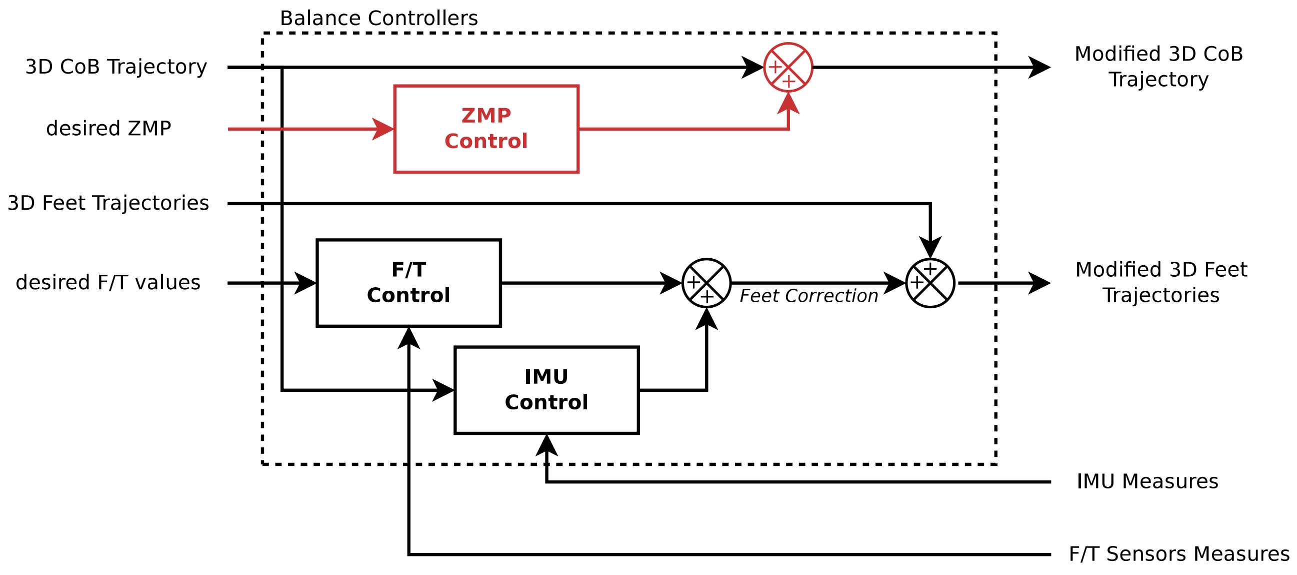


Figure 4.3: Block diagram of the Balance Controllers. The already existing blocks are drawn in black and the new block in red.

4.2.1 Balance Control Phases and Landing Detector

Robotis' Balance Controllers are always active. This means that even if a foot is swinging, a rotation and position correction is applied although it cannot influence on the stability of the robot. Therefore, status variables are implemented that enable to define adapted activation phases for the different controllers (cf. Table 4.1). The Balancing Phases correspond Robotis' Balancing Phases used to determine $F_{z,i}^{des}$ (cf. Section 4.1.3), except that raising and landing phases were added. Some authors also used similar activation phases, especially for controlling the landing of the swinging foot [15, 26, 28].

Since this foot never exactly lands at the planned time but some controllers have to be started at the moment when the foot touches the ground, a Landing Detector is implemented. Its output is set to *true* when $F_{z,i}^{meas}$ is under the predefined landing threshold (here $-150\ N$) and to *false* when it is above the predefined swinging threshold (here $-50\ N$). Based on this detection the landing status of the considered foot is updated. In the next sections the different Balance Controllers are described as well as during which phases they are activated.

Table 4.1: Status variables of the new BC for a walking gait of two default steps (step duration of $1\ s$). The first step illustrates the case of early landing (earlier than planned), while the second step covers the case of late landing.

Timeline [s]	Balancing Phase	First Step Boolean	Last Step Boolean	Landing Detection Right	Landing Detection Left	Landing Status Right	Landing Status Left
	STANDING	true	false	true	true	NO_LANDING	NO_LANDING
0	DSP_MID_TO_L	true	false	true	true	NO_LANDING	NO_LANDING
0.1	RAISING_R	true	false	true / false	true	NO_LANDING	NO_LANDING
	SWINGING_R_1	true	false	false	true	NO_LANDING	NO_LANDING
0.5	SWINGING_R_2	false	false	false	true	NO_LANDING	NO_LANDING
	LANDING_R	false	false	false / true	true	HAS_NOT_LANDED / EARLY_LANDING	NO_LANDING
0.9	DSP_L_TO_MID	false	false	true	true	EARLY_LANDING	NO_LANDING
1	DSP_MID_TO_R	false	false	true	true	EARLY_LANDING	NO_LANDING
1.1	RAISING_L	false	false	true	true / false	EARLY_LANDING	NO_LANDING
	SWINGING_L_1	false	false	true	false	EARLY_LANDING	NO_LANDING
1.5	SWINGING_L_2	false	true	true	false	NO_LANDING	NO_LANDING
	LANDING_L	false	true	true	false	NO_LANDING	HAS_NOT_LANDED
1.9	DSP_R_TO_MID	false	true	true	false / true	NO_LANDING	HAS_NOT_LANDED / LATE_LANDING
2	STANDING	false	true	true	true	NO_LANDING	LATE_LANDING / NO_LANDING

4.2.2 New IMU Control

The IMU control is always activated because the upper body inclination is supposed to be permanently controlled. Robotis' implementation was kept the same, except that corrections in the x and y direction were added, such that the feet are in fact coplanar during the DSP (on flat ground) and keep the exact same relative position during the SSP. This modifies equation (4.6) into

$$\begin{bmatrix} \Delta x_{imu,rf} \\ \Delta y_{imu,rf} \\ \Delta z_{imu,rf} \end{bmatrix} = \boldsymbol{R}(y, \Delta\theta_{imu}) \boldsymbol{R}(x, \Delta\phi_{imu}) \begin{bmatrix} x^{des}_{mid\rightarrow rf} \\ y^{des}_{mid\rightarrow rf} \\ z^{des}_{mid\rightarrow rf} \end{bmatrix} - \begin{bmatrix} x^{des}_{mid\rightarrow rf} \\ y^{des}_{mid\rightarrow rf} \\ z^{des}_{mid\rightarrow rf} \end{bmatrix} \tag{4.12}$$

and

$$\begin{bmatrix} \Delta x_{imu,lf} \\ \Delta y_{imu,lf} \\ \Delta z_{imu,lf} \end{bmatrix} = - \begin{bmatrix} \Delta x_{imu,rf} \\ \Delta y_{imu,rf} \\ \Delta z_{imu,rf} \end{bmatrix} . \tag{4.13}$$

As shown in Figure 4.4, when not considering the y correction, the feet are slightly more apart and not coplanar any more. During a swing phase these discrepancies increase due to the vertical relative position of the feet ($z^{des}_{lf\rightarrow rf}$) was neither considered. It is not known if Robotis neglected it on purpose. Assuming that the right foot is currently situated at the highest point of the Swing Foot Trajectory (or equivalently on a stair of 0.1 m) and that the roll inclination error amounts 0.05 rad, i.e.

$$\begin{gathered} y^{des}_{mid\rightarrow rf} = -0.093, \quad z^{des}_{mid\rightarrow rf} = 0.05, \\ \Delta\phi_{imu} = 0.05\ rad\ (\approx 2.9°) \quad \text{and} \quad \Delta\theta_{imu} = 0\ rad, \end{gathered} \tag{4.14}$$

the y correction $\Delta y_{imu,rf}$ amounts to 2.6 mm. So the relative distance of the feet in the y direction is changed of 5.2 mm, which does not seem negligible. Thus, this additional horizontal correction should be tested on the real robot to verify if it has a noticeable effect. It was not achieved in this thesis because the BC could not be tuned as explained in Section 6.4.3.

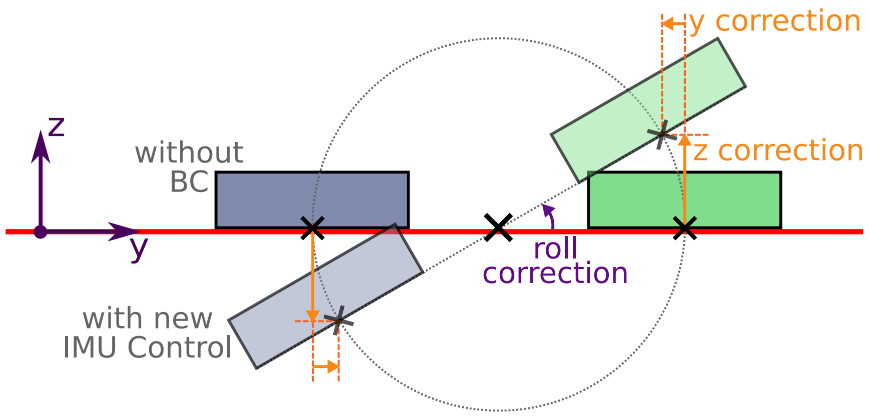


Figure 4.4: Geometrical signification of the position correction of the IMU control

4.2.3 New Force/Torque Control

The controllers of the horizontal forces were removed because the feet cannot move in the horizontal directions while in contact with the ground. This was validated by tests on the real robot (cf. Section 6.2.2).

Though, the adaption towards uneven terrain requires to keep the controllers of the horizontal torques. During the swing phase all torques and therewith correction terms become zero. Hence, an active switching of these controllers is neglected.

The controller of the vertical force is replaced by a landing controller activated for a short period of time beginning at the landing detection and a terrain adaptation controller activated during the DSP, which are the two most common types of vertical force controllers in the literature [27, 28, 34, 35]. The vertical adaptation towards uneven terrain could be done by the PD controllers implemented by Robotis. However, due to calibration offsets or perturbations in the F/T measures (cf. Section 6.4.1), the desired vertical force is not always reachable. More particularly, the goal of the adaptation control during the DSP is not to reach the desired vertical forces $F_{z,i}^{des}$ but to equalize the measured forces on both feet ($F_{z,rf}^{meas} = F_{z,lf}^{meas}$). Therefore, a global PD controller was implemented (instead of local ones for each foot) which takes the target force disparity $\Delta F_z^{des} = F_{z,rf}^{des} - F_{z,lf}^{des}$ as desired value and the measured force disparity $\Delta F_z^{meas} = F_{z,rf}^{meas} - F_{z,lf}^{meas}$ as feedback value (as in [34] or [35] but with another controller type):

$$(\Delta F_z^{des} - \Delta F_z^{meas}) \xrightarrow{\text{PD}} \Delta z_{dsp} \tag{4.15}$$

$$\Delta z_{dsp,rf} = 0.5 \cdot \Delta z_{dsp} \tag{4.16}$$

$$\Delta z_{dsp,lf} = -0.5 \cdot \Delta z_{dsp} \tag{4.17}$$

The generation of the desired vertical force trajectories remains the same as in Robotis' implementation. This controller is activated during the landing of the swinging foot or at the latest during the beginning of the balancing phase "DSP_L_TO_MID" or "DSP_R_TO_MID" (cf. Table 4.1) and is deactivated when one of the landing detectors returns false (after both have returned true). Thus, in case of late landing, the controller is activated before the landing and makes the foot move faster toward the ground because at this moment $\Delta F_z^{des} = 0$ and $\Delta F_z^{meas} \approx \pm m_{total} g$. This enables to diminish the delay of the landing.

In order to reduce the impact during the landing phase, a dedicated landing controller is also implemented. First of all, if the foot lands earlier than planned, its vertical motion is stopped by adding an offset correction $\Delta z_{offset,i}$. At this time, the absolute value of the desired vertical force of the considered foot is possibly smaller than the absolute measured value at the impact. Thus, using this desired value in a control loop would make the foot bounce back in the air. In order to overcome this problem, the desired force is set to the measured force at the impact and is brought to the total weight of the robot using a part of a sinusoidal function. This procedure was proposed in [28] with a linear function. This modification of the desired vertical forces is used for the terrain adaptation controller presented above and for the landing controller. In the literature different types of controllers can be found for the impact reduction (cf. Section 2.4). In this thesis it was decided to keep the possibility to test different types by implementing the following mass-spring-damper controller:

$$k_m \Delta \ddot{z}_{landing,i} + k_d \Delta \dot{z}_{landing,i} + k_p \Delta z_{landing,i} = (F_{z,i}^{des} - F_{z,i}^{meas}) \tag{4.18}$$

where the vertical correction of the foot position $\Delta z_{landing,i}$ ($i \in \{rf, lf\}$) is the ouput of the landing controller. The gains k_m, k_d or k_p can be set to zero depending on which type of controller is to be tested (for instance $k_m = k_d = 0$ to obtain a P controller as in [26]). As explained earlier, this equation is discretized using backwards difference.

At the end of the DSP, the correction of the vertical force control ($\Delta z_{dsp,i} + \Delta z_{offset,i} + \Delta z_{landing,i}$) is generally non null. In order to avoid disturbing the next landing phase, it is smoothly returned to zero during the SSP using a fifth polynomial function (whose implementation is described in Section 3.4.1).

4.2.4 New Feet Controllers Summary

The total correction of the orientation and the position of the feet computed by the new BC is

$$\begin{aligned}
\Delta\phi_i &= \Delta\phi_{imu} + \Delta\phi_{ft,i} \\
\Delta\theta_i &= \Delta\theta_{imu} + \Delta\theta_{ft,i} \\
\Delta x_i &= (\Delta x_{ft,i}) + \Delta x_{imu,i} \\
\Delta y_i &= (\Delta y_{ft,i}) + \Delta y_{imu,i} \\
\Delta z_i &= \Delta z_{imu,i} + (\Delta z_{ft,i}) + \Delta z_{dsp,i} + \Delta z_{offset,i} + \Delta z_{landing,i}
\end{aligned} \tag{4.19}$$

where $i \in \{rf, lf\}$. The terms in red were implemented by Robotis and are removed in the new BC. The terms in black are kept unchanged, the terms in violet are slightly modified and the terms in blue are new. The controllers corresponding to these correction variables are listed in Appendix E.

4.2.5 Estimation of the Real/Virtual ZMP

The measures from the F/T sensors are used to compute an estimation of the position of the real ZMP relative to S_{robot}. The scaled measures $\boldsymbol{F}_i^{scaled}$ and $\boldsymbol{\tau}_i^{scaled}$ ($i \in \{r, l\}$) are in a frame whose origin is at the center of the sensor and whose axes are parallel to the axes of the corresponding foot frame (S_{rf} or S_{lf}). In order to be used in the walking software, these measures have to be transformed into a frame S_{rf}^* (resp. S_{lf}^*) which has the same origin as S_{rf} (resp. S_{lf}) and the same orientation as S_{robot}:

$$\begin{cases} \boldsymbol{F}_i^{meas} = \boldsymbol{R}_{robot \to i} \cdot \boldsymbol{F}_i^{scaled} \\ \boldsymbol{\tau}_i^{meas} = \boldsymbol{R}_{robot \to i} \cdot (\boldsymbol{\tau}_i^{scaled} + \boldsymbol{r}_{ft} \times \boldsymbol{F}_i^{scaled}) \end{cases} \tag{4.20}$$

where

$$\boldsymbol{r_{ft}} = \begin{bmatrix} 0 \\ 0 \\ 0.0275\ m \end{bmatrix} \tag{4.21}$$

is the translation from the S_{rf} (resp. S_{lf}) to the center of the corresponding F/T sensor relative to S^*_{rf} (resp. S^*_{lf}) and $\boldsymbol{R_{robot \to i}}$ is the rotation from S_{robot} to S_{if}.

In Robotis code, the term $\boldsymbol{r_{ft}} \times \boldsymbol{F_i^{meas}}$ was not considered. It was decided to also neglect this term in this thesis, since it is of an order of magnitude smaller than the usual torques values and it would add more noise and errors in $\boldsymbol{\tau_i^{meas}}$. Indeed, as explained in Section 6.4.1 the measures from the F/T are very noisy and can punctually have an error of up to 30% of the used range. Only when the measurements become better, this term should be considered.

Based on the transformed measures $\boldsymbol{F_i^{meas}}$ and $\boldsymbol{\tau_i^{meas}}$, the ZMP position relative to S_{robot} can be computed. The calculation is different depending on the actual support phase (DSP or SSP on right or left foot) and is explained in the next paragraphs. The calculated ZMP corresponds to the virtual ZMP defined by Sato et al. [19] and the way of computing it was inspired from the paper of Sun et al. [20]. In order to know which foot is actually on the ground, the landing detector is used. For visualization purposes the estimated ZMP is transformed into the global frame.

This is an estimation of the ZMP because the desired relative positions of the feet are used for the calculation instead of the real relative positions. This approximation has an influence only in the DSP. This is acceptable since the most critical phase for the ZMP control is the SSP (because of the smaller support polygon).

ZMP Estimation at SSP

In this paragraph the subscript i is used to denote the stance foot which can be either the right ($i = rf$) or the left ($i = lf$) foot. As in [19], if assuming that the moments are null at the ZMP, its position vector $\boldsymbol{\hat{p}_i}$ relative to S^*_i verifies

$$\boldsymbol{\hat{p}_i} \times \boldsymbol{F_i^{meas}} = \boldsymbol{\tau_i^{meas}} \tag{4.22}$$

which can also be written as the following system of equations

$$\begin{cases} F^{meas}_{z,i}\hat{p}_{y,i} - F^{meas}_{y,i}\hat{p}_{z,i} = \tau^{meas}_{x,i} \\ F^{meas}_{x,i}\hat{p}_{z,i} - F^{meas}_{z,i}\hat{p}_{x,i} = \tau^{meas}_{y,i} \\ F^{meas}_{y,i}\hat{p}_{x,i} - F^{meas}_{x,i}\hat{p}_{y,i} = \tau^{meas}_{z,i} \end{cases} \tag{4.23}$$

Because of the cross product the solution is a 3D line equation named the Zero-Moment Line (ZML) [19] as already mentioned in Section 2.3.1. Hence, only two equations of the system (4.23) are needed to solve it. Since $F^{meas}_{z,i}$ is always non null (the foot is on the ground) and $F^{meas}_{x,i}$ and $F^{meas}_{y,i}$ can possibly be both zero, the two first equations of (4.23) are kept.

As described in Section 2.3, in order to overcome the definition problem of the ZMP on all kind of terrain, the ZMP is situated at the intersection between the Zero-Moment Line and the Virtual Slope. It is actually a virtual ZMP, however the word “virtual” will be most of the time omitted in the rest of this thesis for practicality. The general equation of the Virtual Slope is

$$k_x \hat{p}_{x,i} + k_y \hat{p}_{y,i} + k_z \hat{p}_{z,i} = c_0. \tag{4.24}$$

For a Virtual Slope between two planned footsteps as described in Section 3.5.2, the parameter c_0 is zero. For straight forward motions the other parameters are

$$k_x = const, \quad k_y = 0 \quad \text{and} \quad k_z = -1. \tag{4.25}$$

The two first equations of system (4.23) and the Virtual Slope equation (4.24) can be transformed into the following matrix equation:

$$\mathscr{F}_i \hat{\boldsymbol{p}}_i = \mathscr{V}_i \tag{4.26}$$

where

$$\mathscr{F}_i = \begin{bmatrix} 0 & F_{z,i}^{meas} & -F_{y,i}^{meas} \\ -F_{z,i}^{meas} & 0 & F_{x,i}^{meas} \\ k_x & k_y & k_z \end{bmatrix}, \quad \hat{\boldsymbol{p}}_i = \begin{bmatrix} \hat{p}_{x,i} \\ \hat{p}_{y,i} \\ \hat{p}_{z,i} \end{bmatrix} \quad \text{and} \quad \mathscr{V}_i = \begin{bmatrix} \tau_{x,i} \\ \tau_{y,i} \\ c_0 \end{bmatrix} \tag{4.27}$$

Then the solution $\hat{\boldsymbol{p}}_i$ is transformed into the robot frame to obtain a final estimation of the ZMP in a common frame independently of the actual stance foot:

$$\boldsymbol{p}^{est} = \boldsymbol{r}_{robot \to i} + \mathscr{F}_i^{-1} \mathscr{V}_i \tag{4.28}$$

where $\boldsymbol{r}_{robot \to i}$ is the translation vector from the robot frame to the right or left foot frame. In order to verify if the matrix $\mathscr{F}_i$ is always invertible, its determinant is considered:

$$\det \mathscr{F}_i = F_{z,i}^{meas} (\boldsymbol{F}_i^{meas} \cdot \boldsymbol{n}) \tag{4.29}$$

where

$$\boldsymbol{n} = \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \tag{4.30}$$

is a normal vector to the Virtual Slope. Since $F_{z,i}^{meas}$ is non null, the matrix $\mathscr{F}_i$ is non invertible if and only if the Virtual Slope and the force vector $\boldsymbol{F}_i^{meas}$ are parallel. Since the Virtual Slope is constructed between two footsteps, it is under 100% grade (45°), otherwise the motion would not be feasible. On the other hand, the force in z direction is always bigger than the forces in x and y direction when the foot is on the ground, so the force vector $\boldsymbol{F}_i^{meas}$ cannot be parallel to the VS.

