SAPIENZA
UNIVERSITÀ DI ROMA

Planning and Executing Humanoid Gaits in a World of Stairs

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Artificial Intelligence and Robotics

Candidate

Michele Cipriano
ID number 1764645

Thesis Advisor

Prof. Giuseppe Oriolo

Academic Year 2018/2019

Thesis defended on 21 January 2020
in front of a Board of Examiners composed by:

Prof. Daniele Nardi (chairman)

Prof. Febo Cincotti

Prof. Alessandro De Luca

Prof. Giuseppe Oriolo

Prof. Simone Scardapane

Planning and Executing Humanoid Gaits in a World of Stairs
Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: January 15, 2020

Author's email: cipriano.1764645@studenti.uniroma1.it

A mamma e papà.

Acknowledgments

Desidero ringraziare il Professor Giuseppe Oriolo per aver seguito il mio lavoro durante lo sviluppo della tesi, per i suoi consigli e per il tempo che ha dedicato a questo progetto. Ringrazio Paolo Ferrari e Nicola Scianca per tutte le chiacchierate sul planner e sull'MPC e per avermi aiutato durante gli esperimenti. Ringrazio Vincenzo e Emanuele per i numerosi consigli sul NAO e su BHuman. Ringrazio Filippo per tutti i suggerimenti e le discussioni sui robot umanoidi.

Ringrazio tutte le persone che ho conosciuto al DIAG, che hanno reso questo periodo fantastico. Ringrazio Tiziano, per i famosi caffè delle otto. Ringrazio Giuseppe, per avermi fatto appassionare ai controlli. Ringrazio Gabriele, Luigi, Luca, Eric, Ivan e tutta la "Egyptian crew" per tutti i bellissimi momenti passati insieme. Ringrazio Roberto, Giovanni, Andrea e Riccardo per la folle esperienza dell'hackathon a Torino.

Ringrazio Giulia, il mio grande amore, che mi ha accompagnato in questa avventura sin dal primo giorno, per tutto il tempo che abbiamo passato insieme, per tutte le serate, per tutti i weekend, per tutte le emozioni, e per avermi insegnato che spesso i punti di vista devono essere obiettivi. Ringrazio Iacopo, che considero da sempre un fratello, per tutte le giornate passate assieme in questi anni. Ringrazio Arianna, la mia sorellina, per essere stata al mio fianco in ogni momento della mia vita. Ringrazio nonno Aurelio, per tutte le passeggiate e tutte le volte che abbiamo cantato insieme. Il rigraziamento più grande va ai miei genitori, per avermi sostenuto in tutte le mie scelte e per tutti i sacrifici che hanno fatto da sempre. Senza di loro tutto ciò non sarebbe stato possibile ed è per questo che gliene sarò eternamente grato.

Abstract

The thesis considers a scenario in which a humanoid robot needs to walk in an unknown *World of Stairs* environment, namely an uneven ground composed by horizontal patches located at different heights. The approach considers three stages: mapping, off-line footstep planning and on-line gait generation. The mapping stage is based on a framework called `elevation_mapping`, that provides a representation of the envionment known as elevation map. The planning stage is based on Rapidly Exploring Random Trees (RRTs), which efficiently searches for a footstep sequence. The gait generation uses a Variable Height CoM Intrinsically Stable MPC, that allows walking on uneven ground. The implementation of the proposed architecture has been tested on both simulated dynamic environment and NAO humanoid robot, which has been equipped with a depth camera.

Contents

Chapter 1

Introduction

One of the biggest challenges in modern research is the study and the development of humanoid robots. Building a machine that is capable of executing the same tasks that the humans do is of fundamental importance for the future of our society, freeing people from dangerous and laborious jobs, increasing productivity and well-being. The idea of replicating the characteristics of the human body has fascinated humanity since the conception of Leonardo's Robot (1495) by Leonardo Da Vinci [1]. Human knowledge has advanced so much in the last centuries that the dream of building a machine that resembles the human body has finally become true.

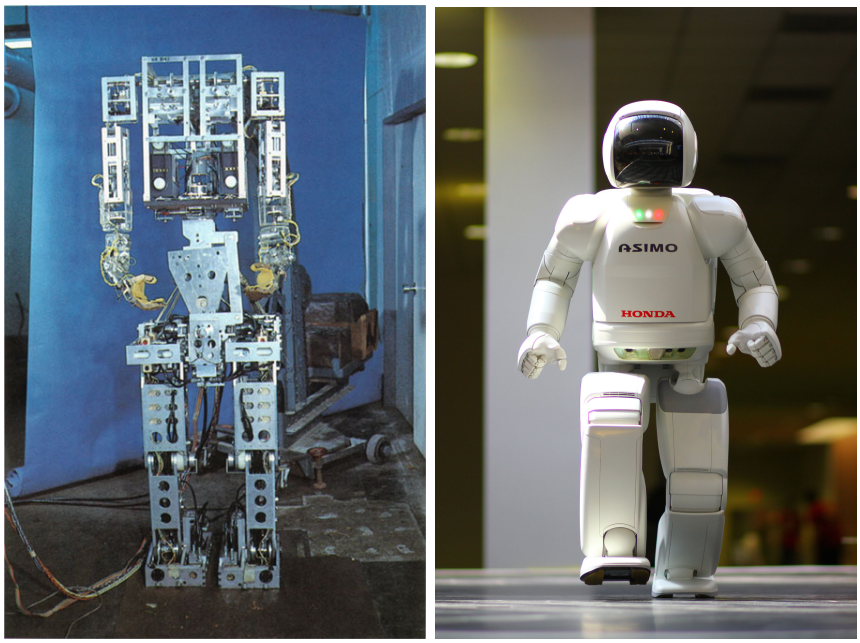
This chapter introduces humanoid robots, giving an overview of the technology that has been developed in the last decades and a comparison to other robots such as manipulators and unmanned ground vehicles. The problem of legged robot locomotion is then discussed, describing the current state-of-the-art research, the modules that allow a humanoid robot to safely move inside a specific environment, such as localization, mapping, planning and control, and the way they seamlessy work together in order to realize humanoid gaits.