Similar equations apply for the ZMP estimation in the DSP and are presented in Appendix D.

Virtual ZMP vs. Extended ZMP

As already mentioned our estimated ZMP corresponds to the virtual ZMP defined by Sato et al. [19], i.e. to the point on the VS where all the moments of the ground reaction are null (considering the whole force reaction). However in the definition of ZMP only the force orthogonal to the ground is considered and only the moments parallel to the ground have to be zero. Sato et al. did not explain why they considered all moments to be null.

Sun et al. [20] proposed another method (EZMP method, cf. Section 2.3.1) where only the force orthogonal to the VCP is considered and only the moments parallel to the VCP have to be zero to calculate the EZMP position. Hence, this point is conform to the definition of ZMP [16]. On flat ground both methods lead to the same equations, i.e. the ZMP is on the Zero-Moment Line (cf. proof in Appendix D).

This definition of EZMP could be used to compute our estimated ZMP. However, it is not obvious which method is better. Indeed, on stairs in the SSP the real ZMP is on the ZML(as on flat ground). Thus, an advantage of the virtual ZMP method is that there is a simple relation between the real and the virtual ZMP (on local horizontal ground), since both are on the ZML. On the contrary the EZMP of Sun et al. has no explicit relation with the real ZMP. Moreover, this method of EZMP was used with a triple preview control which differs from the implemented VS method.

We had no time to investigate both methods and chose to implement the virtual ZMP since our GPG is based on the same equations and a similar VS method. Moreover, the EZMP method was only tested in simulation [20] while the virtual ZMP method was also successfully tested on a real robot [19].

4.2.6 ZMP Control

In the GPG the desired CoM trajectory is generated from the desired ZMP reference. Thus, it seems logical to control the ZMP error (difference between the desired and estimated ZMP) by directly changing the CoM horizontal position. For this purpose PD controllers were implemented for the x and y correction and are always activated.

4.3 Safety Features

Safety features are really important for testing the software on the real robot to avoid hazardous motions for the robot itself and for humans around. This generally means, when a malfunctioning is detected, the motion has to stop (i.e. no new command values are sent to the motors). This is done by adding specific detectors described in the following sections. Except for the saturation functions in the BC there were no safety features in Robotis software.

4.3.1 Saturation of the Balance Control Correction

The BC correction is saturated to prevent the robot from doing too large movements. In this case, the whole motion of the robot is not supposed to be stopped, since it does not necessarily means that something is going wrong or that the system became unstable. However, the saturation values chosen by Robotis were too large, making it possible for the robot to collide with itself. Thus, new values were determined based on the standard correction values when the robot is stably walking. The old and new values are listed in Appendix E.

4.3.2 Fall Detection

When the pitch or roll absolute angles measured by the IMU are bigger than 0.23 rad ($\approx 13°$), the motion of the robot is stopped because the robot is surely falling over. This threshold value has been set by comparing the measured angle values when the robot was stably walking with the values when it tipped over during walking.

4.3.3 Detection of Communication Break Down

By running experiments on the real robot, communication break downs have sometimes been encountered in the right leg of the robot, i.e. no new measures were being received from the encoders and the F/T sensors. In such a situation, the robot has an unpredictable behavior and tips over because the BC cannot work correctly without sensory feedback.

In order to avoid this hazardous reaction of the robot, a detector of communication break downs was implemented. It stops the motion if the measures received from the IMU or from the F/T sensors keep the same value for a given period. Because of the noise of these sensors, this only happens, when the communication fails. On the contrary, this detector cannot be applied to the encoder measures because they take discrete values and may therefore stay constant during a long period even if the communication works perfectly. Since the F/T sensors are at the end of the communication chain which goes through the motors of the leg to the serial board, any break in the chain will affect the F/T measures. Therefore, there is no need to implement an extra detector for the encoder measures.

This new safety feature has successfully been tested by turning on the Balance Control when the robot is in the air in the initial pose and disconnecting a communication cable from the serial board.

Later, it has been found out that the communication break downs in the right leg were caused by a defective motor in the chain which was then replaced.

5 Implementation

The existing software of Robotis was not well structured, hence it was very time-consuming to develop it further (cf. Section 5.1). So it was decided to implement it in a new modular structure named the Legged Locomotion Library (l3) which is presented in Section 5.2. Moreover, a 2D visualization tool is implemented in order to better evaluate the performances of the algorithms (cf. Section 5.3). The former as well as the new walking software are based on ROS and C++.

5.1 Robotis' Code Structure

The part of Robotis' software directly responsible for the walking of the robot is mainly composed of two ROS packages:

- the *thormang3_walking_module* package which contains the GPG, the IK and the PD Tracking blocks (cf. Figure 4.1)
- the *thormang3_balance_control* package which contains the Balance Controllers and is called by the previous package

Inside this packages the code is essentially divided in only two large *process()* functions. This absence of clear structure hinders code readability and development. Moreover, instead of classically make use of the existing Unified Robot Description Format (URDF) files[1], the IK was implemented based on hard coded values. Generally, most of the parameters were manually set in *initialize()* functions to a fixed value and sometimes in several functions at the same time. This does not permit an easy change of the parameter values and can lead to inconsistencies in the code. Robotis' walking software is open source on GitHub and can be found at the following link:

`https://github.com/ROBOTIS-GIT/ROBOTIS-THORMANG-MPC`

The version implemented on Johnny #5 before the beginning of this thesis can be found in this other GitHub repository:

`https://github.com/thor-mang/ROBOTIS-THORMANG-MPC`

5.2 New Implementation

The main goal of implementing a library structure is to have the possibility to easily test different methods as the ones presented in Section 2.3 and Section 2.4 and thus to gradually improve the code. For instance it could enable to test different Swing Foot Trajectories, different Balance Controllers, etc. On the long term the other goal is to make l3 open source and thus it should be implemented such a way that it can easily be adapted to different humanoid robots.

The re-implementation itself of Robotis' software into a library is a parallel project conducted by another student. However, it uses the knowledge acquired in this thesis about humanoid walking and especially about ZMP-PC methods and the abstract structure of l3 was elaborated collectively. Sadly, the project was not finished before the end of this thesis. Nevertheless, our improvements of the walking

[1] `http://wiki.ros.org/urdf`

software will be added into l3 when the time comes. The improved walking software is stored on the internal server of the SIM lab (as well as l3) under the following link:

`https://git.sim.informatik.tu-darmstadt.de/ThorMangDev/ROBOTIS-THORMANG-MPC`

Other parts of the software of Johnny # 5 had to be adapted and the modifications are saved in branches of the GitHub repositories as listed below:

- **Package:** *thormang3_walking_module_msgs*
  Branch: *walking_vis_upgrade*
  Description: contains the modified ROS messages for the visualization tool (cf. Section 5.3)
  URL: `https://github.com/thor-mang/ROBOTIS-THORMANG-msgs`

- **Package:** *thor_mang_gazebo*
  Branch: *gazebo_fix_friction*
  Description: contains the .world files for simulating the robot on different terrains (stairs, slopes, plateaus)
  URL: `https://github.com/thor-mang/thor_mang_simulation`

- **Package:** *thormang3_description*
  Branch: *urdf_upgrade_housings* (merged into *kinetic_devel*)
  Description: added the 3D printed housings in the URDF files
  Branch: *gazebo_fix_friction*
  Description: added "PD gains in the feet" to improve the measured vertical forces in the simulation (cf. Section 6.1)
  Branch: *urdf_fix_inertias*
  Description: set the real inertias of the components in the URDF files (cf. Section 6.1)
  URL: `https://github.com/thor-mang/ROBOTIS-THORMANG-Common`

5.3 Visualization Tool

In order to better evaluate the performances of the software than just watching the robot walk, a 2D Visualization Tool was implemented (as a ROS package). ROS publishers and messages were added in the walking software in order to access all the important variables. In the 2D Visualization Tool, the corresponding topics are subscribed and all the data are stored in a table form in files (using blank spaces and line breaks), such that they can directly be opened as a matrix in Scilab (free software equivalent to MATLAB)[1]. This way, all the signals can easily be plotted in 2D. Examples of these plots can be seen throughout this thesis. The recording of the data can be started manually at every moment or automatically when the robot starts walking. In the first mode the recording has to be stopped manually too, while in the second mode it is automatically stopped when the current walking gait is finished. The visualization tool as well as Scilab templates are also stored on the internal server of the SIM lab under:

`https://git.sim.informatik.tu-darmstadt.de/ThorMangDev/ROBOTIS-THORMANG-MPC`

[1] See `https://wiki.scilab.org/` and `https://help.scilab.org/` for more information on Scilab.

6 Tuning and Functional Testing

For safety reasons, the software is always tested in the simulation first as presented in Section 6.1. However, many aspects of the implementation can only be tested on the real robot. In Section 6.2 the functional tests performed with Robotis code are presented, while Section 6.4 focuses on the functional tests with the improved implementation and on the tuning of its parameters. Most of the tests were conducted on Robotis' default walking gait, which corresponds to a straight forward motion on flat ground with a step length of 0.1 m and a step duration of 1 s. Before the beginning of this thesis, the robot was only able to walk with these values of the step parameters.

6.1 Open Loop Tests in Simulation

Since the physical models in the simulation software Gazebo are much simpler than the reality and since they do not take the imperfections of the hardware into account, the robot can walk in open loop mode in the simulation (i.e. without BC). This permits to test the new GPG independently, like for instance the new Virtual Slope method or the new Swing Foot Trajectory and its collision avoidance feature. With the new software, the robot can walk on uneven terrain like stairs or plateaus as shown in Section 7.

Because of hardware upgrades the URDF model had to be modified to better fit the reality. The new housings designed to hold electrical components and the spacers to heighten the torso were added in the URDF files with their measured weights. Moreover, some CoM positions and some weights of the "links" of the robot had to be corrected. After these modifications the model returned a total mass of about 40 kg, although the real mass corresponds to 48.5 kg. This discrepancy can be partially explained by some electrical components like cables that were not weighted. The model was not further improved since it was not the main focus of this thesis and it was not possible to disassemble the robot to verify the weight of every link.

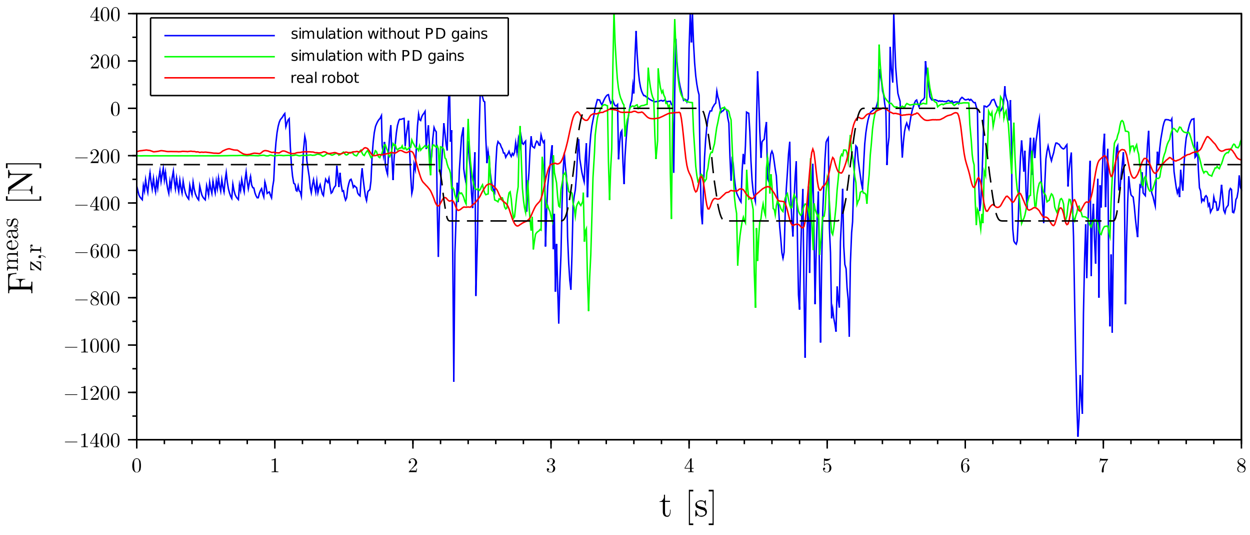
In the URDF files, all the inertias were arbitrarily set to the identity matrix. Robotis furnishes a table of the real inertias of the links of the robot skeleton. The inertias of the new hardware (like the housings) were determined from the CAD[1] software Solid Edge. When setting the real inertias in the URDF files, the gains of the simulated motors have to be tuned again. Indeed, the motors of the joints are simulated as PID transfer functions in Gazebo. Since the real inertias are much smaller than the identity matrix, the robot exploded in the simulation with the existing (very large) PID gains. On the contrary, too small gains make the robot collapse on the spot. Tuning these gains is really time-consuming and was thus not pursued, since it is not the aim of this thesis and it is not sure if it will noticeably improve the simulation accuracy.

It was not possible to test the software in closed loop mode in the simulation (i.e. with BC), because the simulated F/T sensors in Gazebo deliver wrong values. As found in Robotis repository, PD gains were added in the URDF file to model the contact between the feet and the ground. This helped improve the simulated measures of the vertical forces, nevertheless they are still very different from the real measures and the other forces and torques did not become better (cf. Figure 6.1). A lot of other parameters can be tuned in Gazebo to improve the physics, but were not studied here.

[1] Computer-Aided Design

Even if the robot could not walk stably with the BC in the simulation for the mentioned reasons, some primary aspects have been tested, that do not depend on the quality of the measures. These are listed below:

- activation/deactivation of the Balance Controllers at the right time
- direction (i.e. sign) of the corrections
- continuity of the corrections
- safety features



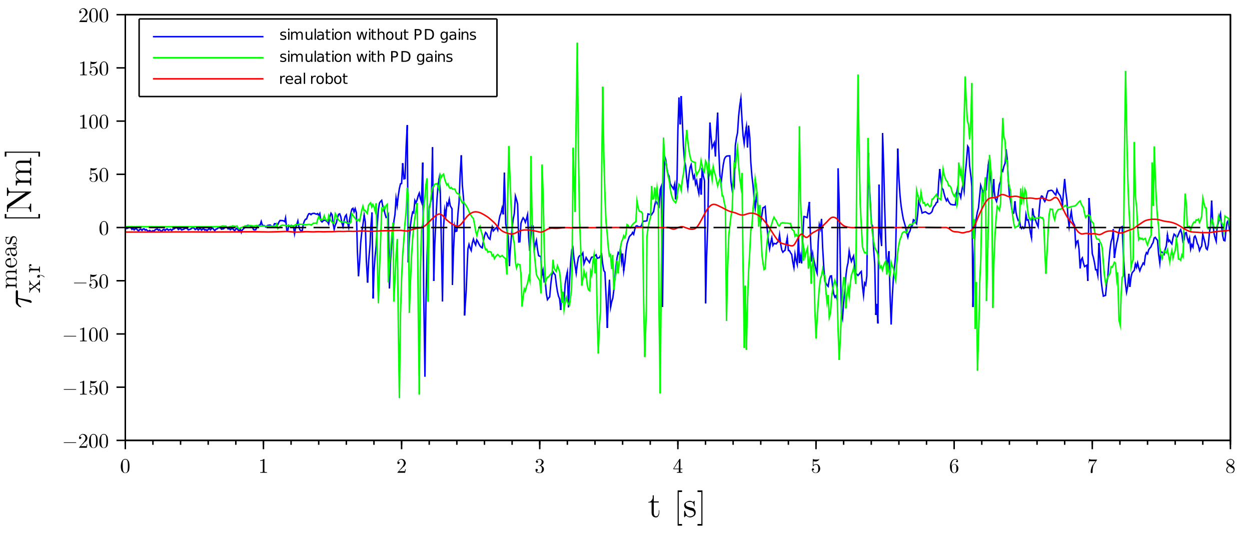


Figure 6.1: Comparison between the measures by the simulated and by the real F/T sensors during a default walking gait. The dashed lines correspond to the desired values.

6.2 Testing Former Code Performances

Because no documentation is provided with Robotis' code, the aim of the body swap motion and of the horizontal forces controllers were not clear (cf. Section 3.5.1 and Section 4.2.3). Thus, it was decided to test their influence on a default walking gait. The results are presented in Section 6.2.1 and Section 6.2.2 respectively.

6.2.1 Center of Body Swap Motion

As explained in Section 3.5.1, Robotis implemented a vertical swap motion of the CoB as illustrated in Figure 6.2. Its aim is unknown and the amplitude of this motion is fixed to an arbitrary value of 0.01 m. Thus, it was decided to test Robotis' default walking gait with and without swap motion on the real robot. Johnny #5 could walk stably in both cases and all the recorded signals were very similar except for the knee pitch position (cf. Figure 6.3). When the robot is standing on one foot, for instance on the right foot between $t = 4\ s$ and $t = 5\ s$, the knee pitch joint is not strong enough to reach the command value generated without swap motion. On the other hand, the joint follows the desired trajectory in the case of a swap motion, probably because the command value is larger. This may also be explained by the generated vertical CoB (i.e. CoM) acceleration which reduces the efforts in the knees. For these reasons, it was decided to keep this swap motion, though it should be further investigated how to automatically adapt its amplitude to the gait and the terrain.

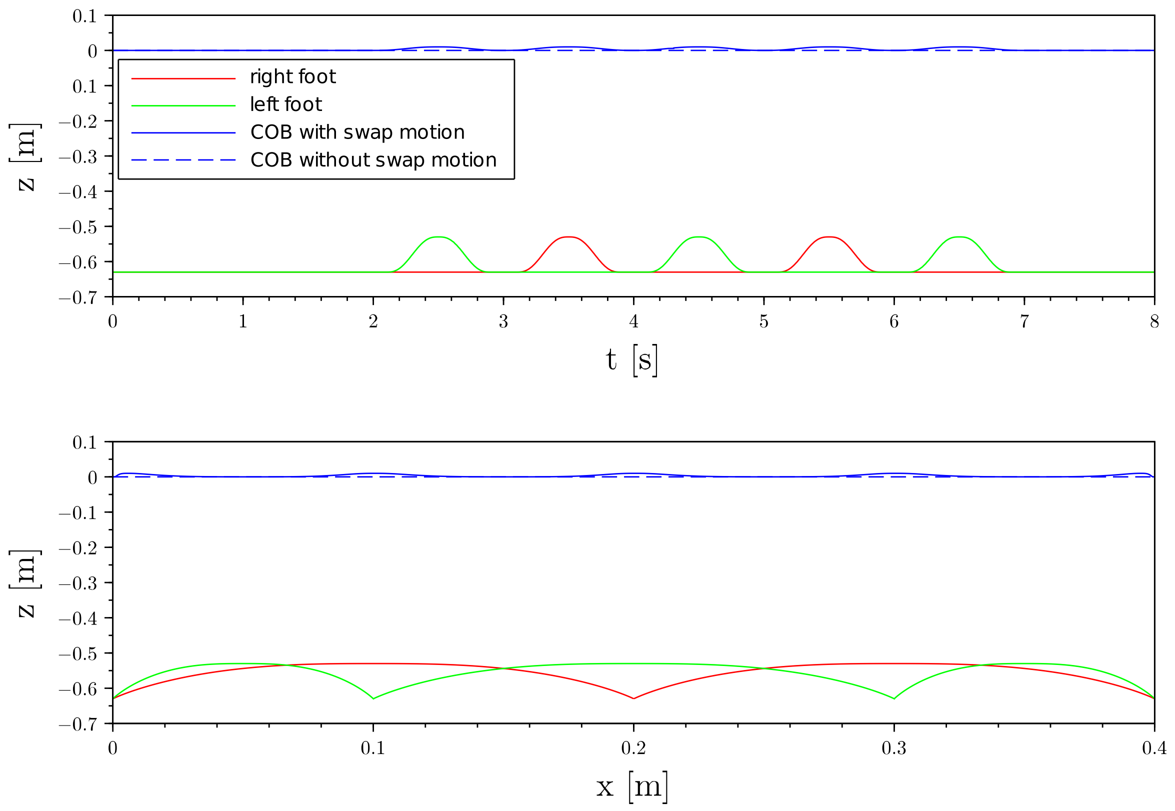


Figure 6.2: Robotis' feet and CoB trajectories generated for a default walking gait with or without swap motion of the CoB

In Robotis implementation, the origins of S_{robot} and S_{cob} correspond to the generated CoB trajectory with swap motion. In order to distinguish the VS motion generation from it, in the new implementation the origin of S_{robot} was set at the CoB position without swap motion. The definition of S_{cob} remains unchanged. Thus, S_{robot} is always situated at a constant height above the feet which is meaningful since the trajectories of the CoB and the feet are transformed into this frame for the BC. Moreover, the horizontal position of S_{robot} was set to the middle of the feet. In Robotis' code it was horizontally moving with the CoB (as computed in the ZMP-PC). These modifications enable to have constant y and z positions of the feet relative to S_{robot} as input for the BC when the robot is moving straightforward on flat ground.