Even if humanoid robots are nowadays capable of realizing astonishing tasks, many challenges are yet to be solved before their introduction in our society, making them part of our daily life. The goal of this thesis is that of studying humanoid robot locomotion in a world known as *World of Stairs*, where the environment surrouding the robot is composed of horizontal patches located at different heights [2]. This chapter concludes by presenting an overview of the thesis, discussing its structure and the content of each chapter.

1.1 Humanoid Robots

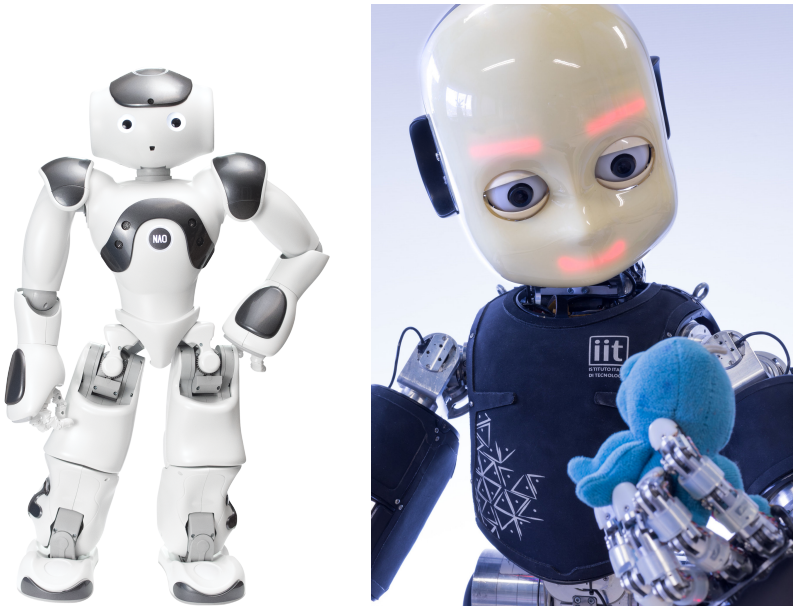
The development of humanoid robots started in 1967 with the WABOT-1 [3], shown in Fig. 1.1a, created by Waseda University. WABOT-1 is the first anthropomorphic robot able to walk with its limbs and carry objects with its hands. It was also equipped with a vision and a communication system that allowed it to communicate with people in Japanese. The most famous humanoid robot is probabily ASIMO [4], developed by HONDA (Fig. 1.1b). Its development started in the 1980s and it was preceded by many different version before being presented in 2000. The last version of ASIMO is 130 cm tall and is able to recognize objects, gestures, sounds and faces, making it able to interact with humans. It is equipped with multiple sensors such as laser and infrared that allow it to navigate autonomously. ASIMO is able to walk and run up to a speed of 9 km/h with an autonomy of one hour. Another very famous robot is NAO [5], which development has been started by Aldebaran and continued by Softbank after its acquisition. NAO (Fig. 1.2a) is the official robot used in the RoboCup [6] Standard Platform League, an international competition that consists in a soccer tournament where the teams are composed of 5 robots. The goal of RoboCup is that of winning against a FIFA World Cup winner team by 2050. Most recent and advanced humanoid robots includes iCub [7], shown in Fig. 1.2b, designed by the RoboCub Consortium and built by Italian Institute of Technology, with the idea of being a research platform for cognitive robotics; ATLAS (Fig. 1.3a), developed by Boston Dynamics for emergency service and rescue operations, such as those described by DARPA Robotics Challenge [8], illustrated in Fig. 1.4; Valkyrie (Fig. 1.3b), developed by NASA with the aim of advancing human spaceflight and extraterrestrial exploration [9].

Unlike other traditional robots, such as industrial manipulators (Fig. 1.5a) or wheeled mobile robots (Fig. 1.5b), humanoid robots are capable of handling a larger number of scenarios, such as rescue operations and space exploration, described above, but also classical ones that are part of our daily life, namely all the activities that humans perform during the day, that go from the manipulation of simple objects to communication. Humanoids are able to cover all these tasks because of their similarity to the humans, allowing them to interact with our environment and with other people. In order to successfully perform these tasks, humanoid robots need to properly move inside complex environments.



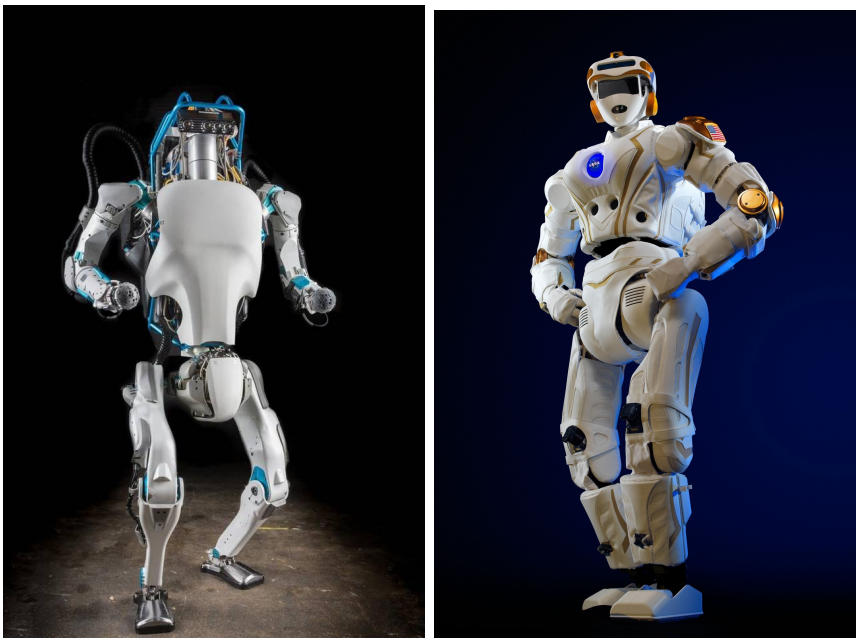
(a) (b)

Figure 1.1. WABOT-1 [3] on the left, the first anthropomorphic robot. ASIMO [4] on the right.



(a) (b)

Figure 1.2. NAO v6 [5] on the left, used in RoboCup Standard Platform League. iCub [7] on the right, a cognitive robotics research platform.



(a) (b)

Figure 1.3. ATLAS, on the left, and Valkyrie [9], on the right, have been developed with the purpose of advancing rescue operations and space exploration missions.

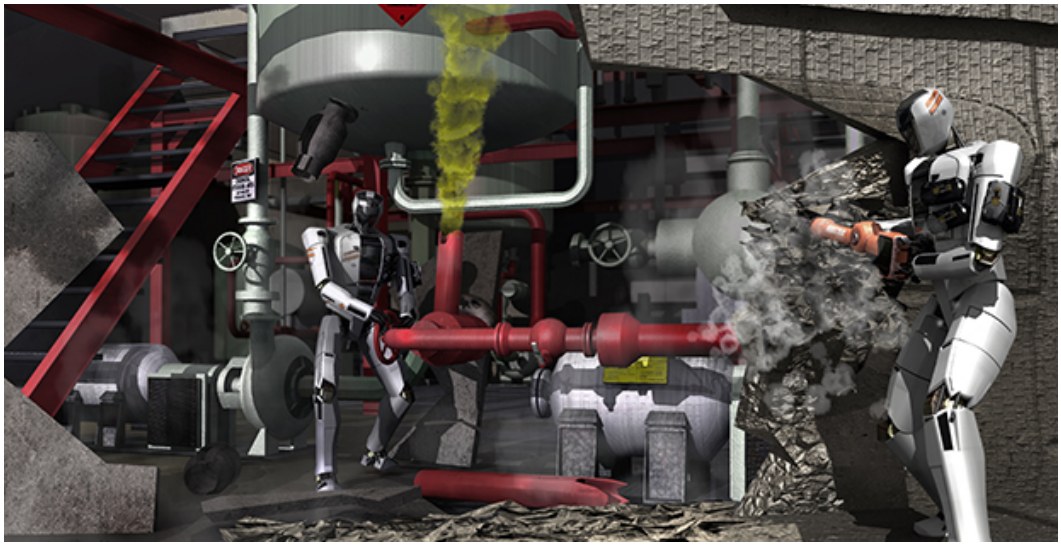


Figure 1.4. Typical scenarios described by DARPA Robotics Challenge [8], where the human intervention is dangerous and could put at risk life of rescuers.

(a) (b)

Figure 1.5. On the left, KUKA LWR [10], a famous industrial manipulator. On the right, HEAP [11], an autonomous hydraulic walking excavator.

1.2 Legged Robot Locomotion

The problem of humanoid robot locomotion, or more in general legged robot locomotion, consists in developing algorithms that allow humanoids to safely move inside an environment. In order to successfully perform this task, a humanoid robot needs to understand the structure of the surrounding environment and its configuration with respect to the world. Once a representation of the environment has been created, the robot needs to plan its motion in order to move without colliding with obstacles and without compromizing its integrity. The plan needs to be correctly executed, keeping the robot in equilibrium, avoiding falls and collisions. These are the most important characteristics that allow a sound generation of humanoid gaits.

A typical humanoid robot locomotion pipeline consists in a localization module, that determines the configuration of the robot with respect to the world; a mapping module, that determines a representation of the world that can be used efficiently by the robot; a planning module, that uses the configuration of the robot and the configuration of world in order to generate a plan that allows the humanoid to move from its configuration to a specified one; a control module, that guarantees that the motion generated by the planning module is correctly executed.

1.2.1 Localization and Mapping

The purpose of localization is to provide the pose of the robot with respect to the world. Localizing accurately is of fundamental importance in order to guarantee that the robot behaves in a correct way during the execution of its tasks. The introduction of advanced sensors, such as IMU (accelerometer, gyroscope), RGB-D

cameras, LIDAR and sonars, makes it possible to observe the surrouding environment and consequently to localize the robot. Many different techniques can be used for robot localization, such as particle filters [12], Extended Kalman Filters [13] and V-SLAM [14].

The purpose of mapping is to provide a representation of the environment that can be used to make the robot navigate. A mapping program should be able to recreate the world with enough details to make locomotion safe. Moreover, the implementation should be efficient enough to provide the map in real-time. These characteristics are essential to make the robot truly autonomous. The most used representations are elevation maps [15], which store the map as a grid such that for each point (x, y, z) on the surface there exists a cell (x, y) in the grid that stores the corresponding height z, and OctoMaps [16], a representation based on octrees that efficiently determines if a point in space is occupied or free.

1.2.2 Planning

A planning algorithm consists in generating a sequence of motion primitives that allow a robot to move from its current location to a desired one. In order to generate this sequence, the planner needs a map that represents the environment. Regarding humanoid robots, most famous planning approaches generate sequence of consecutive footsteps that the robot should follow in order to reach the desired goal. There exists a variety of techniques based on heuristic search, such as A* [17], that provides an optimal plan, and ARA* [18], that provides a suboptimal plan but is faster than the previous method. Other techniques are optimization-based, i.e. they generate a plan by solving an optimization problem, such as [19], that solves a quadratic programming (QP) problem, and [20], that solves a mixed-integer quadratically-constrained quadratic programming (MIQCQP) problem. Another family of algorithms are sampling-based and generates a tree of footstep configuration randomly, for example by using Rapidly-Exploring Random Trees (RRTs) [2].