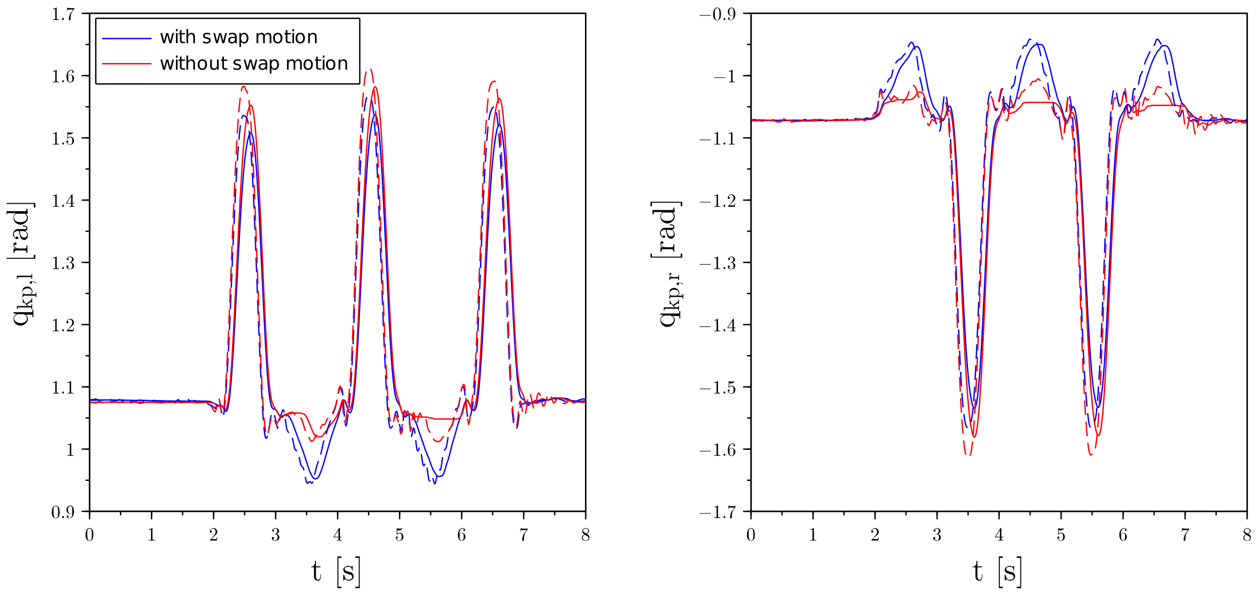


Figure 6.3: Position of the left and right knee pitch joints by a default walking gait with or without swap motion on the real robot. The dashed lines correspond to the commands sent to the motors and the continuous lines to the measured joint position.

6.2.2 Fx/Fy Controllers

As mentioned in Section 4.2.3, Robotis' control of the horizontal forces was suppressed. This was validated by walking experiments. The aim of the implemented controllers is to reduce the error between the desired and the measured horizontal forces by correcting the horizontal position of the feet. However, as illustrated in Figure 6.4 the controllers have no noticeable influence on the course of the horizontal forces and generally on all the recorded signals. Thus, these controllers have no influence on the stability of the locomotion.

Because the stance foot cannot move in the horizontal directions, this control loop physically resulted in a displacement of the CoB in the opposite direction. Thus, these controllers were replaced by a ZMP control which corrects the horizontal position of the CoB directly (cf. Section 4.2.6).

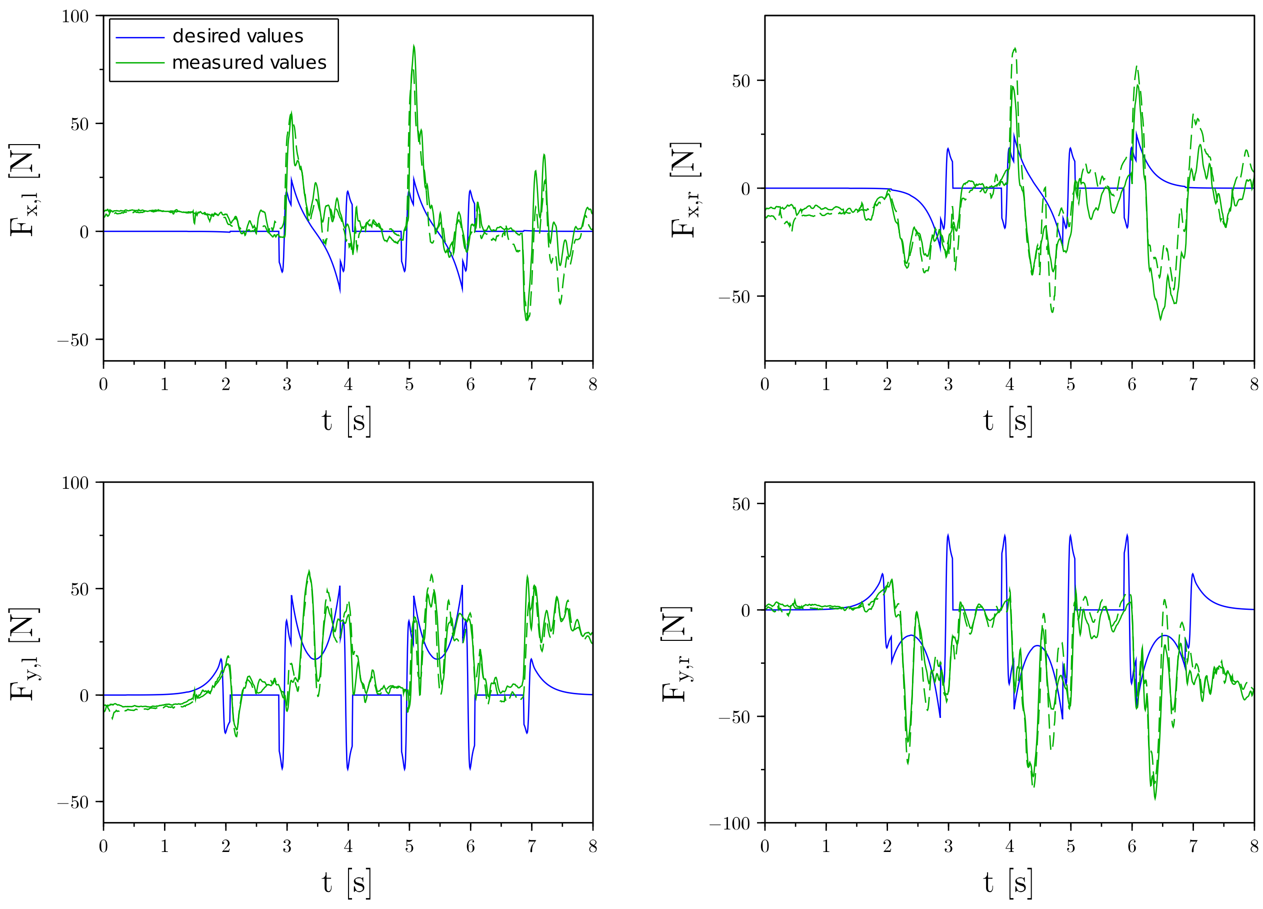


Figure 6.4: Measured horizontal forces during a default walking gait with the real robot. The continuous and dashed lines correspond to the experiment where the controllers of the horizontal forces were activated and deactivated, respectively.

6.3 Tuning and Testing of the New Gait Pattern Generator

In this section, the tuning of the parameters of the new GPG and some functional tests are presented. In order to test and compare it with the former implementation, Robotis' BC is used. The first parameters to be set are the calibration offsets of the legs as explained in Section 6.3.1. Then, the horizontal CoB offsets are determined to center the CoM above the feet (cf. Section 6.3.2). The GPG also needs the height of the CoM, which was experimentally determined as described in Section 6.3.3.

6.3.1 Calibration of the Legs

After bringing the robot in the initial pose (cf. Section 3.2) while hanging in the air, the joints of the legs can be calibrated by setting offsets in order to ensure that they are in the right position. By this process, the following characteristics should be checked:

- parallelism of the links (from the highest to the lowest)
- hip pitch joints placed vertically above the ankle pitch joints
- height of 0.630 m of the CoB above the floor

- zero roll and pitch angles of the upper body (using the IMU)
- inner edges of the feet slightly lower than the outer edges

Nevertheless, it is difficult and time consuming to set perfectly the roll and pitch angles to zero using the calibration offsets. Thus, this was done approximatively and refined using CoB pitch and roll offsets that were added to the existing x and y offsets (cf. Section 3.1 and Section 6.3.2). The feet are calibrated to be slightly not parallel in order to diminish the risk that they land on the outer edge when walking.

This calibration process is very important in order to have an accurate IK. If for instance the right leg is slightly longer than the other because of a wrong calibration, this offset error will create a much harder landing impact on the right foot. Even if the robot now possesses a landing controller, such an error should be avoided since it occurs during the whole locomotion and therefore has a negative influence on the stability of the gait.

6.3.2 Centering of the Center of Mass above the Feet

The CoB horizontal offsets (cf. Section 3.1) are set such that the CoM is situated above the middle of the feet. Since the CoB is actually at this position at the initial pose (cf. Section 3.2), it is moved slowly to the offset position when switching to the walking mode.

The following procedure enables to tune the x and y offsets as well as the roll and pitch offsets (cf. Section 6.3.1):

1. Bring the robot in the initial pose while hanging in the air.
2. Switch to the walking mode, deactivate the BC and put down the robot.
3. Increase/decrease the pitch offset until the IMU pitch value is zero.
4. Repeat with the roll offset.
5. Increase/decrease the x offset until the estimated ZMP in x direction is zero.
6. Repeat with the y offset.
7. Check again the IMU measurement and the estimated ZMP values and refine the tuning if necessary.

Similar horizontal offsets were used in [5] for carrying heavy objects which were shifting the CoM of the robot. As shown in this reference, the procedure can be automated. Though, in our case it is not necessary since it is done before and not during the walking.

These experimental offsets have the advantage to overcome calibration and modeling errors in comparison to a method which would use the robot model to determine the position of the CoM relative to the CoB. The former x and y offsets tuned in an unknown way by Robotis are summed up in Table E.6. Actually, the CoB offsets should be tuned again after each new calibration, therefore no values of the new ones are given.

6.3.3 Estimation of the Height of the Center of Mass

The gains of the ZMP-PC equations depend on the height H_{com} of the CoM (cf. Section 2.2). In Robotis code two sets of gains were computed for the two values 0.5 m and 0.6 m and the first set was used in the ZMP-PC. Apparently, Robotis tuned manually H_{com} by trial and error without looking for a physically plausible value because $H_{com} = 0.5\ m$ means that the CoM is situated between the knees and the hips which is unlikely.

A way to compute the height of the CoM could have been to use the URDF files of Johnny #5. But the total mass computed using the URDF is too different from the real mass of the robot (cf. Section 6.1), so the position of the CoM does probably not correspond to the reality. For this reason, it was decided to experimentally estimate H_{com}.

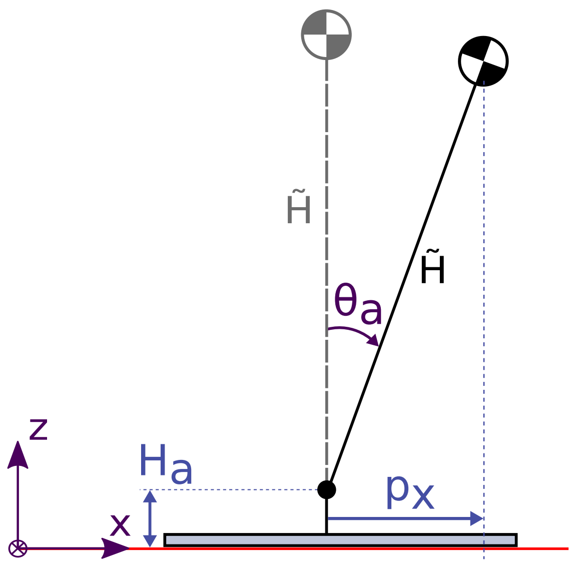


Figure 6.5: Principle of the experimental estimation of the CoM height

The principle of the experiment is illustrated in Figure 6.5 and is based on the fact that the ZMP corresponds to the projection of the CoM on the ground when the robot is standing. Moreover, the assumption was made that a rotation of the feet has a marginal influence on the CoM position relative to the CoB. The stages of the experiment are described below:

1. The Balance Control is deactivated.
2. The CoM is placed above the feet using the tuned offsets (cf. Section 6.3.2).
3. The pitch ankle joints are moved to an arbitrary angular position θ_a.
4. The position p_x of the ZMP in the x direction is estimated from the measures of the F/T sensors as described in Section 4.2.5.

5. The height H_{com} of the CoM is then computed using the equation

$$H_{com} = H_a + \underbrace{\frac{p_x}{\sin\theta_a}}_{\tilde{H}} \tag{6.1}$$

with H_a the height of the ankle above the ground ($H_a = 0.120\ m$).

6. The experiment is repeated for several values of θ_a.

The results are plotted in Figure 6.6. The ankles were moved from 0.05 rad to 0.13 rad because for smaller values the x coordinate of the ZMP was smaller than the uncertainty on its measure (cf. Section 6.4.2) and for greater values the robot tipped over. Due to the backlash in the joints and the deformation of the rubber sole, the pitch angle of the robot is actually different from the angle of the ankle pitch joints. Thus, θ_a was set to the absolute measures of the IMU in equation (6.1). We decided to determine H_{com} to the decimeter since a change of this parameter in the centimeter range leads to a change of the CoM horizontal position in the millimeter range (it was observed by computing the ZMP-PC for heights between 0.5 m and 0.8m). Moreover, the accuracy of the estimated ZMP does not permit to obtain a more precise value (cf. Section 6.4.2). Therefore, based on Figure 6.6 the result of the experiment is

$$H_{com} \approx 0.7\ m \tag{6.2}$$

which approximatively corresponds to the IMU height and seems physically plausible since it is just underneath the batteries.

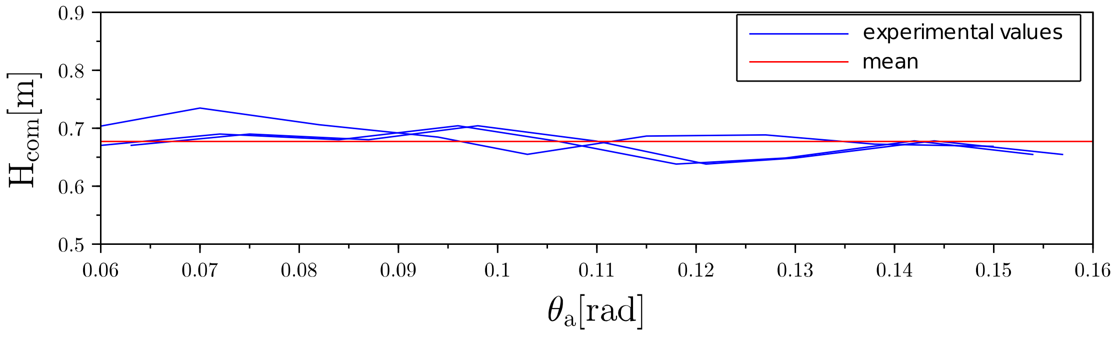


Figure 6.6: Computed values of H_{com} relative to the pitch angle of the robot measured by the IMU and their mean value

In Figure 6.8 the CoM trajectories generated respectively with $H_{com} = 0.5\ m$ (Robotis value) and $H_{com} = 0.7\ m$ (new value) are illustrated. In the second case, the CoM is moving earlier in the forward direction and the minimal distance between the CoM and the middle of the stance foot in the transverse direction is bigger. Indeed, the new preview gain takes lower values for the near future and greater values later, as illustrated in Figure 6.7.

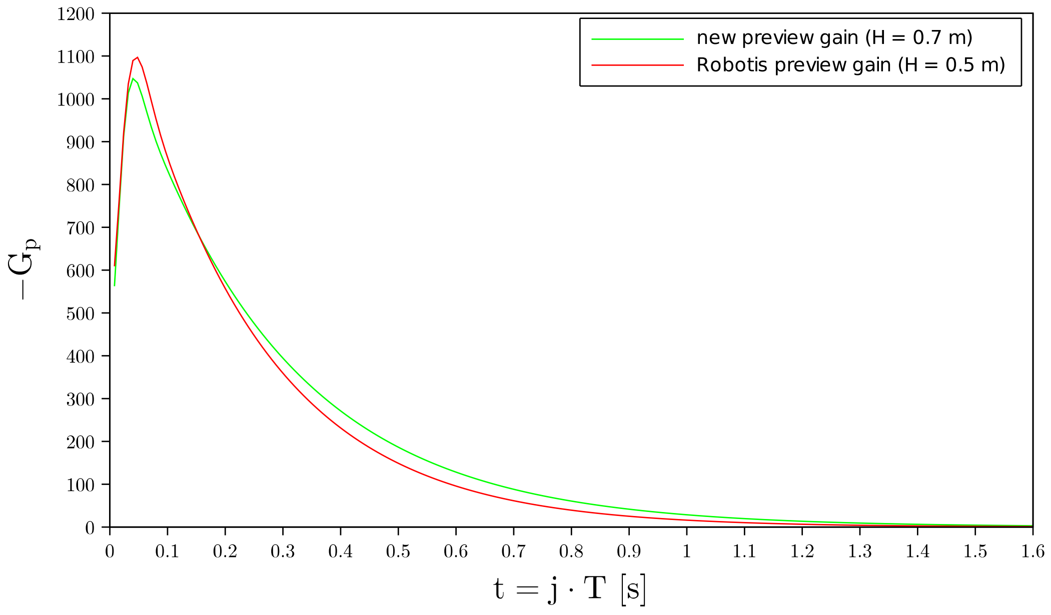


Figure 6.7: Preview Gain of the ZMP-PC depending on the height H_{com}

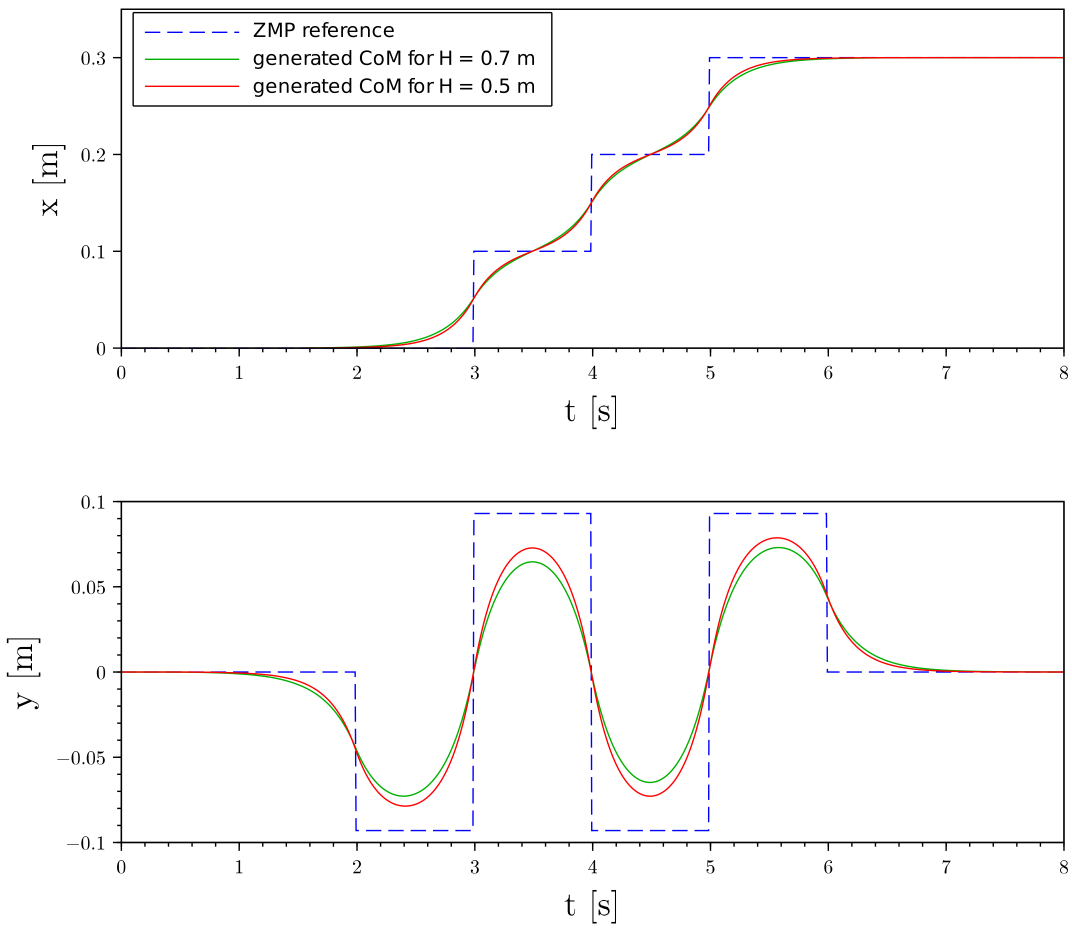


Figure 6.8: Trajectories of the CoM generated by the ZMP-PC depending on its height H_{com}

Another way to determine the height of the CoM could have been to lay the robot on a board balancing on a cylinder, as the SIM lab did in the past. However, the proposed method is easier to reproduce, since no additional material is needed, and can therefore be rapidly done after a significant hardware modification.