1.2.3 Control

Once a footstep plan has been generated, the robot needs to execute it without falling. Here, a motion model is needed to describe the behaviour of the system. Usually, since the dynamics of the humanoid robots is complex, a simplified once is used, such as the dynamics of the Linear Inverted Pendulum (LIP) [21]. The introduction of the LIP allowed the development of control modules that generate humanoid gaits. The most famous works include the Preview Control [22], which uses a dynamic extension of the Cart-Table model solving a LQR problem; the

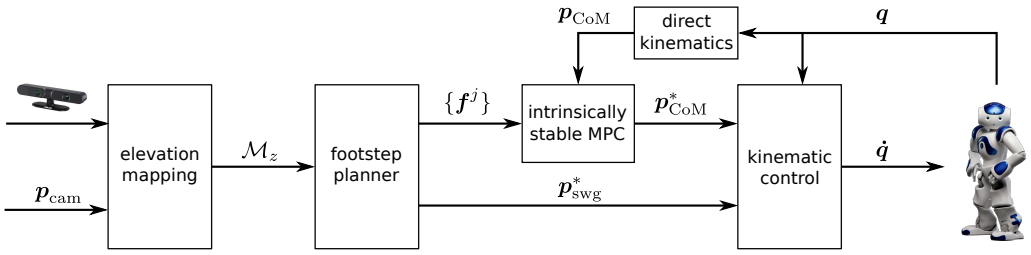


Figure 1.6. Block scheme of the approach.

Model Predictive Control (MPC) [23], which constraints ZMP to keep the robot balanced, solving a QP problem; the Intrinsically-Stable MPC (IS-MPC) [24], which guarantees to produce stable CoM trajectories by constraining both the ZMP and the unstable mode of the LIP. After the generation of a desired trajectory, a kinematic controller is responsible for execution of the motion task at the lowest level.

1.3 Thesis Overview

The goal of the thesis is that of studying humanoid robot locomotion in a world known as *World of Stairs*, a particular uneven ground composed by horizontal patches located at different heights [2]. The idea is that of making NAO humanoid robot walk in a complex scenario, making it climb stairs and avoid obstacles. The robot needs to autonomously perceive the surrouding world, create a representation of the environment, generate a plan that allows the robot to safely reach a specified region of the world and correctly execute the motion. The robot has been equipped with an RGB-D sensor, which has been mounted on its head.

The approach is to use the depth sensor of the camera in order to generate an elevation map of the environment. To do this, `elevation_mapping` [25] has been used. Once the map has been created, a footstep planner based on RRT [2] generates a sequence of footsteps with the respective swing foot trajectories (taking into account collisions and the limitations of the robot) that allow the robot to reach a desired final position. To guarantee that the motion is properly executed, a Variable-Height CoM IS-MPC [26] is used to generate a trajectory of the CoM of the robot. In the end, a pseudo-inverse based kinematic controller computes the joint commands of the robot needed to execute the motion. Fig. 1.6 illustrates a block scheme of the described approach.

This thesis extends [2], providing an implementation of the Variable-Height CoM IS-MPC for a real humanoid robot (NAO), extending the RRT-based footstep planner to work in a real environment with NAO and introducing `elevation_mapping`

(directly extending the block scheme of [2]) in order to generate an elevation map of the terrain, making NAO able to execute gaits in a *World of Stairs* unknown environments.

Chapter 2 introduces `elevation_mapping`, describing the framework and its integration with the camera, the planner and the robot. Chapter 3 describes the RRT-based footstep planner introduced in [2], formulating the problem, the algorithm and discussing its integration with NAO. Chapter 4 describes the Variable-Height CoM IS-MPC and the implementation on the BHuman framework [27]. Chapter 5 presents the experiments performed on the robot in different scenarios, using the three modules described in the previous three chapters. Chapter 6 concludes the thesis by summarizing the obtained results and discussing future works.

Chapter 2

Elevation Map Generation

When dealing with robot locomotion, the representation of the environment plays a fundamental role. It is, in fact, extremely important to properly understand the structure of the world in order to safely make the robot move, avoiding obstacles and dangerous zones, and to make it successfully complete its tasks. The world that surrounds the robot can be represented in many different ways; it is important to choose a proper representation to keep computational costs low and to make locomotion realizable.

In *World of Stairs* scenarios, introduced in the previous chapter, the most efficient way to represent environments is by using elevation maps. An elevation map is a grid that contains for each coordinate (x, y) of the world its respective coordinate z. Hence, it can be seen as a function $\mathcal{M}_z$ such that, for each element i of the grid, $z_i = \mathcal{M}_z(x_i, y_i)$. This kind of representation enables the development of planners that quickly find plans to make robots move from a position to another inside the world.

This chapter introduces `elevation_mapping` [25], the framework used in this thesis to generate elevation maps, which allow NAO to navigate in unknown environments (more precisely in *World of Stairs* environments); the ASUS Xtion Pro, an RGB-D sensor equipped on top of NAO, which has been used to send depth informaton to `elevation_mapping`; the behaviour of the framework when a map is build using the ASUS Xtion Pro placed on the head of NAO humanoid robot. The generated map is the one used in the experiment "Stair Climbing in Unknown Environment", described in Section 5.5, and it is used by the footstep planner (Chapter 3) to make NAO climb the stairs.

2.1 Framework

`elevation_mapping` [25] is a framework that allows to create elevation maps in rough environments using proprioceptive localization and distance sensors of the robot in real-time. The generated map is robot-centric, meanining that the map is a local representation of the environment that surrounds the robot. Using a robot-centric representation instead of a world-centric one (where the map is expressed with respect to an inertial frame) has the advantage of not depending on the global pose of the robot. This allows to avoid problems related to drift in the state estimation which would cause the maps to be inconsistent and not accurate enough to be used for locomotion.

The framework represents the map as a grid, providing for each cell a probabilistic distribution of the height of the terrain in the corresponding position. The map is updated using range measurements, provided by a sensor mounted on the robot, and the motion of the robot, provided by a localization module such as a Kalman filter. The program supports dynamic environments and it is built in C++, providing a ROS [28] interface which makes it simple to be integrated with external sensors and other programs. `elevation_mapping` is built upon GridMap [29], which efficiently manages 2D grid representations.

Even if not all the characteristics of the framework have been used in this thesis, the choice to integrate `elevation_mapping` into the project has been essential to quickly build elevation maps that can be used by a footstep planner (Chapter 3). Moreover, its versatility (because of the ROS interface) and its capabilities to generate efficiently maps for rough and dynamic environments, set a good starting point to extend this project to more complex scenarios. This section gives a brief overview of the framework without describing all the features in detail, thoroughly described in [25].

2.1.1 Definitions

The notation of the reference frames used by the framework are illustrated in Fig. 2.1. In particular, I is the inertial frame, B is the base frame, S is the sensor frame and M is the map frame. The inertial frame I is considered to be fixed with respect to the terrain. Both the base frame B and the sensor frame S are attached to the robot and it is assumed that the transformation (r_{BS}, Φ_{BS}) between the two is known in advance. The elevation map is attacched to the reference frame M. B and M are related through a transformation (r_{BM}, Φ_{BM}) given by the user. The z axis of the map frame M and the inertial frame I are always aligned. A cell i of the

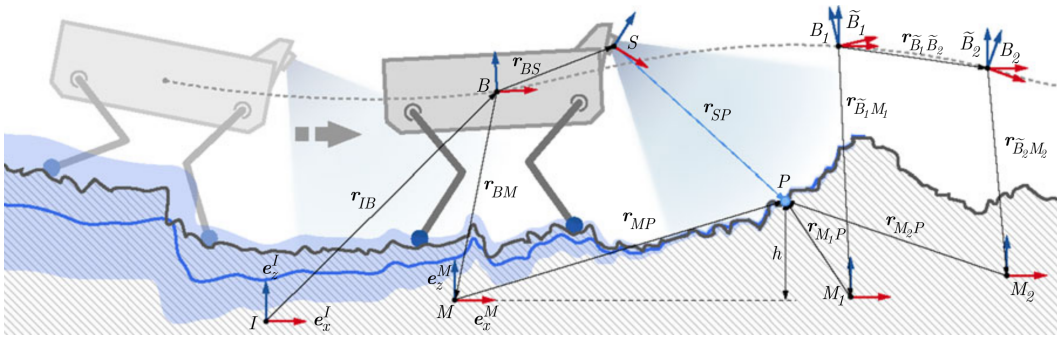


Figure 2.1. The figure shows the notation of the reference frames used by `elevation_mapping`. [25] together with the robot moving inside a rough environment.

elevation map corresponding to the point of the surface $P_i = (x_i, y_i, \hat{h}_i)$, which is expressed with respect to M. The transform (r_{IB}, Φ_{IB}) between the position of the robot B and the inertial frame I is provided by a state estimation module.

2.1.2 Map Update

The map is updated when new measurements are read by the distance sensor and when the localization module updates the position of the robot. Each measurement ${}_S r_{SP}$ in the sensor frame S is first transformed into the corresponding height measurement p using the kinematic representation described before (Fig. 2.1). The new height measurement can be represented as a Gaussian probability distribution $\tilde{p} \sim \mathcal{N}(p, \sigma_p^2)$ given the sensor noise model [30]. In this way, it is possible to update the current estimate of the height $\mathcal{N}(\hat{h}, \sigma_h^2)$ by using a 1D Kalman filter, obtaining a new estimate of the height $\mathcal{N}(\hat{h}^+, \sigma_h^{2+})$. The filter is updated only if the new measurement is not too far away (in terms of Mahalanobis distance) from the current estimate.

The height estimate of each cell needs to be updated also during motion in order to take into account the uncertainty of the motion itself. To speed up the process, each cell i is extended with a spatial covariance matrix Σ_{P_i} that, in addition to the uncertainty of the height $\sigma_{h_i}^2$, considers the uncertainty along the x and y axes.

2.1.3 Map Fusion and Dynamic Environments

The map fusion step is performed whenever required and it consists in transforming the elevation map data from a probabilistic representation of the kind $\mathcal{N}(\hat{h}_i, \Sigma_{P_i})$ to a deterministic representation $(\hat{h}_i, h_{i,\min}, h_{i,\max})$, where the values $h_{i,\min}, h_{i,\max}$ respectively represent the lower and the upper bound of the height estimation such that h_i has 95% probability of being in the range $[h_{i,\min}, h_{i,\max}]$. In this thesis the

Figure 2.2. The ASUS Xtion Pro [32] is equipped with a depth sensor and it is easily configurable to make it work with ROS. This simplifies the integration with `elevation_mapping` and, consequently, the construction of a navigable map.

fused map is used by the footstep planner (Chapter 3) to generate a plan for the robot, similarly to [31].

`elevation_mapping` handles dynamic environments by performing a visibility check based on ray tracing. Since this is a computationally expensive task, it is performed in parallel to the data acquisition process and it is run at a low frequency (1 Hz).

2.2 ASUS Xtion Pro

The camera (Fig. 2.2) used in this thesis is an ASUS Xtion Pro [32], which is equipped with a depth sensor, used to generate the height measurements sent to `elevation_mapping`. The communication between the camera and the mapping framework is handled by ROS. ASUS Xtion Pro works up to a resolution of 640×480 at 30 fps. Its working range is from 0.5m up to 3.5m. The camera has been calibrated with a tool called `camera_calibration` [33].