Real Test

Robotis' GPGs with $H_{com} = 0.5\ m$ and the new one with $H_{com} = 0.7\ m$ were tested and compared on the real robot for a default walking gait. In both cases Robotis' BC was activated and the robot walked stably. As illustrated in Figure 6.9, in the second case the estimated ZMP stays nearer to the middle of the foot, reducing the risk to tip over sideways which was the biggest issue so far (cf. Section 4.1.5). Nevertheless, during the DSP the rear foot is shortly landing again (for instance at $t \approx 3.7\ s$) because the swinging foot landed earlier than planned, making the robot slightly oscillating back and forth during the DSP. This can be fixed by activating and tuning the new landing and ZMP control loops (cf. Section 4.2.3 and Section 4.2.6) and is less critical for the stability than the ZMP moving to the outer edge of the stance foot as with Robotis' GPG (cf. Figure 6.9). Moreover, in [28, 25] the same phenomenon of re-landing appears by testing the GPG.

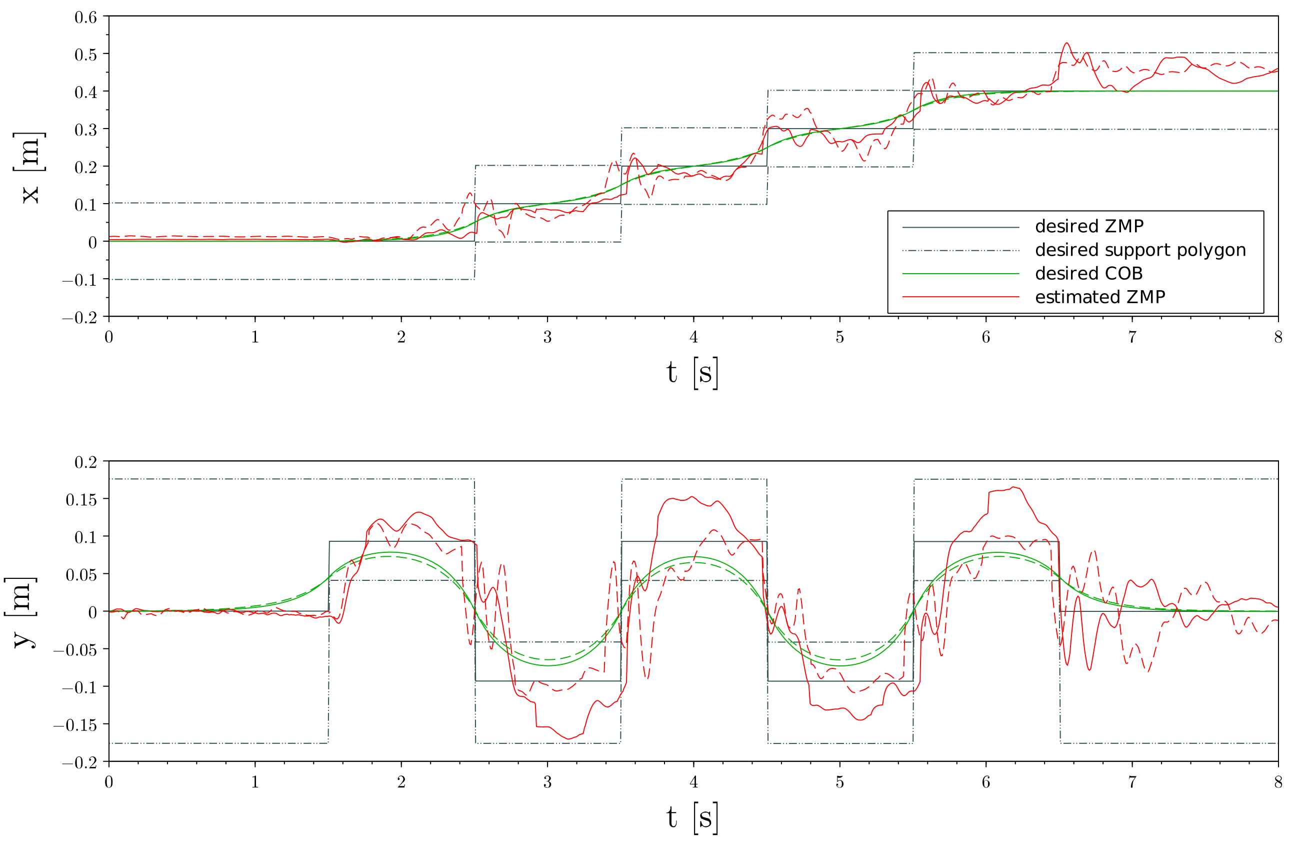


Figure 6.9: Estimated ZMP by walking on flat ground for different values of H_{com}. The continuous line corresponds to a walk with Robotis GPG ($H_{com} = 0.5\ m$) and Robotis BC. The dashed line corresponds to a walk with the new GPG ($H_{com} = 0.7\ m$) and Robotis BC.

6.4 Tuning and Testing of the New Balance Control

For testing the improved BC, the new GPG with the tuned parameters is used. Before verifying the validity of the ZMP estimation (cf. Section 6.4.2), the existing calibration process of the F/T sensors has been studied (cf. Section 6.4.1). However, the most complex work with the BC is the tuning of all the gains of the implemented controllers (cf. Section 6.4.3).

6.4.1 Calibration of the F/T Sensors

The calibration implemented by Robotis works as follows. Measures of the F/T sensors when the robot is hanging in the air ($\tilde{\boldsymbol{W}}_i^{air}$) and when the robot is standing on the ground ($\tilde{\boldsymbol{W}}_i^{ground}$) are recorded and used to scale the measures of the sensors. The aim of the calibration is that $F_{z,l}^{scaled} + F_{z,r}^{scaled}$ is equal to the total weight of the robot when it is on the ground and to zero when it is in the air. The implemented scaling equation is

$$\boldsymbol{W}_i^{scaled} = \alpha_{scale} \cdot (\boldsymbol{W}_i^{raw} - \tilde{\boldsymbol{W}}_i^{air}), \quad i \in \{l, r\} \tag{6.3}$$

with

$$\boldsymbol{W}_i^k = \begin{bmatrix} F_{x,i}^k \\ F_{y,i}^k \\ F_{z,i}^k \\ \tau_{x,i}^k \\ \tau_{y,i}^k \\ \tau_{z,i}^k \end{bmatrix} \tag{6.4}$$

and

$$\alpha_{scale} = \frac{m_{total} \cdot g}{\tilde{F}_{z,l}^{ground} + \tilde{F}_{z,r}^{ground} - \tilde{F}_{z,l}^{air} - \tilde{F}_{z,l}^{air}} \tag{6.5}$$

where m_{total} is the total mass of the robot. Robotis computed this mass using a hard coded model of the robot (not even the URDF) which delivers a mass of 42.533 kg. By weighting the real robot a mass of 48.5 kg is obtained, so the variable in the code is set to this value.

Remark

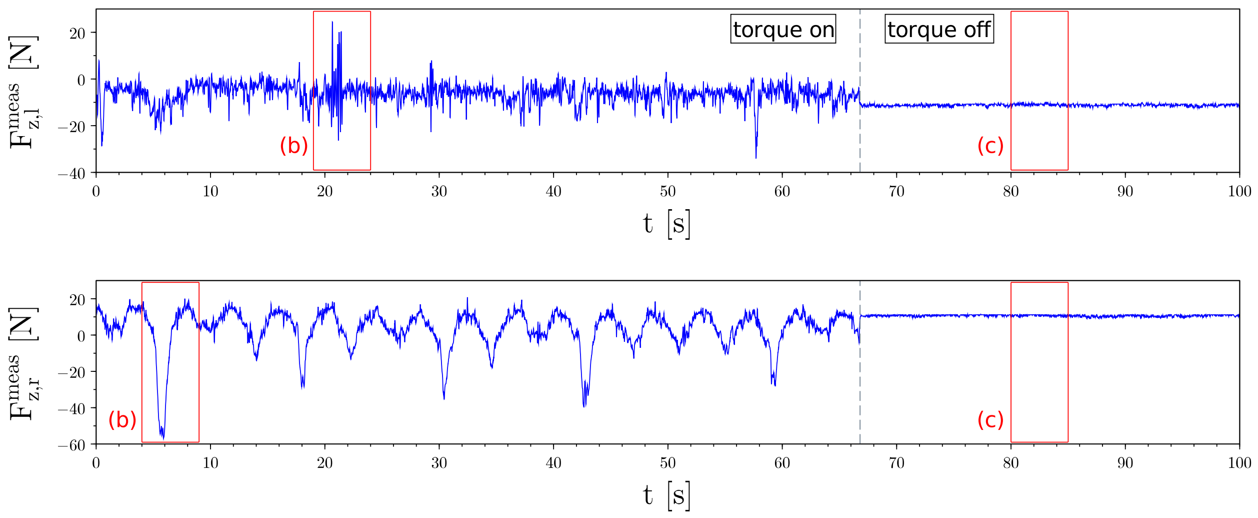
In all this thesis every force or torque variable named with the upper word "meas" (e.g. F_z^{meas}) corresponds to the scaled and filtered value.

Further Improvements

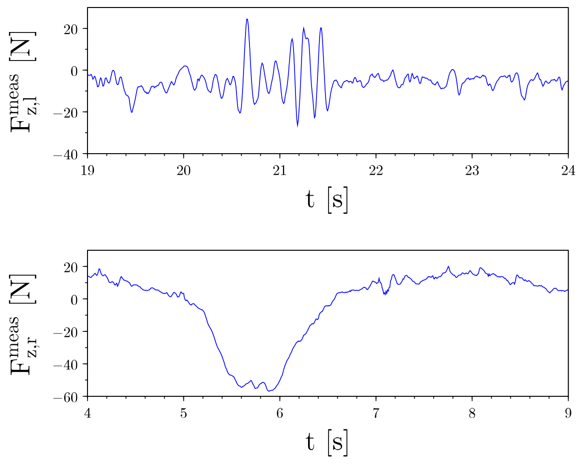
Instead of using a single scaling factor, each sensor and each spatial direction of the measures should be independently calibrated. With the current implementation, when the robot is standing on one foot, the scaled vertical force $F_{z,i}^{scaled}$ is not necessarily equal to the total weight of the robot because of this common scaling factor.

It is though very complex to perfectly calibrate F/T sensors, while they are mounted on the robot, because the latter is not fixed to the ground and cannot stand on one foot without BC. Since this is not the focus of this thesis, the existing calibration has been kept unchanged.

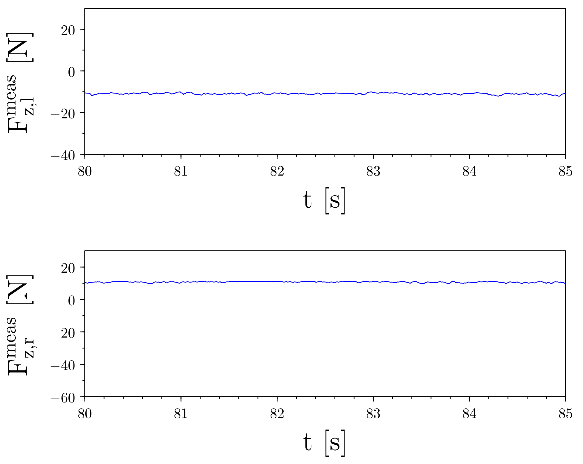
Problems with Analog-to-Digital Converter



(a) Total recording



(b) Zooms on highly disturbed measures (in "torque on" mode)



(c) Zooms on undisturbed measures (in "torque off" mode)

Figure 6.10: Force in z direction measured by the F/T sensors when the robot is hanging in the air (comparison between "torque on" and "torque off" mode)

In "torque off" mode (i.e. the communication with the motors is powered but not the motors themselves), the measures of the sensors have a high quality (almost no noise). On the other hand, in "torque on" mode (i.e. the motors are turned on), perturbations appears and have become worse with the time. Typical (filtered) measures are illustrated in Figure 6.10 in both modes. Zooms were made on the time axis, so the period is of the same order of magnitude as for a walking gait.

In "torque on" mode the error on the measures can amount up to 50 N during almost 1 s, which corresponds to the default duration of a step. Therefore, these discrepancies in the measures can be neither filtered nor neglected. Moreover, reducing the cut-off frequency of the low-pass filters (cf. Section 4.1.1) would delay the measures and slow the BC. This is especially critical during the landing phase. These discrepancies between the two modes are due to the analog-to-digital (A/D) converters in the servomotors of the ankle joints, which are disturbed by the current consumed by the motors. Additionally, the mean value of the signals sometimes suddenly changed without any noticeable reason.

A parallel project was launched to add external converters, but was not finished before the end of this thesis. Later, it was discovered that the smaller and regular oscillations are due to the capacitors of the power supply. By removing them, these oscillations disappeared but the other perturbations remain (like the ones in Figure 6.10b).

6.4.2 Testing the Estimation of the ZMP Position

As already mentioned in Section 2.1, when the robot is standing, the real ZMP and the projected CoM coincide. This property is used to test the accuracy of the ZMP estimation. An experimentation in the simulation was not possible, since the F/T measures were wrong (cf. Section 6.1). Indeed, the estimated ZMP was often situated meters away from the robot, although it was standing.

For the real tests, the CoM is first centered above the feet as described in Section 6.3.2 and then horizontally moved by changing the CoB offsets. In Figure 6.11 and Figure 6.12, the following tests are illustrated:

- Test 1: CoM displacements in x direction (forwards and backwards)
- Test 2: CoM displacements in y direction (to the left and to the right)
- Test 3: CoM displacements in x+ direction until the robot falls over
- Test 4: CoM displacements in y- direction (i.e. to the right) until the robot falls over

As shown in these figures, the estimated ZMP follows the curve of the commanded position of the CoM as expected. In test 3, the robot fell over at $t \approx 70\ s$ and had to be caught, creating additional forces on the F/T sensors, which explains the peak in the ZMP estimation in Figure 6.12. It was not possible to complete test 4, since the right knee pitch servomotor automatically turned off at $t \approx 50\ s$ because of the torque (i.e. current) limitation. Indeed, the robot was almost standing on one single foot and the efforts in the joint were too high for the motor. This problem was already known in the SIM lab and described in the thesis of Stein [36]. The same phenomenon occurred by trying to move the CoM backwards until the robot falls over.

Tests 1 and 3 were conducted first. When the robot fell over, it had to be lifted in the air before putting in down again. Through this process, the joints take a slightly different position every time because of the stress and the backlash. Because of the high position of the CoM, this has certainly an influence on its horizontal position and could explain why the estimated ZMP is not situated at the same x coordinate at the beginning of tests 2 and 4. However, it is not disturbing, since the CoM was only moved in the y direction. The oscillations in x direction in Figure 6.11 are due to the oscillations of the robot itself, especially in x direction because of the backlash in the joints and the brutal stop in the command of the CoB position.

After the CoM is stabilized, the error between the curves of the estimated ZMP and the desired CoM is generally still about 5 mm and about 1 cm near the limits of the support polygon. This can be explained with the following facts:

- The CoM is approximated by the position of the CoB and offset values, so the dashed lines in Figure 6.11 and Figure 6.12 do not exactly correspond to the real position of the CoM.

- The model of the robot used in the IK is not perfect and this also adds errors in the positioning of the CoM.

- The imprecision of the calibration of the F/T sensors (cf. Section 6.4.1) and the errors in the measures of the F/T sensors (cf. Section 6.4.1) are duplicated in the estimated ZMP, since the computation is based on several measures (cf. equations (6.8) and (6.9)).

- The deformation of the feet rubber sole (when the ZMP is near an edge of the foot) affects the measurements and the estimation.

Moreover, the estimated ZMP has more noise in y direction because in the initial pose the positions of the left and right feet relative to the robot frame verify

$$x_{robot\to lf} = x_{robot\to rf} \ (= 0) \tag{6.6}$$
$$y_{robot\to lf} = -y_{robot\to rf} \ (= 0.093\ m) \tag{6.7}$$

which transforms equation (D.1) into

$$p_x^{est} = \hat{p}_x^{est} = -\frac{\tau_{y,l}^{meas} + \tau_{y,r}^{meas}}{F_{z,l}^{meas} + F_{z,r}^{meas}} \tag{6.8}$$
$$p_y^{est} = \hat{p}_y^{est} = \frac{y_{robot\to lf} \cdot (F_{z,l}^{meas} - F_{z,r}^{meas}) + \tau_{x,l}^{meas} + \tau_{x,r}^{meas}}{F_{z,l}^{meas} + F_{z,r}^{meas}} \tag{6.9}$$

Therefore, the noise of the F/T measures is added two more times in the y direction as in the x direction.

The computation of the estimated ZMP on a VS has also been tested on the real robot. The VS grade was arbitrarily set to 50%. The robot was held by the front handles while standing in the initial pose and pushed in the x and y directions to manually move the ZMP as illustrated in Figure 6.13. The ZMP sometimes goes outside the support polygon because the robot turned around the edges of the feet but did not fall over since it has been continuously held. The 2D representation in the x-z plane in Figure 6.14 confirms that the estimated ZMP is moving on the VS.

To conclude, it was shown in this section that the calculation of the estimated ZMP is correct. Nevertheless, the precision should be improved for control purposes. This can be done by removing the perturbations in the measured F/T signals as explained in Section 6.4.1.

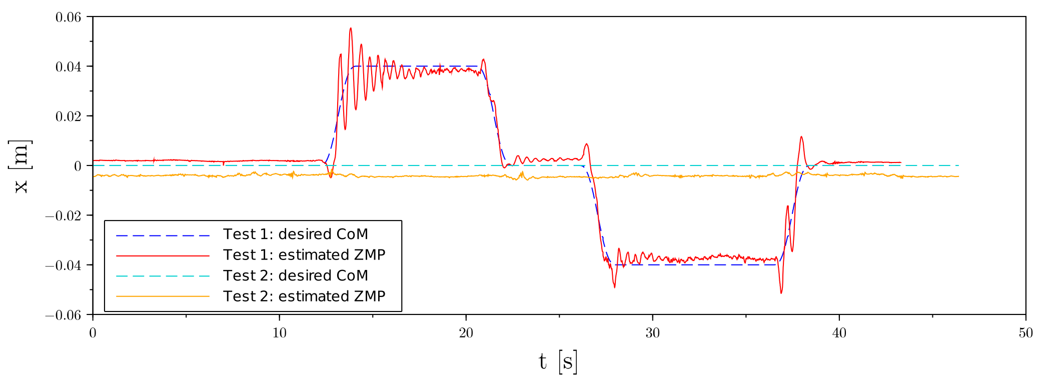
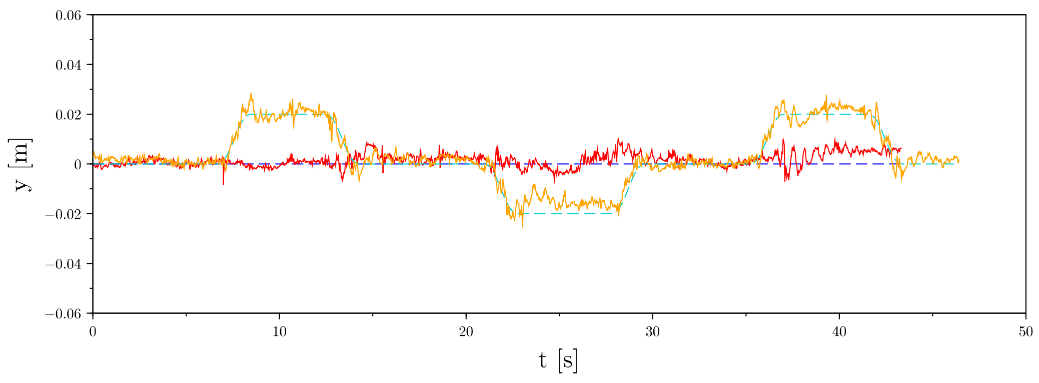


Figure 6.11: Tests 1 and 2 of the ZMP estimation

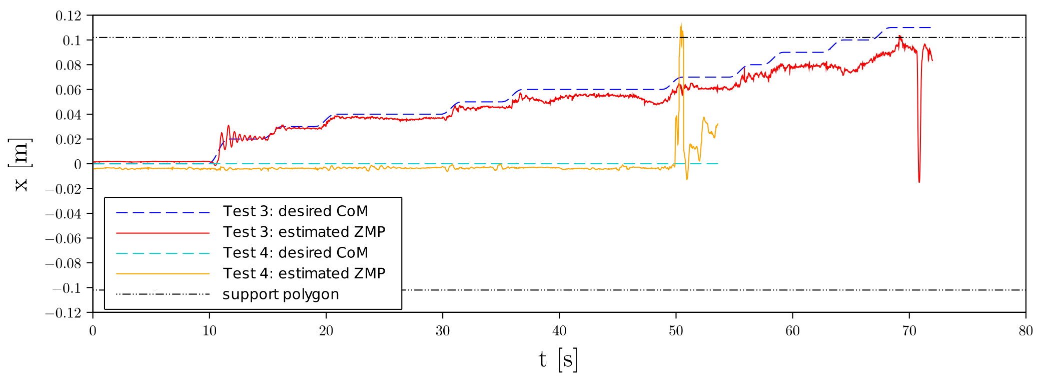
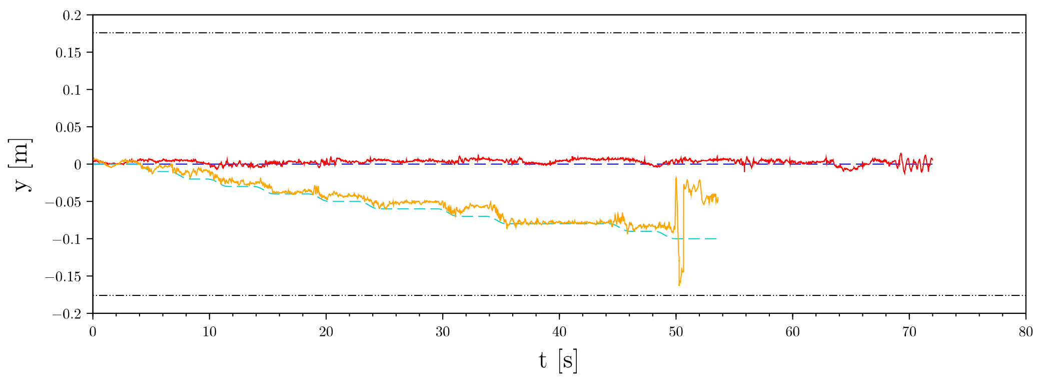


Figure 6.12: Tests 3 and 4 of the ZMP estimation

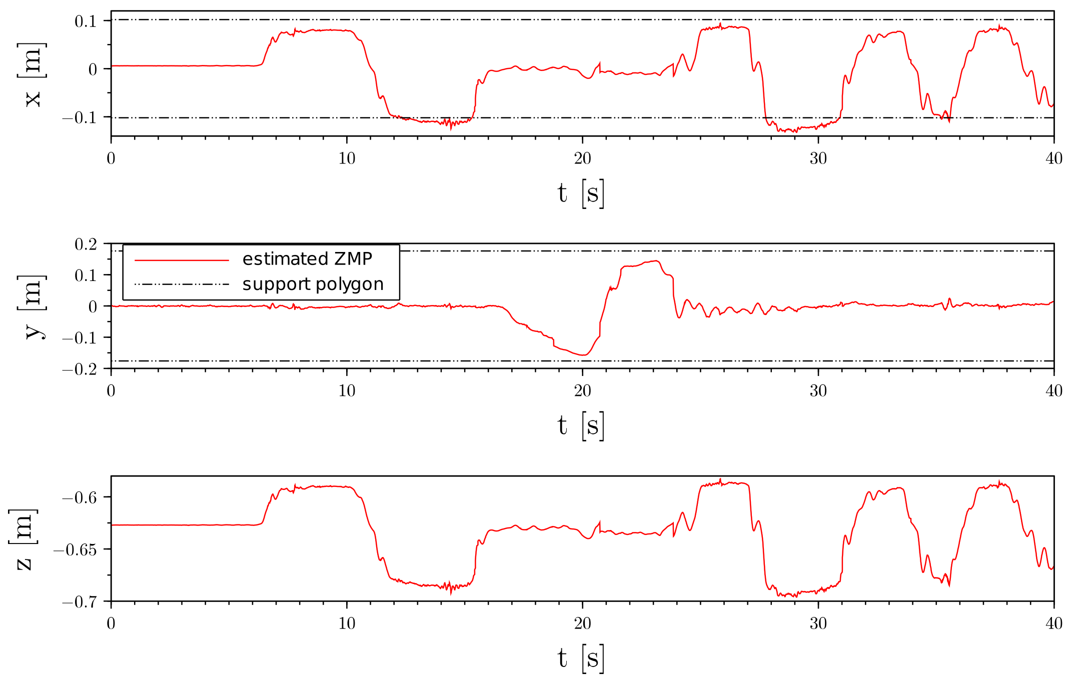


Figure 6.13: Position of the estimated ZMP on a VS of 50% relative to the time during a push experiment

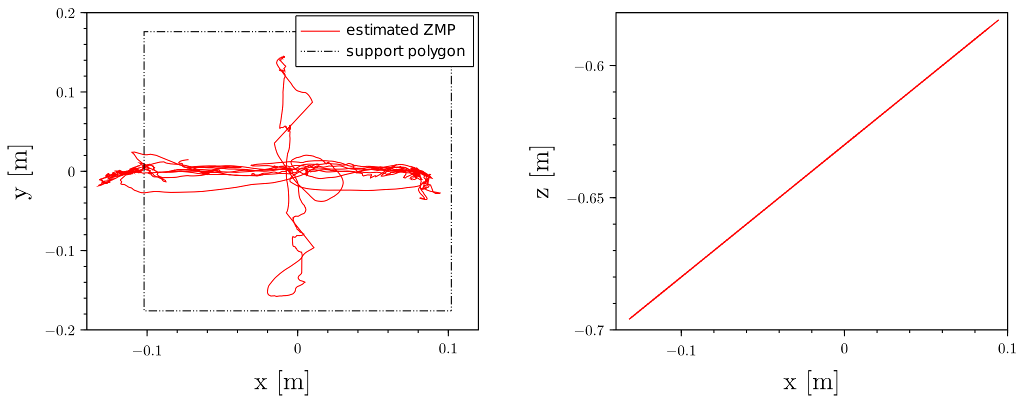


Figure 6.14: 2D representations of the position of the estimated ZMP on a VS of 50% relative to the time during a push experiment

6.4.3 Tuning the Gains of the Controllers

As already explained in Section 2.3.2, there is no general method for tuning the gains of the BC. When no model (local or global) is available, they are experimentally determined by trial and error, as done by Robotis for THORMANG3 (but the gains had to be refined on Johnny #5). For the tuning a long experience with the robot is necessary to "feel" how to modify the gain to improve the biped locomotion. The main goal of the 2D visualization tool (presented in Section 5.3) is to ease this process by enabling a better evaluation of the global performances of the robot and a comparison of the signals of different experiments. This tool was already useful for tuning and validating the parameters of the GPG (cf. Section 6.3).

Ziegler et al. [37] were among the firsts to propose heuristic rules to tune the gains of a PID controller. It works as follows: The I and D gains (k_i and k_d) are set to zero and the P gain (k_p) is increased until the system oscillates. The period P_u is measured and the corresponding value of the gain is named S_u. The heuristic Ziegler–Nichols method sets the gains to the following values

$$k_p = 0.6 \cdot S_u \tag{6.10}$$

$$k_i = k_p \frac{2}{P_u} \tag{6.11}$$

$$k_d = k_p \frac{P_u}{8} \tag{6.12}$$

Other similar heuristic methods have been found in the literature, but none for PD controllers. In any case, such a method is not appropriate for a humanoid robot, because deliberately creating oscillations can be very dangerous, especially for the robot itself.

The manual tuning of the gains is a very long and complex process, thus it was not carried out during this thesis. In fact, it is much more difficult to tune the gains of the new controllers than refining gains that were already tuned like it was done on Johnny #5 based on the values fixed by Robotis for THORMANG3. This is principally due to the fact that the controllers cannot be tested in the simulation as explained in Section 6.1. Indeed, a working simulation could enable to test tuning methods on the virtual robot. Even if the gains must be tuned again on the real robot because of the modeling discrepancies, this would permit to speed up the process and make it safer.

The difficulty in determining the P and D gains is based on the fact that these gains cannot be adjusted independently. Indeed, one idea could be to increase the gain P until a quick response with little or no overshooting is achieved, then increase the gain D to further improve the reactivity of the response. But this would not lead to the best possible performances. In fact, with the combination of the P and D gains (actually two P gains as explained in Section 4.1.2), Robotis' IMU control is stable, but as experimentally tested on Johnny #5, it is instable when the D gain is set to zero (the ankles quickly oscillated during the test). This means that the proposed procedure would not have lead to the gains found by Robotis. Generally, higher gains permit to make the system quicker and reduce the steady error. Nevertheless, too high gains increase the chance for the system to become unstable. A good procedure is probably to set the P gain, then the D gain (as proposed previously), and then try to increase both gains in parallel and gradually.

Although the BC is designed as decoupled controllers, they have an influence on each other. Thus, it is difficult to tune them separately. For instance, the terrain adaptation control and the IMU control act both on the correction of the rotation of the foot. This can partially be overcome by designing test experiments that target one specific control loop. Some propositions are listed below (the robot is standing in the initial pose):

- Pushing on the robot to test the IMU and the ZMP control loops
- Putting the robot down on a small slope to test the IMU control loop
- Putting the robot down on locally uneven terrain to test the terrain adaptation controllers (for instance placing a small plate under one foot)

Once the robot can stand stably, the landing controllers could be tested by walking on place such that the feet are only moving vertically. The gains could then be refined on a default walking gait and later by walking on uneven terrain.

7 Evaluation

In this section, the implemented concepts are tested on different walking scenarios. As explained in Section 6.4.3, the tuning of the gains could not be achieved in this thesis, so the new GPG was tested with Robotis' BC on the real robot for a default walking gait (cf. Section 7.2). Before starting a walking experiment, the robot is prepared as detailed in Section 7.1. For walking on uneven terrain like stairs, the footstep length must be increased. With Robotis' BC and fixed gait parameters, the robot was not able to make these big steps (cf. Section 7.3). Thus, the walking on stairs has only be tested in the simulation as presented in Section 7.4. This permitted to validate the proposed VS method and Swing Foot Trajectory Generator.

7.1 Before Walking

In Johnny #5's software a *supervisor* Graphical User Interface (GUI) is provided, which enables to switch between different modes. The robot is moved to the initial pose by selecting the *stand_prep* mode in the GUI. The *stand* mode permits to start the rotation of the scanning laser and the *walk* mode launches the walking software.

Before each walking experiment, the condition of the robot should be checked. The different stages of this process are listed below:

1. Calibration check of the joints of the legs in *stand_prep* mode as described in Section 6.3.1 using the ROS tool *dynamic_reconfigure*[1]. The joints often need to be re-calibrated after a walking experiment. The causes are not clear, but it is possibly due to the backlash in the joints.

2. Check of the soldering of the F/T boards in *stand* mode. The soldering of the cables on the boards are very fragile. Because of the vibrations in the feet during walking, cracks appear in the soldering, which can strongly disturb the measurements. To check the soldering, the cables are gently moved, while plotting the measures (for instance using the ROS tool *rqt_multiplot*[2]). If the curves suddenly oscillate with a large amplitude, the plugs must be soldered again.

3. Calibration check of the F/T sensors in *stand* mode as described in Section 6.4.1. Because of the errors in the vertical forces measurement (cf. Figure 6.10), it is difficult to calibrate the F/T sensors. In fact, when doing the calibration twice in a row, the difference between the first and the second scaled signal of the vertical forces measurement can be up to 50 N (cf. Section 6.4.1).

4. Check of CoB offsets in *walk* mode as described in Section 6.3.2. The BC must be deactivated for the calibration and the 2D visualization tool is used to check the position of the ZMP.

7.2 Default Walking Gait on Flat Ground

Robotis' default walking gait correspond to a straightforward motion on flat ground with a step length $\Delta X_{step} = 0.1\ m$ and a step duration $T_{step} = 1\ s$. As already mentioned, the real tests are made with

[1] The *dynamic_reconfigure* tool permits to dynamically change predefined software parameters. See `http://wiki.ros.org/dynamic_reconfigure` for more information.

[2] `http://wiki.ros.org/rqt_multiplot`

Robotis' BC. Snapshots of the walking with the new GPG can be found in Appendix F Figure F.1. The results obtained with the former and the new GPG are already presented in Section 6.3.3. In brief, the margin between the estimated ZMP and the outer edges of the feet is increased, making the motion in the y direction more stable. Nevertheless, during the DSP the feet do not always remain in contact with the ground. This can certainly be overcome by tuning the proposed landing and ZMP control loops.

7.3 Big Steps

For walking on uneven terrain like stairs, the robot needs to be capable of making steps of a length bigger than the dimension of its feet ($l_{foot} \approx 0.22\ m$ for THORMANG3). Thus, it was decided to test a walking gait with a step length of $\Delta X_{step} = 0.25\ m$ in order to have some margin. In Robotis' code, this parameter can be changed by the footstep planner, but not the step duration which is fixed to $T_{step} = 1\ s$. However, for big steps this duration should be increased, otherwise the foot is moving too fast, i.e. the joint velocities are too high. On the other hand, if the step duration is too long, the ZMP may go out of the support polygon (making the robot falling over) because the CoM acceleration may be too small to compensate its position outside the support polygon (cf. ZMP equation (2.3)).

The step duration should be computed by the GPG depending on the step distance. This could be done by automatically checking the velocities of the Swing Foot Trajectories by the generation and increasing the step duration if they are too high. This means that the range of acceptable velocities must be determined. Alternatively, a relation between the step duration and length may be found. Moreover, a walking gait with bigger steps needs a more robust BC because the LIPM makes larger angles with the vertical producing higher torques in the ankle joints. These propositions could not be investigated in this thesis for the reasons explained in Section 6.4.3, but finding a solution for taking big steps certainly is the next objective in the development of the walking software.

Because the real robot can not make big steps, it was not possible to test the new implementation on stairs or slopes. However, in the simulation the robot could make big steps with a duration of $T_{step} = 1\ s$, since it can walk without BC and the velocities and acceleration are not limited. Thus, as presented in the next sections, the new GPG has been tested on stairs and slopes in the simulation.

7.4 Stairs

As in [19, 23] the stairs were chosen as test case for walking on uneven terrain in the simulation. In fact, it enables to test the implemented VS method and collision avoidance at the same time. With the new GPG the robot can walk up stairs of a height of 5 cm in the simulation as illustrated in Appendix F Figure F.3. This was not possible with Robotis' GPG because the CoM could only move horizontally (no vertical adaptation to the terrain, cf. Section 3.5) and nothing was implemented to avoid collisions. In the new code, the VS is automatically computed by the GPG and the bypass points are manually passed because they are not implemented in the footstep planner yet. This collision avoidance works as already shown in Section 3.4.2 and illustrated in Figure 7.2. The CoB and feet trajectories generated by the former and the new GPG for stairs of 5 cm are plotted in Figure 7.1. In the second case, the swap motion of the CoB was deactivated to better visualize the implemented motion on the VS. In fact, as illustrated in the lower plot of Figure 7.1, the CoB moves in a plane parallel to the stairs with the new GPG. In the upper plot, its trajectory is not linear with respect to the time since the x position of the CoB does not evolve linearly with respect to the time. With the former implementation, only the feet are going up, making the CoB loosing height above the feet which is an undesirable effect.

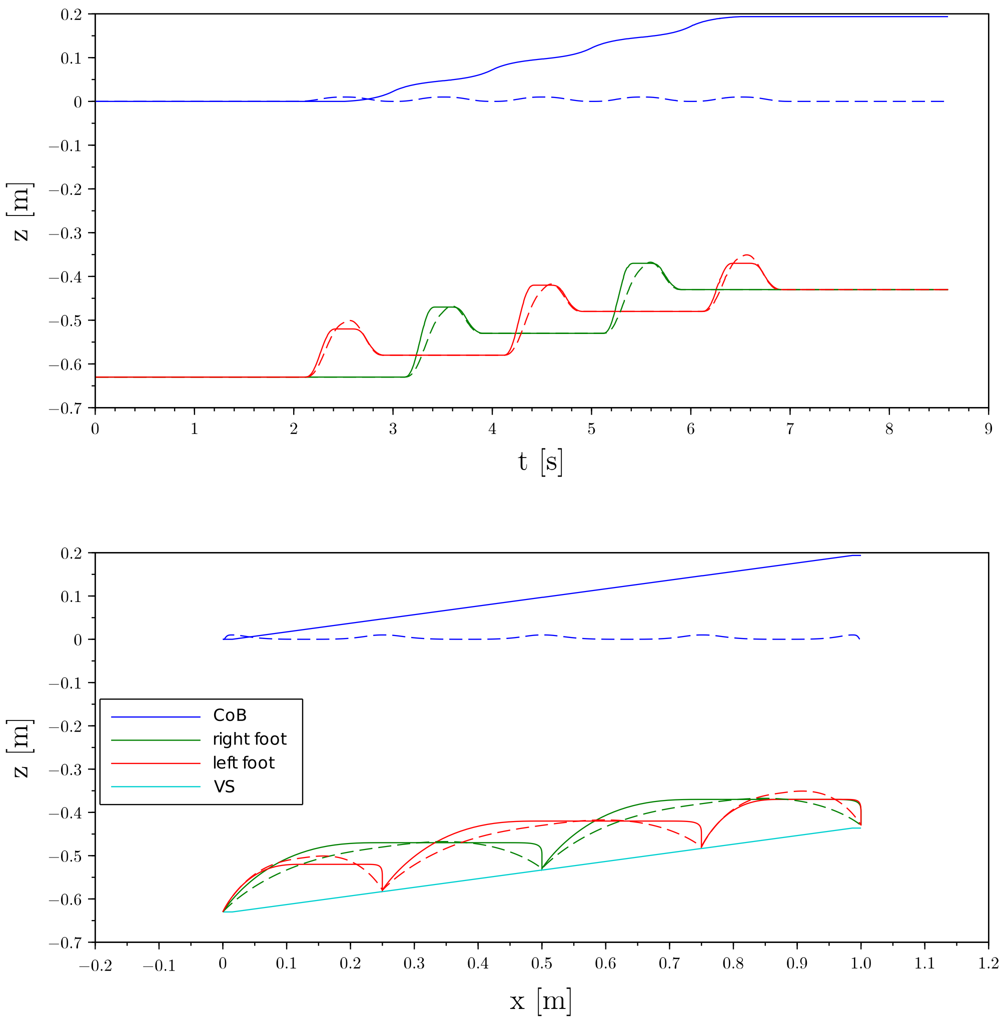


Figure 7.1: CoB and feet trajectories generated by the GPG for walking on stairs of 5 cm (without bypass points). The dashed lines correspond to Robotis' GPG and the continuous lines to the new GPG without swap motion of the CoB.

For the real robot, it would probably be better to lean the upper body forwards in order to reduce the torques in the knees when the feet are on different stairs as in first snapshot of Figure 7.2. This may prevent them from shutting down because of the torque limitation (cf. Section 6.4.2). Nevertheless, THORMANG3 could probably not walk up stairs designed for humans (The standard height is about 17 cm.[1] because the knees are already overstretched on stairs of 10 cm (cf. Figure 7.2). A solution could be to bring the CoB down, but this would increase the torques in the knees because they are more bent. Moreover, the simulated stairs have a length of 25 cm, against 28 cm for standard stairs[1]. This means that the steps should be even bigger which is not possible with the already overstretched knees. A solution could be to make an additional small step (of 3 cm) on each stair.

[1] http://www.spreng-gmbh.de/documents/spreng/Plakat_Treppenmasze.pdf

The new GPG was also tested on a plateau (illustrated in Figure 7.4) to test the VS transition between flat ground and stairs. In Figure 7.3 the generated CoB and feet trajectories are plotted (again, without swap motion to visualize the VS). The robot was able to walk on the plateau in the simulation.