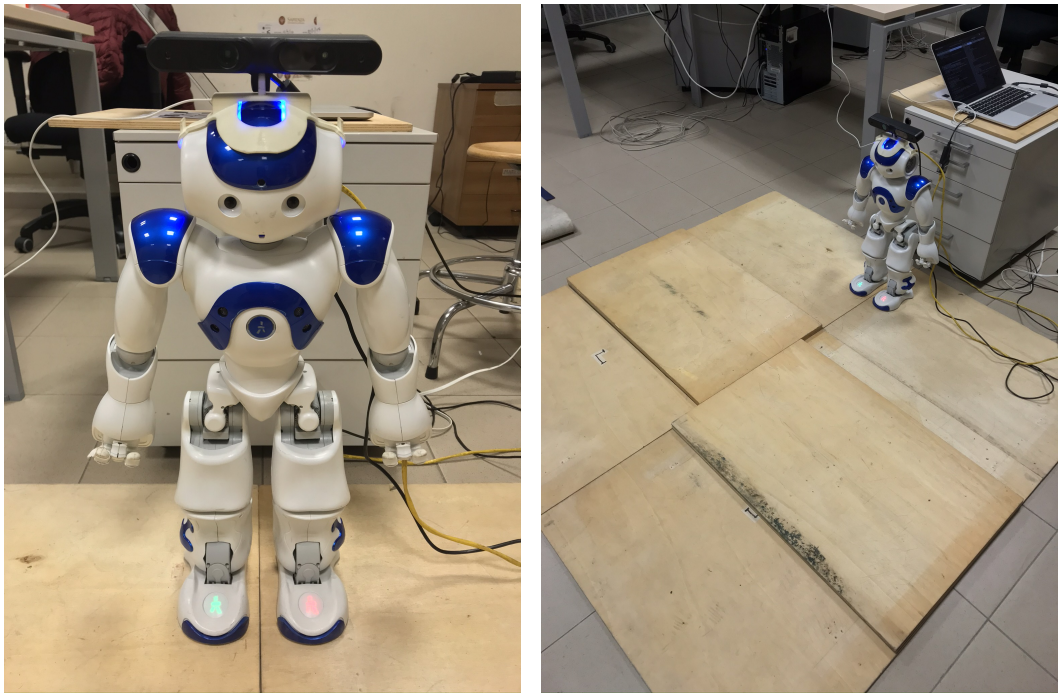
2.3 World of Stairs

The integration between `elevation_mapping` and the ASUS Xtion Pro allows to easily generate elevation maps in rough environments, meaning that the same settings could be used to generate elevation maps for *World of Stairs* scenarios, described in

the previous chapter.

As mentioned before, ROS simplifies the communication between the camera and the mapping framework, allowing to develop a block scheme (Fig. 1.6) that connects the camera to the robot. The humanoid robot used in this thesis is NAO, shown in Fig. 2.3a where it has been equipped with the ASUS Xtion Pro.

The *World of Stairs* scenario considered for the experiments is the one shown in Fig. 2.3b, where the robot stands in front of a stairway. The idea is to position the camera towards the stairs in order to generate an elevation map accurate enough to be used by a footstep planner (Chapter 3). Given the pose of the camera, which is sent to `elevation_mapping` together with the depth information of the camera, it is possible to quickly generate a map. Of course, the area underneath the robot can not be seen by the camera. That is why a "safe zone" is manually added to the final elevation map before sending it to the planner. Fig. 2.5 shows the map generated by the framework (together with the "safe zone") when the camera is oriented towards the stairs (as in Fig. 2.4). The generated map is the one used by the footstep planner in the scenario "Stair Climbing in Unknown Environment", described in Section 5.5, where NAO is able to climb the stairs in an unknown environment.



(a) (b)

Figure 2.3. On the left, NAO humanoid robot with ASUS Xtion Pro placed on top. On the right, NAO humanoid robot in the environment described in Section 5.5, right before starting the execution of the experiment.

Figure 2.4. RGB image seen by the ASUS Xtion Pro placed on top of the robot. The corresponding depth image is sent to `elevation_mapping` to build the map.

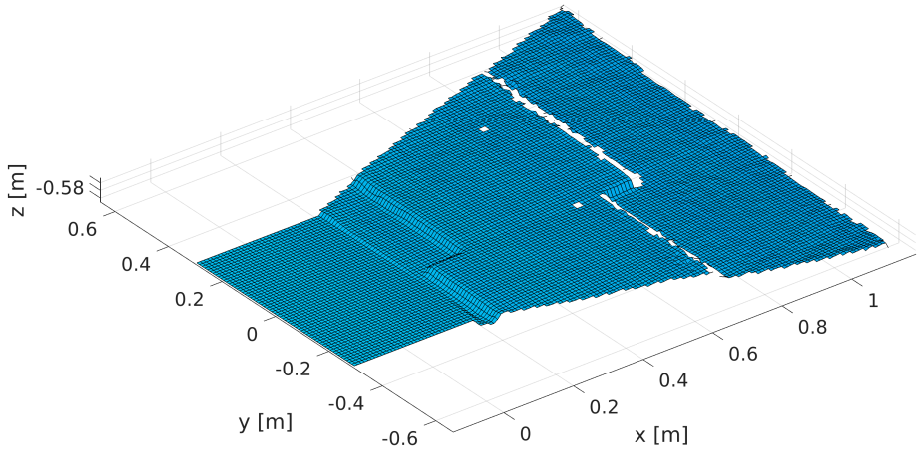


Figure 2.5. Elevation map build by `elevation_mapping` for the scenario "Stair Climbing in Unknown Environment" described in section 5.5.

Chapter 3

RRT-based Footstep Planning

Considering the *World of Stairs* scenarios discussed previously, the aim of a footstep planner is to determine a feasible sequence of footsteps that allows the humanoid robot to reach a desired goal region $\mathcal{G}$, given a representation of the environment, in this case the elevation map described in the previous chapter.

This chapter presents a footstep planner based on Rapidly Exploring Random Trees (RRTs), presenting its algorithm and discussing its integration into the project architecture.