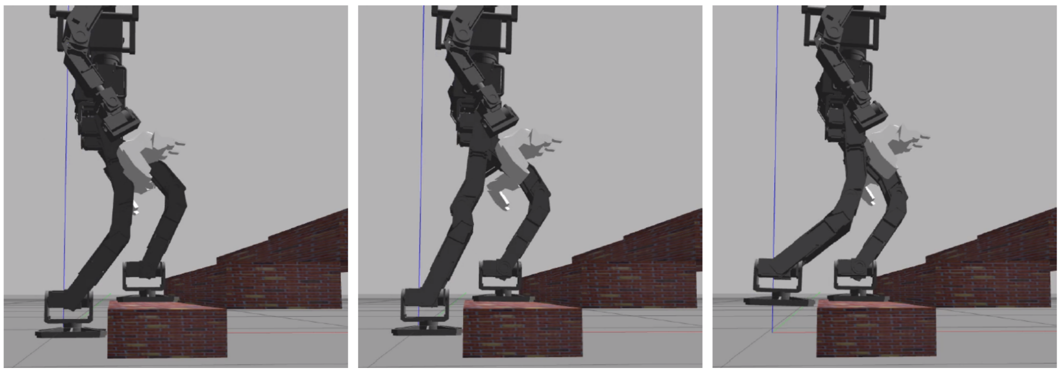


Figure 7.2: Overstretched knees on a stair of 10 cm in the simulation

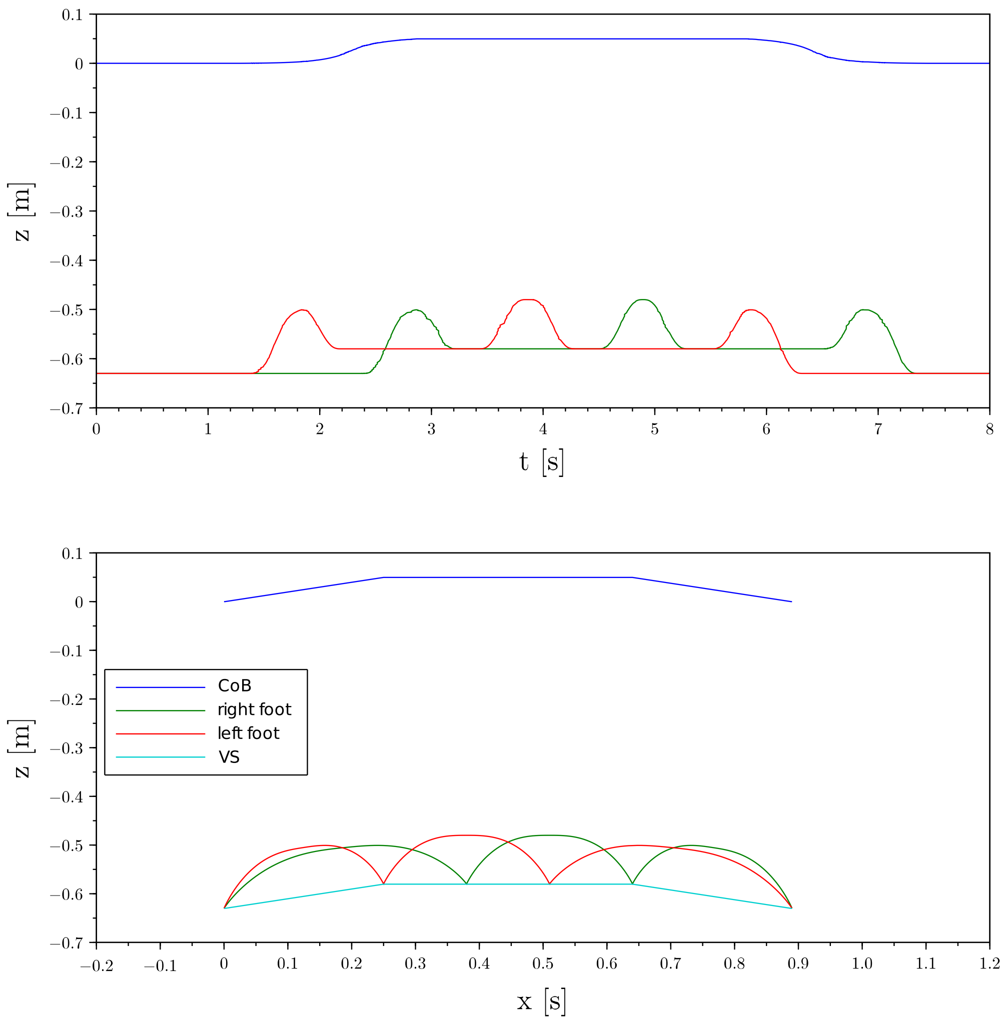


Figure 7.3: CoB and feet trajectories generated by the new GPG (without swap motion of the CoB) for walking on a plateau of 5 cm

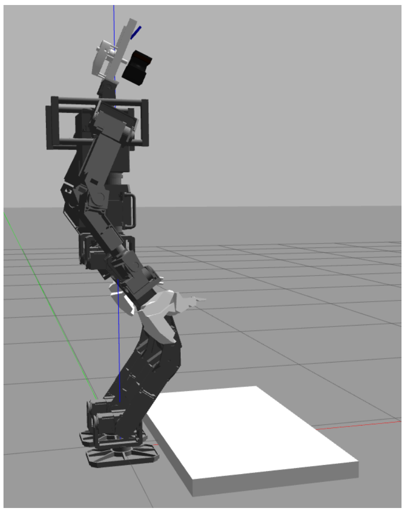


Figure 7.4: Test case walking on a plateau of $5\ cm$

8 Conclusion

Section 8.1 resumes the achievements of this thesis. Suggestions for improvement and development in the short and long term are proposed in Section 8.2,.

8.1 Achievements

Before the beginning of this thesis the humanoid robot Johnny #5 of the SIM lab was able to dynamically walk on flat ground, however not on uneven terrain. In a former thesis for the SIM lab, it has been tried to use the Drake toolbox developed by the MIT to improve its legged locomotion. Thus, in the present thesis it was first investigated why the method has led to unsuccessful results back then. The principal reason why it was then decided not to continue with Drake is that the toolbox was designed for robots disposing of torque control at the joint level while Johnny #5 is position-controlled.

Thus, the existing software has been studied and it turned out that it was based on the well-spread method of ZMP Preview Control. However, only the basic version was implemented, which do not enable a vertical motion of the CoM. This explains why Johnny #5 was able to walk on flat ground only. ZMP control concepts to extend this method were then investigated to enable robust walking motions on uneven terrain like stairs. The literature research was divided into two parts based on the two main blocks of the walking software: the Gait Pattern Generator, which is responsible for generating the desired trajectories of the CoM and the feet, and the Balance Control, which is responsible for maintaining the dynamic balance of the robot by making it follow the desired trajectories.

In this work, the ZMP Preview Control implemented in the GPG was extended with a unique method using Virtual Slopes to generate a vertical motion of the CoM adapted to the terrain. Moreover, a Swing Foot Trajectory was implemented, which disposes of the two further features: vertical landing and collision avoidance using the knowledge of the terrain (furnished by the already implemented Footstep Planner). These extensions led to successful results on stairs in the simulation.

On the real robot however, a robust ZMP Balance Control is additionally required to make the robot stably walk on uneven terrain. A new combination of state of the art balance controllers was implemented to overcome the problems encountered with the former BC: poor stabilization of the roll inclination of the robot and large impact at the landing of the swinging foot. To remedy this, a ZMP controller based on the computation of the virtual ZMP on the Virtual Slope and a landing controller were proposed. In order to test the new software on the real robot with as little risk as possible, safety features were implemented, which aim to stop the motion when failures are detected. For helping assess the performances of the implementation and tune the gains of the controllers, a 2D Visualization Tool was implemented that enables to easily plot all the important signals.

8.2 Outlook

In this section, improvements on the short and long term of the software developed in this thesis are proposed. In the continuity of this work, the GPG parameters should be tuned to enable the real robot making bigger steps on flat ground. This is a necessary stage before developing the walking on uneven terrain like stairs. Moreover, the gains of the new Balance Controllers have to be tuned, which may be a theme for a future thesis in the SIM lab. To ease and accelerate this complex process, another axis of work would be to improve the simulation, such that the BC could be tested on the simulated robot first. This would enable to test tuning methods and reduce the risks of damages on the real robot. To improve the simulation, the robot model and the simulator physics parameters should be tuned to better fit the reality.

On the long term, some other state of the art extensions could be considered. Considering the GPG, the Three-Mass Model [25] could be implemented and compared with the actual LIPM by assessing its effect on the robustness of the locomotion. Moreover, an additional swinging movement of the arms could be added to improve the balance [11, p. 90]. When the foot is not entirely in contact with the ground, the ZMP reference should be adapted to be in the middle of the real support polygon (which is computed by the Footstep Planner) and not of the foot sole. The Swing Foot Trajectory could also be optimized regarding to the speed and torques at the joint level [38].

Considering the BC, the precision of the model for the IK should be tested and improved if necessary. Since the controllers of the horizontal torques and the controllers of the upper body inclination both correct the rotation of the feet, their coupled influence could be considered by combining them in a cascade control loop [34]. To reduce the landing impact even more, the gains of the position controllers of the joints could be reduced at the landing detection [11, p. 94]. When the perturbations are too strong, the BC cannot stabilize the robot any longer and a method of capture step is supposed to be used to avoid a fall [5, 6]. While the capture steps proved to be efficient on flat ground, this is a real challenge on uneven terrain.

A Torque Control using Drake (MIT)

The Robot Locomotion Group at the MIT[1] Computer Science and Artificial Intelligence Lab (CSAIL) developed a project named "Drake" [39], whose aim is to provide tools for "planning, control and analysis" of nonlinear dynamical systems. They have especially developed algorithms to control the locomotion of Atlas, the humanoid robot of Boston Dynamics [40].

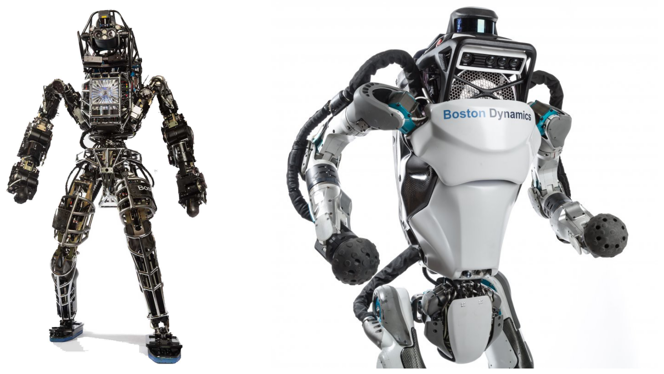


Figure A.1: Boston Dynamics Atlas humanoid robot (former and new version) [photo credits Boston Dynamics]

In a master thesis for the SIM lab [36], Drake was successfully used as whole-body motion planning for manipulation tasks (i.e. without control). Later, Reimold [5] has tried to make THORMANG3 walk on uneven terrain with the help of the MIT algorithms. The optimization for the control of a biped locomotion implemented in Drake returns torque and velocity commands, which are inputs of the joint controllers of Atlas [40]. Since the DYNAMIXEL controllers take a position as input, the idea of Reimold was to only use the velocity command and integrate it to obtain the position [5]. This method did not work, probably because the optimization is made respectively to the torques and not to the positions of the motors. Indeed, it can be set as "torque only control" or as "torque and velocity control". In the first case, the algorithm makes the legs compliant and in the second case, the performances of the position tracking are improved at the cost of the compliance [5].

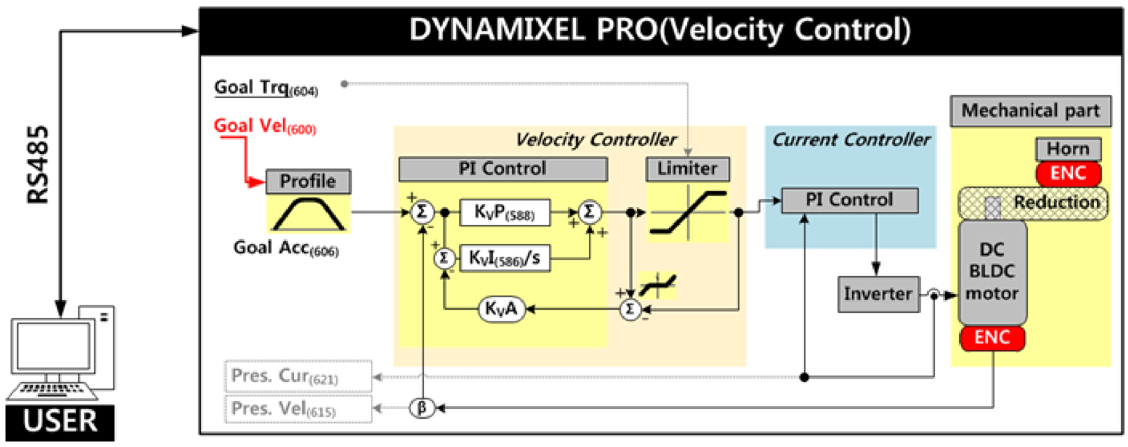
The MIT has successfully used a similar idea for controlling the arms of Atlas[2] but has never tried on the legs. Position-only control may have worked for the arms because they are freely moving, i.e. without external forces applied on them (except the gravitation forces). On the contrary the kinematics chain of legs on the ground is closed. Hence, the contact forces and torques between the feet and the ground have a large influence on the behavior on the system and certainly have an influence on the optimized torques. Moreover, when walking, the legs need some compliance at the landing phase, which is not possible without considering the torque control part of the algorithm.

Del Prete et al. confirmed this supposition that the Drake method cannot be directly applied to robots like THORMANG3, since they wrote: *"Torque control is a necessary requirement for the implementation of rigid-body inverse-dynamics control"* [9, p. 2]. Hence, they have developed a method to use algorithm based on inverse dynamics (like Drake) on humanoid robots having position controlled servomotors and have tested it on the platform HRP-2 [41]. In this method, a good model of the servomotors is necessary

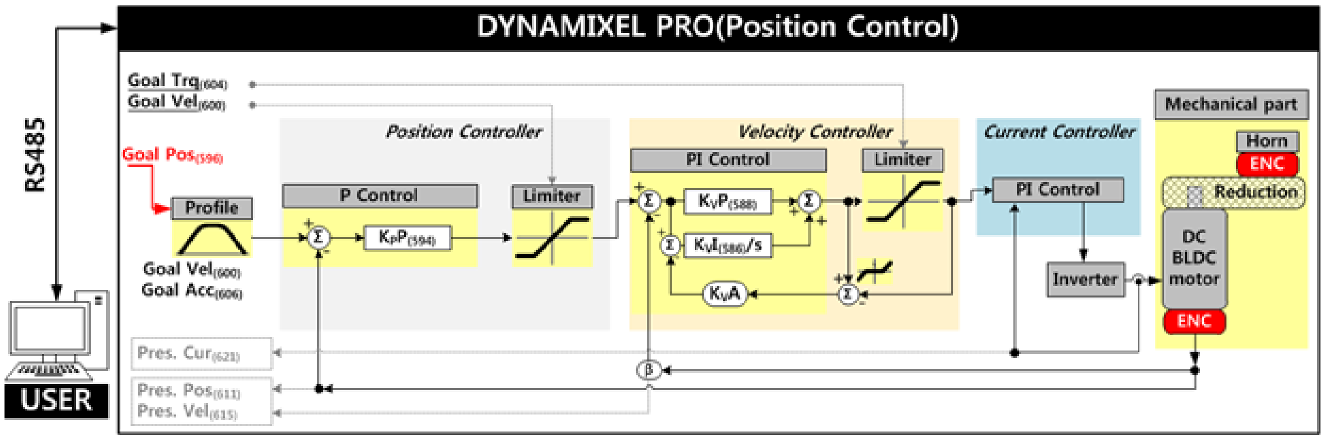
[1] Massachusetts Institute of Technology
[2] cf. Reimold's issue on GitHub: `https://github.com/RobotLocomotion/drake/issues/1233`

to derive the position commands and estimate the torques in the joints. In her master thesis for the SIM lab, Schumacher [42] however showed that it is very complex to establish a precise model of the DYNAMIXEL PROs and thus to estimate the load torque.

Another solution to use Drake could have been to set the DYNAMIXEL PROs to current control (which is almost torque control). According to the online documentation of the DYNAMIXEL PROs [43], the motors are meant to be used in position or velocity control mode, but they can also be set in current control mode (cf. Figure A.2). This last mode was investigated by Schumacher and she has showed that the dynamic performances are low [42, p. 75], making this mode inadequate for THORMANG3. For these reasons, it has been decided not to use Drake in this thesis.



(a) Velocity control mode



(b) Position control mode

Figure A.2: Control modes of DYNAMIXEL PRO servomotors [image credits Robotis [43]]

B Mathematical Proofs of Monotonic Trajectories

Lemma 1. *A fifth order polynomial function*

$$x(t) = a_0 \cdot t^5 + a_1 \cdot t^4 + a_2 \cdot t^3 + a_3 \cdot t^2 + a_4 \cdot t + a_5, \quad t \in [t_0, t_1] \tag{B.1}$$

with the six boundary conditions

$$\begin{cases} x(t_0) = x_0 & \text{(B.2a)} \\ \dot{x}(t_0) = 0 & \text{(B.2b)} \\ \ddot{x}(t_0) = 0 & \text{(B.2c)} \\ x(t_1) = x_1 \neq x_0 & \text{(B.2d)} \\ \dot{x}(t_1) = 0 & \text{(B.2e)} \\ \ddot{x}(t_1) = 0 & \text{(B.2f)} \end{cases}$$

is strictly monotone on the interval $]t_0, t_1[$.

Proof. Without loss of generality we assume $x_0 = 0$, $x_1 = \Delta X > 0$, $t_0 = 0$ and $t_1 = \Delta T > 0$ and so we will prove that the function x is strictly increasing on $]0, \Delta T[$, i.e. that $\dot{x}(t) > 0, \forall t \in]0, \Delta T[$.

The boundary conditions (B.2a), (B.2b) and (B.2c) trivially lead to $a_5 = a_4 = a_3 = 0$. The remaining boundary conditions (B.2d), (B.2e) and (B.2f) can be rewritten as follows:

$$\begin{cases} a_0 \cdot \Delta T^5 + a_1 \cdot \Delta T^4 + a_2 \cdot \Delta T^3 = \Delta X \\ 5a_0 \cdot \Delta T^4 + 4a_1 \cdot \Delta T^3 + 3a_2 \cdot \Delta T^2 = 0 \\ 20a_0 \cdot \Delta T^3 + 12a_1 \cdot \Delta T^2 + 6a_2 \cdot \Delta T = 0 \end{cases} \tag{B.3}$$

Using the transformation

$$\begin{cases} \alpha = \frac{a_0 \cdot \Delta T^5}{\Delta X} \\ \beta = \frac{a_1 \cdot \Delta T^4}{\Delta X} \\ \gamma = \frac{a_2 \cdot \Delta T^3}{\Delta X} \end{cases} \tag{B.4}$$

the system (B.3) becomes

$$\begin{cases} \alpha + \beta + \gamma = 1 \\ 5\alpha + 4\beta + 3\gamma = 0 \\ 20\alpha + 12\beta + 6\gamma = 0 \end{cases} \tag{B.5}$$

and its solution is

$$\begin{cases} \alpha = 6 \\ \beta = -15 \\ \gamma = 10 \end{cases} \tag{B.6}$$

For all $t \in]0, \Delta T[$ we have:

$$\begin{aligned}
& \dot{x}(t) > 0 \\
\Longleftrightarrow\ & 5a_0 \cdot t^4 + 4a_1 \cdot t^3 + 3a_2 \cdot t^2 > 0 \\
\Longleftrightarrow\ & 5a_0 \cdot t^2 + 4a_1 \cdot t + 3a_2 > 0 \\
\Longleftrightarrow\ & 5a_0 \cdot t^2 \cdot \frac{\Delta T^5}{\Delta X} + 4a_1 \cdot t \cdot \frac{\Delta T^5}{\Delta X} + 3a_2 \cdot \frac{\Delta T^5}{\Delta X} > 0 \\
\Longleftrightarrow\ & 5\alpha \cdot t^2 + 4\beta \Delta T \cdot t + 3\gamma \Delta T^2 > 0 \\
\Longleftrightarrow\ & 30 \cdot t^2 - 60\Delta T \cdot t + 30\Delta T^2 > 0 \\
\Longleftrightarrow\ & t^2 - 2\Delta T \cdot t + \Delta T^2 > 0 \\
\Longleftrightarrow\ & (t - \Delta T)^2 > 0
\end{aligned}$$

□

Lemma 2. *A fifth order polynomial function*

$$x(t) = a_0 \cdot t^5 + a_1 \cdot t^4 + a_2 \cdot t^3 + a_3 \cdot t^2 + a_4 \cdot t + a_5, \quad t \in [t_0, t_1] \tag{B.7}$$

has the six boundary conditions

$$\begin{cases}
x(t_0) = x_0 & \text{(B.8a)} \\
\dot{x}(t_0) = 0 & \text{(B.8b)} \\
\ddot{x}(t_0) = 0 & \text{(B.8c)} \\
x(t_1) = x_1 \neq x_0 & \text{(B.8d)} \\
\dot{x}(t_1) = v_1 \neq 0 & \text{(B.8e)} \\
\ddot{x}(t_1) = a_1 \neq 0 & \text{(B.8f)}
\end{cases}$$

such that $(x_1 - x_0)$, v_1 and $(-a_1)$ have the same sign.

There exists $\Delta T_0 > 0$ such that if $(t_1 - t_0) < \Delta T_0$ the function x is strictly monotone on the interval $]t_0, t_1[$.

Proof. Without loss of generality we assume $x_0 = 0$, $x_1 = \Delta X > 0$, $\dot{x}_1 = V > 0$, $\ddot{x}_1 = -A < 0$, $t_0 = 0$ and $t_1 = \Delta T > 0$ and so we will prove that $\exists \Delta T_0 > 0, \forall \Delta T < \Delta T_0, \forall t \in]0, \Delta T[, \dot{x}(t) > 0$.