3.1 Problem Formulation

Before describing the algorithm, introduced in [2], a more detailed formulation of the problem should be given.

3.1.1 Notation and Plan Feasibility

As introduced before, an elevation map is a proper choice for the representation of the scenarios described by *World of Stairs*, since it is efficient to store and to query. Let the map be denoted by $\mathcal{M}_z$. The height from the ground of a cell having coordinate (x, y) is $z = \mathcal{M}_z(x, y)$.

Given $\mathcal{M}_z$, the goal of the footstep planner is to find a feasible sequence of footsteps $\{\boldsymbol{f}^j\}$ which leads to a desired location $\mathcal{G}$, together with the corresponding swing foot trajectories $\{\boldsymbol{p}^j_{\text{swg}}\}$. Let $\boldsymbol{f}^j = (x^j_{\text{f}}, y^j_{\text{f}}, z^j_{\text{f}}, \theta^j_{\text{f}})^T$ be the pose of the j-th footstep, with $(x^j_{\text{f}}, y^j_{\text{f}}, z^j_{\text{f}})$ the position of the footstep and θ^j_{f} the yaw orientation of the footstep. Note that in the scenarios represented by *World of Stairs* the roll and pitch angles of the footsteps are always zero.

In order to make the footstep plan feasible, the following requirements are introduced:

R1 The height variation between two consecutive footsteps is with a maximum range: $|z_{\mathrm{f}}^{j} - z_{\mathrm{f}}^{j-1}| \leq \Delta z_{\max}$.

R2 The footstep is fully in contact with the ground, hence, each cell of $\mathcal{M}_z$ which belongs to the footprint has the same height z_{f}^{j}.

R3 The swing foot trajectory $\boldsymbol{p}_{\mathrm{swg}}^{j}$ is collision free (apart of course at the start and at the end of the trajectory itself).

Once the footsteps have been generated, they are passed to an online gait generation block (described in Chapter 4) which computes the optimal CoM trajectory $\boldsymbol{p}_{\mathrm{CoM}}^{*}$ that allows the humanoid robot to execute the plan. The optimal swing foot trajectory $\boldsymbol{p}_{\mathrm{swg}}^{*}$ is defined by the appropriate subtrajectory $\boldsymbol{p}_{\mathrm{swg}}^{j}$.

3.2 Algorithm

The following algorithm [2], which is based on RRT (Rapidly Exploring Random Tree), finds a feasible sequence of footsteps in a *World of Stairs* scenario given the elevation map $\mathcal{M}_z$ of the environment, the goal region $\mathcal{G}$ and the starting configuration of the feet $\boldsymbol{f}_L, \boldsymbol{f}_R$. The generated sequence connects the starting point to the goal.

3.2.1 Pseudocode

The footstep planner, whose behaviour is described by Algorithm 1 iteratively builds a tree $\mathcal{T}$ of feet configurations in a randomized way. In general, a vertex $v = (\boldsymbol{f}_{\mathrm{sup}}, \boldsymbol{f}_{\mathrm{swg}})$ specifies the pose of the support foot and the swing foot during the phase of double support. An edge connecting two vertices exists when there exists a collision free trajectory of the swing foot between the two configurations.

The algorithm starts by initializing the tree with the initial configuration of the left and the right foot (line 1). At each iteration, a point $\boldsymbol{p}_{\mathrm{rand}}$ is selected randomly on the ground (line 5). Here, the planner randomly choose between exploration and exploitation mode. In the first case $\boldsymbol{p}_{\mathrm{rand}}$ is generated by randomly selecting a pair of coordinates x, y and retrieving the z corrdinate from the elevation map $\mathcal{M}_z$. In the second case $\boldsymbol{p}_{\mathrm{rand}}$ is sampled from $\mathcal{G}$. At this point, the vertex v_{near} of $\mathcal{T}$ that is closest to $\boldsymbol{p}_{\mathrm{rand}}$ is selected (line 6) in order to check whether it can be connected to the tree $\mathcal{T}$. The distance between v_{near} and $\boldsymbol{p}_{\mathrm{rand}}$ is determined using the following metric:

$$\gamma(v, \boldsymbol{p}) = d(\boldsymbol{m}, \boldsymbol{p}) + \alpha|\theta_p| \tag{3.1}$$

where $d(\boldsymbol{m}, \boldsymbol{p})$ is the Euclidean distance between the midpoint $\boldsymbol{m}$ between the feet (hence, between $\boldsymbol{f}_{\text{sup}}$ and $\boldsymbol{f}_{\text{swg}}$ in v) and $\boldsymbol{p}$, θ_p is the angle between the robot sagittal axis (determined by the average orientation of the feet) and the line joining $\boldsymbol{m}$ to $\boldsymbol{p}$, and $\alpha > 0$. Once v_{near} has been selected, the foot poses $\boldsymbol{f}_{\text{sup}}^{\text{near}}, \boldsymbol{f}_{\text{swg}}^{\text{near}}$ are extracted from it. A candidate footstep $\boldsymbol{f}^{\text{cand}}$ is randomly generated (line 7) by selecting a final pose of the swing foot from the catalogue of primitives U defined with respect to $\boldsymbol{f}_{\text{sup}}^{\text{near}}$, as shown in Fig. 3.1 and defined in the next section. As before, the z coordinate of $\boldsymbol{f}^{\text{cand}}$ is determined by $\mathcal{M}_z$. On line 8, requirements R1-R2 defined above are checked for $\boldsymbol{f}^{\text{cand}}$. In positive case, a collision checking algorithm (lines 9-13) is performed to verify whether there exists a collision free trajectory $\boldsymbol{p}_{\text{swg}}^{\text{cand}}$ (a second degree polynomial equation) that brings the swing foot from $\boldsymbol{f}_{\text{swg}}^{\text{near}}$ to $\boldsymbol{f}^{\text{cand}}$. In positive case, also requirement R3 is verified and a new vertex $v_{\text{new}} = (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}_{\text{sup}}^{\text{near}})$ is added to the tree $\mathcal{T}$ as a child of v_{near} (lines 14-17). The algorithm terminates when the midpoint $\boldsymbol{m}$ between the feet at the new vertex v_{new} is inside the goal region $\mathcal{G}$ or a maximum number of iterations $i_{\max}$ has been reached (line 18). When a solution has been found, the path joining the initial vertex $(\boldsymbol{f}_L, \boldsymbol{f}_R)$ to v_{new} is extracted from the tree and the footstep sequence $\{\boldsymbol{f}^j\}$ is reconstructed together with the swing foot trajectories $\{\boldsymbol{p}_{\text{swg}}^j\}$.