The boundary conditions (B.2a), (B.2b) and (B.2c) trivially lead to $a_5 = a_4 = a_3 = 0$. The remaining boundary conditions (B.2d), (B.2e) and (B.2f) can be rewritten as follows:

$$\begin{cases} a_0 \cdot \Delta T^5 + a_1 \cdot \Delta T^4 + a_2 \cdot \Delta T^3 = \Delta X \\ 5a_0 \cdot \Delta T^4 + 4a_1 \cdot \Delta T^3 + 3a_2 \cdot \Delta T^2 = V \\ 20a_0 \cdot \Delta T^3 + 12a_1 \cdot \Delta T^2 + 6a_2 \cdot \Delta T = -A \end{cases} \tag{B.9}$$

Using the transformations

$$\begin{cases} \alpha = a_0 \cdot \Delta T^5 \\ \beta = a_1 \cdot \Delta T^4 \\ \gamma = a_2 \cdot \Delta T^3 \end{cases} \tag{B.10}$$

and

$$\begin{cases} c_1 = \Delta X \\ c_2 = V \Delta T \\ c_3 = -A \Delta T^2 \end{cases} \tag{B.11}$$

the system (B.9) becomes

$$\begin{cases} \alpha + \beta + \gamma = c_1 \\ 5\alpha + 4\beta + 3\gamma = c_2 \\ 20\alpha + 12\beta + 6\gamma = c_3 \end{cases} \tag{B.12}$$

and its solution is

$$\begin{cases} \alpha = 6c_1 - 3c_2 + \frac{1}{2}c_3 \\ \beta = -15c_1 + 7c_2 - c_3 \\ \gamma = 10c_1 - 4c_2 + \frac{1}{2}c_3 \end{cases} \tag{B.13}$$

Finally we obtain

$$\begin{cases} a_0 = \frac{1}{\Delta T^5}(6c_1 - 3c_2 + \frac{1}{2}c_3) \\ a_1 = \frac{1}{\Delta T^4}(-15c_1 + 7c_2 - c_3) \\ a_2 = \frac{1}{\Delta T^3}(10c_1 - 4c_2 + \frac{1}{2}c_3) \end{cases} \tag{B.14}$$

For all $t \in]0, \Delta T[$ we have:

$$\begin{aligned} & \dot{x}(t) > 0 \\ \Longleftrightarrow\ & 5a_0 \cdot t^4 + 4a_1 \cdot t^3 + 3a_2 \cdot t^2 > 0 \\ \Longleftrightarrow\ & 5a_0 \cdot t^2 + 4a_1 \cdot t + 3a_2 > 0 \end{aligned} \tag{B.15}$$

The discriminant of the left side of inequality (B.15) is:

$$
\begin{aligned}
Disc_t(\dot{x}) &= 16a_1^2 - 60a_0a_2 && \text{(B.16)}\\
&= \frac{1}{\Delta T^8}(64V^2\Delta T^2 + A^2\Delta T^4 - 120V\Delta X\Delta T - 14AV\Delta T^3) && \text{(B.17)}
\end{aligned}
$$

Thus for $\Delta T \rightarrow 0$

$$Disc_t(\dot{x}) \sim -120\frac{V\Delta X}{\Delta T^7} \tag{B.18}$$

which implies

$$\exists \Delta T_0 > 0, \forall \Delta T < \Delta T_0, Disc_t(\dot{x}) < 0. \tag{B.19}$$

Thus for $\Delta T < \Delta T_0$ the sign of $\dot{x}(t)$ is constant on $]0, \Delta T]$ and corresponds to the sign of $\dot{x}(\Delta T)$ which is positive. Therefore if $0 < \Delta T < \Delta T_0$, the function $\dot{x}$ is strictly increasing on $]0, \Delta T]$. □

Application of Lemma 2 to the Swing Foot Trajectory

For the phase No. 3 of the Swing Foot Trajectory in z direction the coefficients a_0, a_1 and a_2 are computed using the correspondence:

$$
\begin{aligned}
\Delta T &\leftarrow (t_{appro} - t_{max,end})\\
\Delta X &\leftarrow (z_{appro} - z_{max})\\
V &\leftarrow V_{max}\\
-A &\leftarrow A_{appro}
\end{aligned}
$$

In order to obtain a strictly decreasing trajectory on $]t_{max,end}, t_{appro}]$ the parameter $t_{max,end}$ is increased (i.e. $(t_{appro} - t_{max,end})$ is decreased) until one of the following cases is true:

(i) $16a_1^2 - 60a_0a_2 < 0$

(ii) $16a_1^2 - 60a_0a_2 \geq 0$, $a_0 < 0$ and $\max(t_-, t_+) \leq 0$

(iii) $16a_1^2 - 60a_0a_2 \geq 0$, $a_0 < 0$ and $\min(t_-, t_+) > \Delta T$

(iv) $16a_1^2 - 60a_0a_2 > 0$, $a_0 > 0$, $\min(t_-, t_+) \leq 0$ and $\max(t_-, t_+) > \Delta T$

(v) $a0 = 0$, $a1 > 0$ and $r > \Delta T$

(vi) $a0 = 0$, $a1 < 0$ and $r \leq 0$

(vii) $a0 = 0$, $a1 = 0$ and $a2 < 0$

with the roots

$$t_{\pm} = \frac{-4a_1 \pm \sqrt{16a_1^2 - 60a_0a_2}}{10a_0} \tag{B.20}$$

$$r = -\frac{3a_2}{4a_1} \tag{B.21}$$

Indeed in all these cases the function $\dot{x}$ is strictly negative on $]0, \Delta T]$ as shown in Figure B.1. Lemme 2 attests that this methods always leads to a solution.

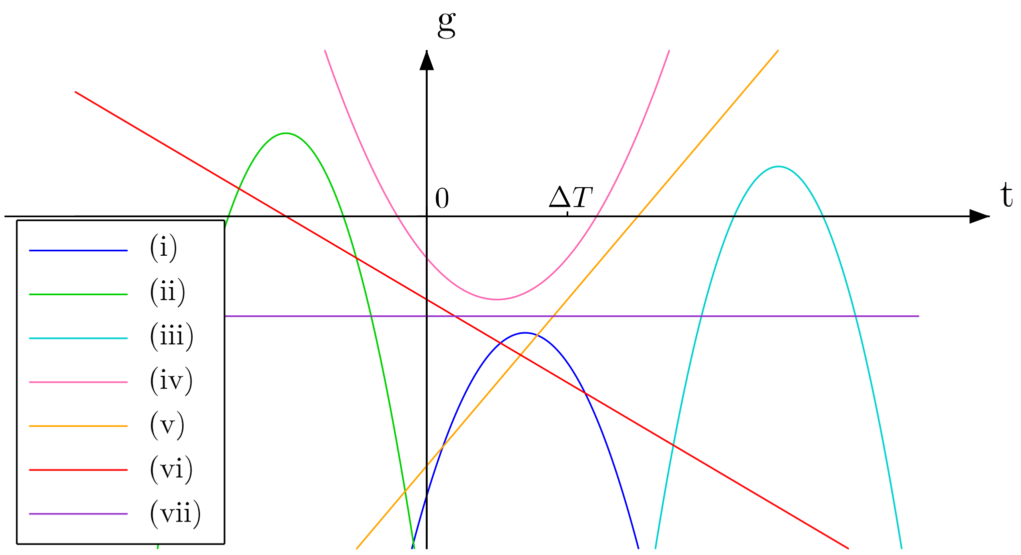


Figure B.1: Cases where the function $g(t) = 5a_0 \cdot t^2 + 4a_1 \cdot t + 3a_2$ is strictly negative on $]0, \Delta T]$

C Pseudo-Code

The Algorithm 1 describes the implemented procedure to generate a Swing Foot Trajectory passing over given bypass points (cf. Section 3.4.2).

Algorithm 1: Swing Foot Trajectory Generator considering bypass points

Data: Start position in 3D, goal position in 3D, list of bypass points

Result: 3D Swing Foot Trajectory passing above the bypass points in x-z plane

```
Generate nominal 3D Swing Foot Trajectory;
foreach BypassPoint in ListOfBypassPoints do
    if $z_{max}$ < BypassPoint.z then
        $z_{max} \leftarrow$ BypassPoint.z;
        Recompute phases No.1, No. 2 and No. 3 of z-trajectory;
    end
    if BypassPoint in phase No.1 then
        Find $t_x$ such that $x(t_x) =$ BypassPoint.x;
        while $z(t_x) <$ BypassPoint.z do
            if (BypassPoint.z $- z_{max}$) is small and $z_{max} < limit$ then
                Increase $z_{max}$ ;
                Recompute phases No. 1 and No. 3 of z-trajectory;
            end
            if $t_{xyrpy} < limit$ then
                Increase $t_{xyrpy}$ and decrease $t_{max,start}$ ;
                Recompute x, y, $\phi$, $\theta$, $\psi$ trajectories and phase No.1 of z-trajectory;
                Recompute $t_x$;
            end
        end
    else if BypassPoint in phase No. 3 or No. 4 then
        Find $t_x$ such that $x(t_x) =$ BypassPoint.x;
        while $z(t_x) <$ BypassPoint.z do
            if (BypassPoint.z $- z_{max}$) is small and $z_{max} < limit$ then
                Increase $z_{max}$ ;
                Recompute phases No. 1 and No. 3 of z-trajectory;
            end
            if $t_{max,end} < limit$ then
                Increase $t_{max,end}$ ;
                Recompute phase No.3 of z-trajectory;
            end
        end
    end
end
```

D Computation of the Estimated ZMP

This appendix contains the mathematics to compute an estimation of the ZMP during a DSP based on the virtual ZMP method of Sato et al. [19] (cf. Section D.1). The mathematics were adapted to fit to our framework. The calculations for the SSP are already presented in Section 4.2.5. In Section D.2 it is shown that computing the ZMP with the virtual ZMP method or with the EZMP method (both presented in Section 2.3.1 and Section 4.2.5) leads to the same result.

D.1 Estimated ZMP at DSP

During a DSP the ZMP position $\boldsymbol{p}^{est}$ relative to the robot frame can be computed using the relation

$$\boldsymbol{p}^{est} \times \boldsymbol{F}_{tot}^{meas} = \boldsymbol{p}_{rf}^{est} \times \boldsymbol{F}_{rf}^{meas} + \boldsymbol{p}_{lf}^{est} \times \boldsymbol{F}_{lf}^{meas} \tag{D.1}$$

where

$$\boldsymbol{F}_{tot}^{meas} = \boldsymbol{F}_{rf}^{meas} + \boldsymbol{F}_{lf}^{meas} \tag{D.2}$$

and the vectors $\boldsymbol{p}_{rf}^{est}$ and $\boldsymbol{p}_{lf}^{est}$ are the ZMP of the right and left foot computed as at SSP (cf. Section 4.2.5). Equation (D.1) can be rewritten as

$$\boldsymbol{p}^{est} \times \boldsymbol{F}_{tot}^{meas} = (\boldsymbol{r}_{robot\to rf} + \hat{\boldsymbol{p}}_{rf}) \times \boldsymbol{F}_{rf}^{meas} + (\boldsymbol{r}_{robot\to lf} + \hat{\boldsymbol{p}}_{lf}) \times \boldsymbol{F}_{lf}^{meas} \tag{D.3}$$

$$= \boldsymbol{r}_{robot\to rf} \times \boldsymbol{F}_{rf}^{meas} + \boldsymbol{r}_{robot\to lf} \times \boldsymbol{F}_{lf}^{meas} + \underbrace{\boldsymbol{\tau}_{rf}^{meas} + \boldsymbol{\tau}_{lf}^{meas}}_{\boldsymbol{\tau}_{tot}^{meas}} \tag{D.4}$$

To avoid unnecessary addition of measured values which causes a more noisy estimation this equation was changed to be relative to the frame S_{rf}^{*}:

$$\hat{\boldsymbol{p}}^{est} \times \boldsymbol{F}_{tot}^{meas} = \hat{\boldsymbol{p}}_{rf} \times \boldsymbol{F}_{rf}^{meas} + (\boldsymbol{r}_{rf\to lf} + \hat{\boldsymbol{p}}_{lf}) \times \boldsymbol{F}_{lf}^{meas} \tag{D.5}$$

$$= \boldsymbol{r}_{rf\to lf} \times \boldsymbol{F}_{lf}^{meas} + \boldsymbol{\tau}_{tot}^{meas} \tag{D.6}$$

The estimated ZMP relative to the robot frame is then

$$\boldsymbol{p}^{est} = \boldsymbol{r}_{robot\to rf} + \hat{\boldsymbol{p}}^{est} \tag{D.7}$$

As in the case of a SSP equation (D.6) and the VS equation (4.24) can be summed up as a matrix equation, so the estimated ZMP at DSP is

$$\boldsymbol{p}^{est} = \boldsymbol{r}_{robot\to rf} + \mathscr{F}^{-1}\mathscr{V} \tag{D.8}$$

where

$$\mathscr{F} = \begin{bmatrix} 0 & F^{meas}_{z,tot} & -F^{meas}_{y,tot} \\ -F^{meas}_{z,tot} & 0 & F^{meas}_{x,tot} \\ k_x & k_y & k_z \end{bmatrix} \quad \text{and} \quad \mathscr{V} = \begin{bmatrix} F^{meas}_{z,lf} \cdot y_{rf\to lf} - F^{meas}_{y,lf} \cdot z_{rf\to lf} + \tau_{x,tot} \\ F^{meas}_{x,lf} \cdot z_{rf\to lf} - F^{meas}_{z,lf} \cdot x_{rf\to lf} + \tau_{y,tot} \\ c_0 \end{bmatrix} \tag{D.9}$$

D.2 Virtual ZMP and Extended ZMP on flat ground

As explained in Section 4.2.5, the computation of the ZMP position using the method of the virtual ZMP on the ZML [19] do not exactly correspond to the definition of the ZMP. Indeed, it considers the torques in each direction to be zero at the ZMP, although only the horizontal torques needs to be zero, as pointed out in [16]. On the contrary, the mentioned method of EZMP [20] is based on the real definition of the ZMP. However, on flat ground both methods lead to the same result as shown below.

ZMP computation on flat ground using the ZML method

In this paragraph, the equations presented in Section 4.2.5 are applied to the case of a SSP on flat ground. The equation (4.22) of the ZML is recalled below:

$$\hat{\boldsymbol{p}}_i \times \boldsymbol{F}_i^{meas} = \boldsymbol{\tau}_i^{meas} \tag{D.10}$$

where $i \in \{rf, lf\}$ depending on which foot is on the ground, $\boldsymbol{F}_i^{meas}$ and $\boldsymbol{\tau}_i^{meas}$ are the forces and torques measured by the F/T sensors and $\hat{\boldsymbol{p}}_i$ is the position vector of the ZMP relative to S_i^* (cf. Section 4.2.5). On flat ground the vertical component of $\hat{\boldsymbol{p}}_i$ verifies

$$\hat{p}_z, i = 0 \tag{D.11}$$

Combining equations (D.10) and (D.11) leads to the system

$$\begin{cases} F^{meas}_{z,i} \hat{p}_{y,i} - F^{meas}_{y,i} \hat{p}_{z,i} = \tau^{meas}_{x,i} \\ F^{meas}_{x,i} \hat{p}_{z,i} - F^{meas}_{z,i} \hat{p}_{x,i} = \tau^{meas}_{y,i} \\ \hat{p}_z, i = 0 \end{cases} \tag{D.12}$$

whose solution is

$$\begin{cases} \hat{p}_{x,i} = -\frac{\tau^{meas}_{y,i}}{F^{meas}_{z,i}} \\ \hat{p}_{y,i} = \frac{\tau^{meas}_{x,i}}{F^{meas}_{z,i}} \\ \hat{p}_z, i = 0 \end{cases} \tag{D.13}$$

ZMP computation on flat ground respecting the ZMP definition

When considering the real definition of the ZMP, only the horizontal torques have to be zero [16]). This consideration lead to the following equation

$$\hat{\boldsymbol{p}}_i \times \boldsymbol{F}_i^{meas} + \begin{bmatrix} 0 \\ 0 \\ \tau_{z,zmp} \end{bmatrix} = \boldsymbol{\tau}_i^{meas} \tag{D.14}$$

where $\tau_{z,zmp}$ is the vertical torque acting at the ZMP. To compute $\hat{\boldsymbol{p}}_i$ the two first equations of (D.14) are considered [16]. Since $\hat{p}_z, i = 0$ is still valid, the same solution (D.13) is obtained.

E Software Parameters

In this appendix, the different software parameters mentioned throughout this thesis are listed with their implemented value. The GPG parameter values in Section E.1 are generally valid for THORMANG3, while the BC parameters have to be specially tuned for the considered robot. Thus, the values listed in Section E.2 are only valid for Johnny #5. The former as well as the new walking software run in real-time with a loop duration of $T_s = 8\ ms$.

E.1 Gait Pattern Generator Parameters

Table E.1: Parameters of Robotis gait and generated trajectories (cf. Section 3)

Param.	**Value**	**Description**
H_{cob}	$0.63\ m$	Height of the CoB
H_{com}	$0.5\ m$	Height of the CoM
m_{total}	$42.533\ kg$	Total mass of THORMANG3 computed in Robotis' software
ΔX_{step}	$0.1\ m$	Step length
T_{step}	$1.0\ s$	Step duration
T_{DSP}	$0.2\ s$	DSP duration
T_{SSP}	$0.8\ s$	SSP duration
ΔZ_{foot}	$0.1\ m$	Amplitude of the swinging complement of the Swing Foot Trajectory
ΔZ_{cob}	$0.01\ m$	Amplitude of the swap motion of the CoB

Table E.2: Parameters of the new gait (cf. Section 3). The question marks correspond to value that have not been tuned yet.

Param.	**Dflt step**	**Big step**	**Description**
H_{cob}	$0.63\ m$	$0.63\ m$	Height of the CoB
H_{com}	$0.7\ m$	$0.7\ m$	Height of the CoM
m_{total}	$48.5\ kg$	$48.5\ kg$	Real total mass of Johnny #5 (measured)
ΔX_{step}	$0.1\ m$	$0.25\ m$	Step length
T_{step}	$1.0\ s$	$?\ s$	Step duration
T_{DSP}	$0.2\ s$	$?\ s$	DSP duration
T_{SSP}	$0.8\ s$	$?\ s$	SSP duration

Table E.3: Parameters of the improved Swing Foot Trajectory with their values (cf. Section 3.4.2)

Param.	Value	Description
Tuned:		
ΔT_{appro}	$0.16\ s$	Period of the vertical approach phase
ΔZ_{min}	$0.10\ m$	Minimal global amplitude of the z trajectory
ΔZ_{max}	$0.30\ m$	Maximal global amplitude of the z trajectory
ΔZ_{appro}	$0.03\ m$	Vertical approach distance for the vertical approach phase
Calculated:		
$\Delta Z_{appro,min}$	$0.06\ m$	Minimal global vertical approach distance of the z trajectory
V_{max}	$0.38\ m.s^{-1}$	Velocity at the beginning of the vertical approach phase
A_{max}	$2.34\ m.s^{-2}$	Acceleration at the beginning of the vertical approach phase

E.2 Balance Control Parameters

Table E.4: Cut-off frequencies of the low-pass filters for the sensor outputs tuned by Robotis (cf. Section 4.1.1)

Sensor	Cut-off freq.
IMU	$12\ Hz$
F/T sensor	$10\ Hz$

Table E.5: Gains of the PD tracking controllers of the joint commands tuned by Robotis (cf. Section 4)

Joint	P gain	D gain
hip yaw	1.0	0
hip roll	1.5	0
hip pitch	0.15	0
knee pitch	0.15	0
ankle pitch	0.05	0
ankle roll	0.05	0

Table E.6: Offset parameters of the CoB (cf. Section 3.1 and Section 6.3.2). The question mark correspond to values that are manually tuned after each new joint calibration as explained in Section 6.3.2.

Param.	Former value	New value
$\bar{x}_{cob}$	$-0.015\ m$	$?\ m$
$\bar{y}_{cob}$	$0.0\ m$	$?\ m$
$\bar{\phi}_{cob}$	-	$?\ rad$
$\bar{\theta}_{cob}$	-	$?\ rad$

Table E.7: Parameters of Robotis controllers (cf. Section 4.1). The outputs of the PD controllers are multiplied by a weight coefficient: $u = w \cdot (k_p e + k_d \dot{e})$. The subscript i corresponds to rf (right foot) or lf (left foot).

Controller	Weight (w)	P Gain (k_p)	D Gain (k_d)	Input (e)	Output (u)
IMU control:					
roll angle	1	1.0	0	$\phi^{des} - \phi^{meas}$	$\Delta\phi_{imu,1}$
roll velocity	0.1	0.3	0	$v_{\phi}^{des} - v_{\phi}^{meas}$	$\Delta\phi_{imu,2}$
pitch angle	1	1.0	0	$\theta^{des} - \theta^{meas}$	$\Delta\theta_{imu,1}$
pitch velocity	0.1	0.3	0	$v_{\theta}^{des} - v_{\theta}^{meas}$	$\Delta\theta_{imu,2}$
F/T control:					
force x	0.001	0.05	0.001	$F_{x,i}^{des} - F_{x,i}^{meas}$	$\Delta x_{ft,i}$
force y	0.001	0.05	0.001	$F_{y,i}^{des} - F_{x,i}^{meas}$	$\Delta y_{ft,i}$
force z	0.001	0.02	0.001	$F_{z,i}^{des} - F_{x,i}^{meas}$	$\Delta z_{ft,i}$
torque x	1	0.0015	0.0002	$\tau_{x,i}^{des} - \tau_{x,i}^{meas}$	$\Delta\phi_{ft,i}$
torque y	1	0.0015	0.0002	$\tau_{y,i}^{des} - \tau_{y,i}^{meas}$	$\Delta\theta_{ft,i}$

Table E.8: List of the new controllers (cf. Section 4.2).

Controller	Type	Input	Output
IMU control:			
roll angle	P	$\phi^{des} - \phi^{meas}$	$\Delta\phi_{imu,1}$
roll velocity	P	$v_{\phi}^{des} - v_{\phi}^{meas}$	$\Delta\phi_{imu,2}$
pitch angle	P	$\theta^{des} - \theta^{meas}$	$\Delta\theta_{imu,1}$
pitch velocity	P	$v_{\theta}^{des} - v_{\theta}^{meas}$	$\Delta\theta_{imu,2}$
F/T control:			
torque x	PD	$\tau_{x,i}^{des} - \tau_{x,i}^{meas}$	$\Delta\phi_{ft,i}$
torque y	PD	$\tau_{y,i}^{des} - \tau_{y,i}^{meas}$	$\Delta\theta_{ft,i}$
landing offset	offset	landing detection	$\Delta z_{offset,i}$
impact reducer	damping	$F_{z,i}^{des} - F_{z,i}^{meas}$	$\Delta z_{landing,i}$
force z	PD	$(F_{z,rf}^{des} - F_{z,lf}^{des}) - (F_{z,rf}^{meas} - F_{z,lf}^{meas})$	$\Delta z_{dsp,i}$

Table E.9: Saturation values of the outputs of the Balance Control (cf. Section 4.3). The subscript i corresponds to rf (right foot) or lf (left foot).

BC Output	Former saturation limits	New saturation limits
Δx_{cob}	$\pm 0.05\ m$	$\pm 0.05\ m$
Δy_{cob}	$\pm 0.05\ m$	$\pm 0.05\ m$
Δx_i	$\pm 0.05\ m$	$\pm 0.01\ m$
Δy_i	$\pm 0.05\ m$	$\pm 0.01\ m$
Δz_i	$\pm 0.05\ m$	$\pm 0.03\ m$
$\Delta \phi_i$	$\pm 15°$	$\pm 0.13\ rad$ ($\approx 7.4°$)
$\Delta \theta_i$	$\pm 15°$	$\pm 0.13\ rad$ ($\approx 7.4°$)
$\Delta \psi_i$	$\pm 15°$	-

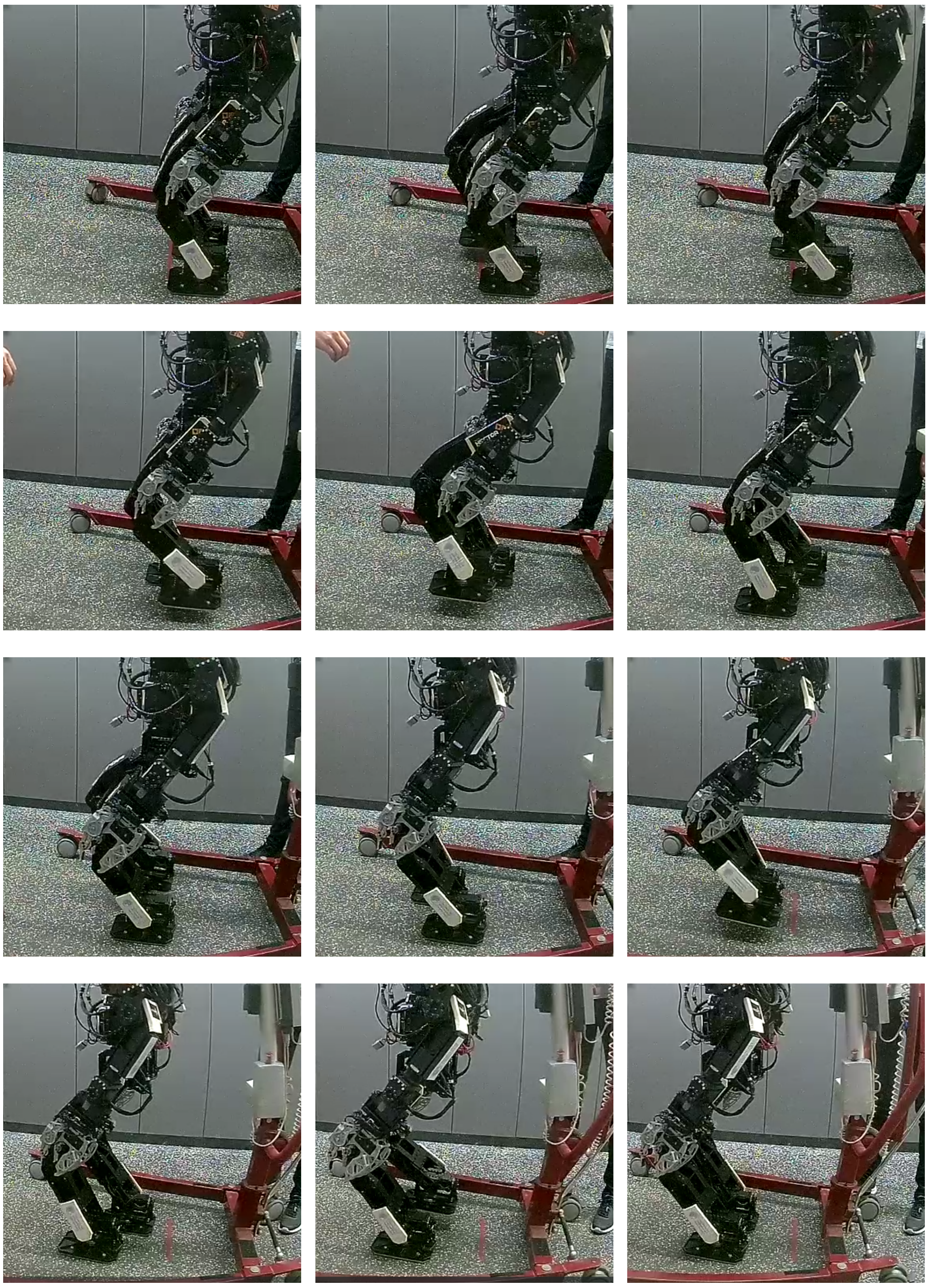


Figure F.1: Snapshots of the default walking gait (step length of $0.1\ m$ and step duration of $1\ s$) tested on the real robot with the new GPG and Robotis' BC

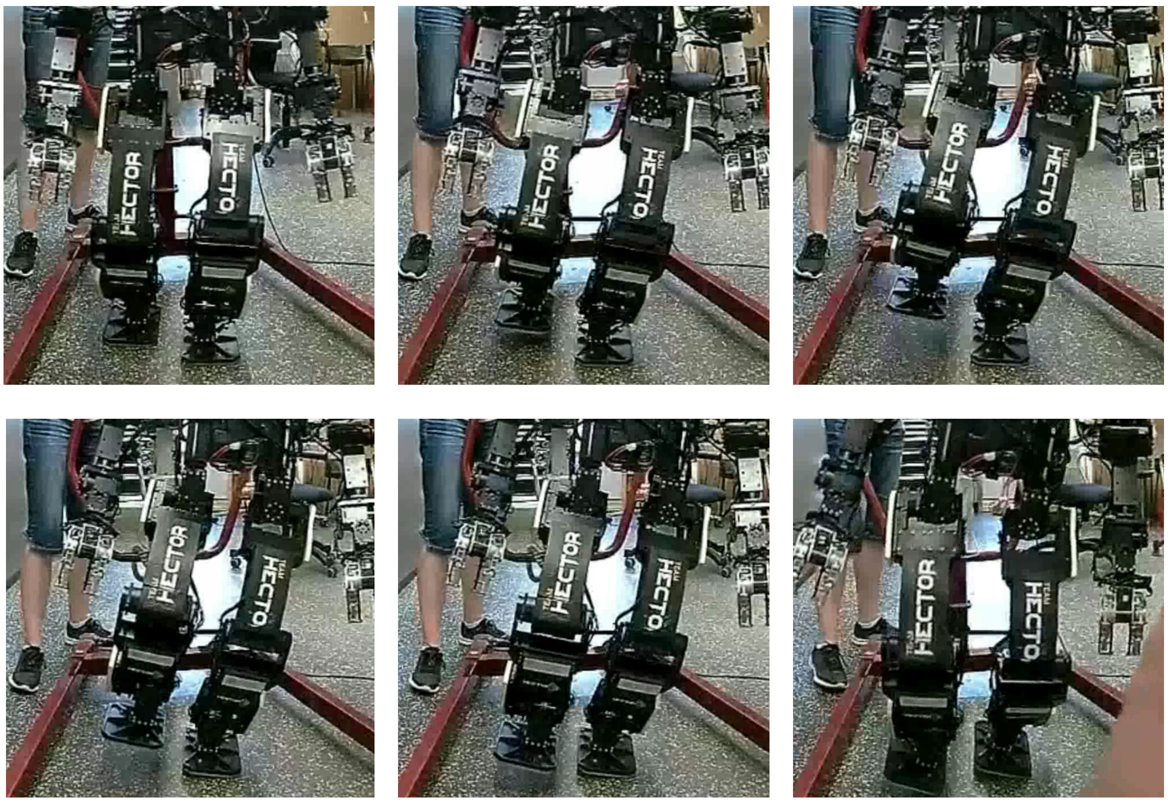


Figure F.2: Case of a large roll angle of the real robot that could not be stabilized by Robotis' IMU control loop (cf. Section 4.1.5). In the last snapshot, the right foot of Johnny #5 should already have reached the ground. The lapse of time between two snapshots is about $0.2\,s$.

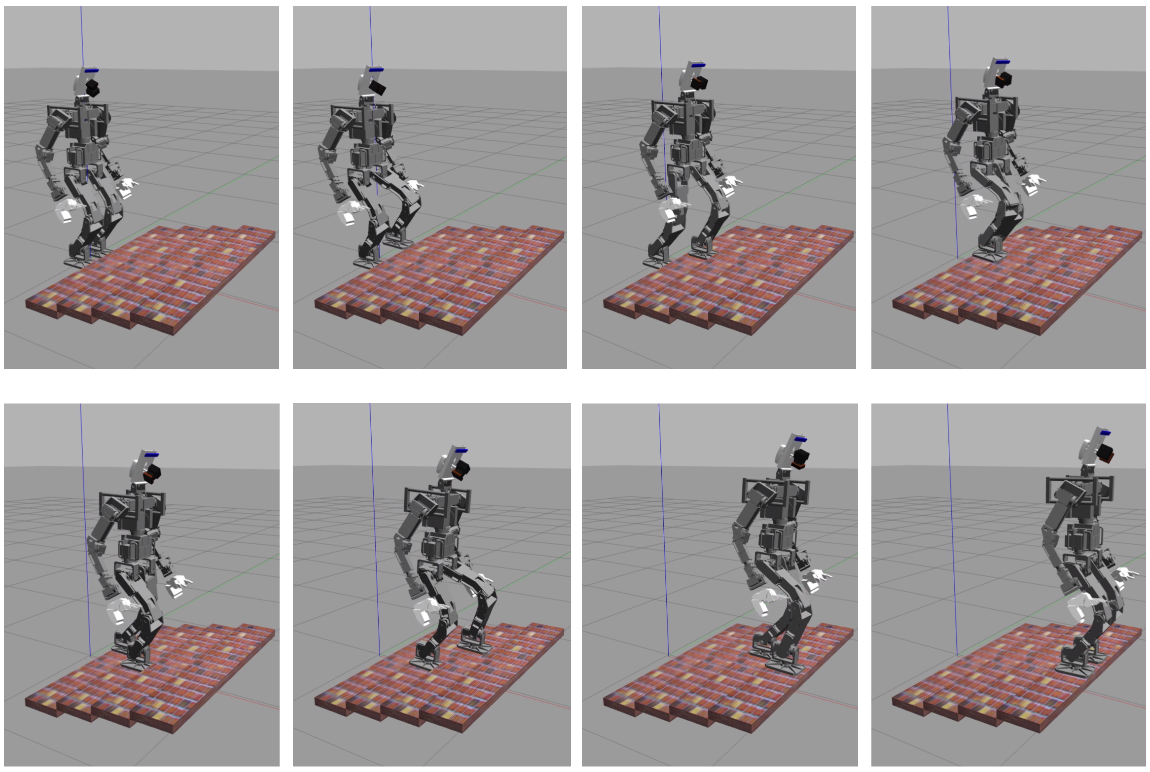


Figure F.3: Snapshots of a walking experiment on stairs of $5\,cm$ in the simulation with the new GPG and no BC (open loop mode)

Acronyms

BC	Balance Control
CoB	Center of Body
CoM	Center of Mass
CoP	Center of Pressure
DoF	Degree of Freedom
DSP	Double-Support Phase
EZMP	Extended Zero-Moment Point
F/T	Force/Torque
FK	Forward Kinematics
GPG	Gait Pattern Generator
GUI	Graphical User Interface
IK	Inverse Kinematics
IMU	Inertial Measurement Unit
l3	Legged Locomotion Library
LIPM	Linear Inverted Pendulum Model
MPC	Model Predictive Control
ROS	Robot Operating System
SSP	Single-Support Phase
URDF	Unified Robot Description Format
VCP	Virtual Contact Plane
VS	Virtual Slope
ZML	Zero-Moment Line
ZMP	Zero-Moment Point
ZMP-PC	ZMP Preview Control

List of Figures

List of Tables

Bibliography

[1] Robin R Murphy et al. *Search and rescue robotics*. In: *Springer handbook of robotics*. Springer, 2008, pp. 1151–1173.

[2] Robotis. *ROBOTIS e-Manual: Introduction*. 2018. URL: `http://emanual.robotis.com/docs/en/platform/thormang3/introduction/#introduction`. (accessed: 28.06.2018).

[3] Michael Thomas Rouleau. *Design and evaluation of an underactuated robotic gripper for manipulation associated with disaster response*. PhD thesis. Virginia Tech, 2015.

[4] Robotis. *THORMANG3: Full size open platform humanoid*. 2018. URL: `http://www.robotis.us/thormang3/`. (accessed: 28.06.2018).

[5] Florian Reimold. *Robust Locomotion of a Humanoid Robot Considering Grasped Objects*. MA thesis. TU Darmstadt, 2016.

[6] Marcell Missura. *Analytic and learned footstep control for robust bipedal walking*. PhD thesis. Universitäts-und Landesbibliothek Bonn, 2016.

[7] Thomas Buschmann et al. *Humanoide Laufmaschinen*. In: *at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs- und Informationstechnik* 61.4 (2013), pp. 217–232.

[8] Scott Kuindersma et al. *Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot*. In: *Autonomous Robots* 40.3 (2016), pp. 429–455.

[9] Andrea Del Prete et al. *Implementing torque control with high-ratio gear boxes and without joint-torque sensors*. In: *International Journal of Humanoid Robotics* 13.01 (2016), p. 1550044.

[10] Shuuji Kajita et al. *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.

[11] Thomas Buschmann. *Simulation and control of biped walking robots*. PhD thesis. Technische Universität München, 2010.

[12] Hayder FN Al-Shuka et al. *Multi-level control of zero-moment point-based humanoid biped robots: a review*. In: *Robotica* 34.11 (2016), pp. 2440–2466.

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

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

[15] Koichi Nishiwaki and Satoshi Kagami. *Frequent walking pattern generation that uses estimated actual posture for robust walking control*. In: *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*. IEEE. 2009, pp. 535–541.

[16] Miomir Vukobratović and Branislav Borovac. *Zero-moment point—thirty five years of its life*. In: *International journal of humanoid robotics* 1.01 (2004), pp. 157–173.

[17] Tohru Katayama et al. *Design of an optimal controller for a discrete-time system subject to previewable demand*. In: *International Journal of Control* 41.3 (1985), pp. 677–699.

[18] Pierre-Brice Wieber. *Viability and predictive control for safe locomotion*. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE. 2008, pp. 1103–1108.

[19] Tomoya Sato et al. *Walking trajectory planning on stairs using virtual slope for biped robots*. In: *IEEE transactions on industrial electronics* 58.4 (2011), pp. 1385–1396.

[20] Guangbin Sun, Hong Wang, and Zhiguo Lu. *A novel biped pattern generator based on extended ZMP and extended cart-table model*. In: *International Journal of Advanced Robotic Systems* 12.7 (2015), p. 94.

[21] Shuhei Shimmyo, Tomoya Sato, and Kouhei Ohnishi. *Biped walking pattern generation by using preview control with virtual plane method*. In: *Advanced Motion Control, 2010 11th IEEE International Workshop on*. IEEE. 2010, pp. 414–419.

[22] Shuhei Shimmyo and Kouhei Ohnishi. *Nested preview control by utilizing virtual plane for biped walking pattern generation including COG up-down motion*. In: *IECon 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE. 2010, pp. 1571–1576.

[23] Weiwei Huang et al. *Pattern generation for bipedal walking on slopes and stairs*. In: *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*. IEEE. 2008, pp. 205–210.

[24] Koichi Nishiwaki and Satoshi Kagami. *High frequency walking pattern generation based on preview control of ZMP*. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE. 2006, pp. 2667–2672.

[25] Shuhei Shimmyo, Tomoya Sato, and Kouhei Ohnishi. *Biped walking pattern generation by using preview control based on three-mass model*. In: *IEEE transactions on industrial electronics* 60.11 (2013), pp. 5137–5147.

[26] Chenglong Fu and Ken Chen. *Gait synthesis and sensory control of stair climbing for a humanoid robot*. In: *IEEE Transactions on Industrial Electronics* 55.5 (2008), pp. 2111–2120.

[27] Evrim Taşkıran et al. *Walking Control of a Biped Robot on an Inclined Plane*. In: *IFAC Proceedings Volumes* 42.19 (2009), pp. 254–259.

[28] Jiang Yi et al. *Walking algorithm of humanoid robot on uneven terrain with terrain estimation*. In: *International Journal of Advanced Robotic Systems* 13.1 (2016), p. 35.

[29] Shuuji Kajita et al. *Biped walking pattern generator allowing auxiliary zmp control*. In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 2993–2999.

[30] Koichi Nishiwaki and Satoshi Kagami. *Strategies for adjusting the zmp reference trajectory for maintaining balance in humanoid walking*. In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 4230–4236.

[31] Koichi Nishiwaki and Satoshi Kagami. *Simultaneous planning of com and zmp based on the preview control method for online walking control*. In: *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE. 2011, pp. 745–751.

[32] Holger Diedam et al. *Online walking gait generation with adaptive foot positioning through linear model predictive control*. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE. 2008, pp. 1121–1126.

[33] Arne-Christoph Hildebrandt et al. *Real-time pattern generation among obstacles for biped robots*. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 2780–2786.

[34] Shuuji Kajita et al. *A running controller of humanoid biped HRP-2LR*. In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 616–622.

[35] Shuuji Kajita et al. *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.

[36] Achim Stein. *Development of a Whole-Body Planning System for Humanoid Rescue Robots*. MA thesis. TU Darmstadt, 2015.

[37] John G Ziegler and Nathaniel B Nichols. *Optimum settings for automatic controllers*. In: *trans. ASME* 64.11 (1942).

[38] Mitsuharu Morisawa et al. *Reactive biped walking control for a collision of a swinging foot on uneven terrain*. In: *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE. 2011, pp. 768–773.

[39] Russ Tedrake and the Drake Development Team. *Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems*. 2016. URL: `http://drake.mit.edu`. (accessed: 20.06.2018).

[40] Scott Kuindersma et al. *Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot*. In: *Autonomous Robots* 40.3 (2016), pp. 429–455.

[41] Kenji Kaneko et al. *Design of prototype humanoid robotics platform for HRP*. In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. Vol. 3. IEEE. 2002, pp. 2431–2436.

[42] Marie-Sophie Elisa Schumacher. *Development and implementation of a demonstrator for human-compatible control approaches with an integrated servo drive*. MA thesis. TU Darmstadt, 2015.

[43] Robotis. *ROBOTIS e-Manual v1.27.00: Control Table*. 2010. URL: `http://support.robotis.com/en/product/actuator/dynamixel_pro/dynamixelpro/control_table.htm`. (accessed: 20.06.2018).