3.3 Implementation

The planner has been implemented in C++ and it has been tested on both dynamic environments and NAO humanoid robot. The elevation map is either generated by the `elevation_mapping` framework or manually generated before the execution of the program. Experiments are described in detail in Chapter 5. Note that to simplify the communication between `elevation_mapping` and the planner and the communication between the planner and the robot, the planner has been executed on an external computer, which is connected to the robot through an ethernet cable. The plan is sent through TCP. The communication is designed with the idea to extend the planner in order to handle replanning phases and dynamic environments. 5

3.3.1 Catalogue of Primitives

As mentioned before, the catalogue of primitives specifies the possible footsteps that the robot can perform at each step. In this thesis the catalogue for the NAO humanoid robot (left foot with respect to right foot) has been defined as:

$$(x, y, \theta) \in \{-6.0, 0.0, 6.0, 8.0, 10.0\} \times \{11.0, 12.0\} \times \{0, \pi/12\} \tag{3.2}$$

Algorithm 1: Footstep Planner

root the tree $\mathcal{T}$ at $v_{\text{ini}} \leftarrow (\boldsymbol{f}_L, \boldsymbol{f}_R)$;
$i \leftarrow 0$;
repeat
  $i \leftarrow i + 1$;
  generate a random point $\boldsymbol{p}_{\text{rand}}$ on the ground;
  select the closest vertex v_{near} in $\mathcal{T}$ to $\boldsymbol{p}_{\text{rand}}$ according to $\gamma(\cdot, \boldsymbol{p}_{\text{rand}})$;
  randomly select from the primitive catalogue U a candidate footstep $\boldsymbol{f}^{\text{cand}}$;
  if $\boldsymbol{f}^{\text{cand}}$ is feasible w.r.t. R1–R2 **then**
    $h \leftarrow h_{\min}$;
    $\boldsymbol{p}_{\text{swg}}^{\text{cand}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}_{\text{swg}}^{\text{near}}, \boldsymbol{f}^{\text{cand}}, h)$;
    while $h \leq h_{\max}$ **and** $\text{Collision}(\boldsymbol{p}_{\text{swg}}^{\text{cand}})$ **do**
      $h \leftarrow h + \Delta h$;
      $\boldsymbol{p}_{\text{swg}}^{\text{cand}} \leftarrow \text{BuildTrajectory}(\boldsymbol{f}_{\text{swg}}^{\text{near}}, \boldsymbol{f}^{\text{cand}}, h)$;
    if $h \leq h_{\max}$ **then**
      $v_{\text{new}} \leftarrow (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}_{\text{sup}}^{\text{near}})$;
      add vertex v_{new} to $\mathcal{T}$ as a child of v_{near};
      compute midpoint $\boldsymbol{m}$ between the feet at v_{new};
until $\boldsymbol{m} \in \mathcal{G}$ **or** $i = i_{max}$;

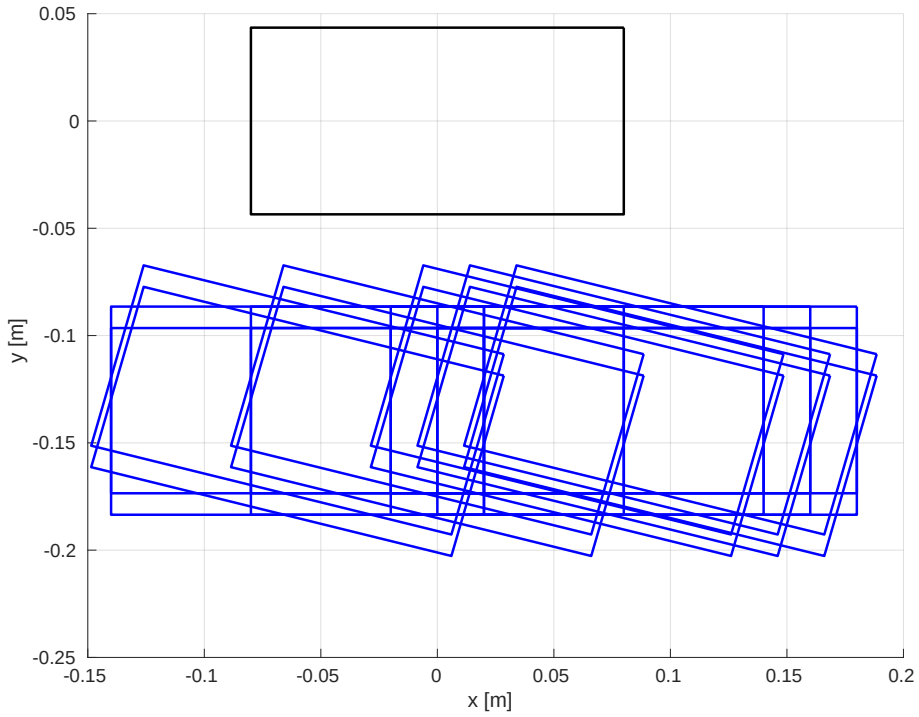


Figure 3.1. The catalogue of primitives (blue color) specifies the possible poses of the candidate footstep $\boldsymbol{f}^{\text{cand}}$ with respect to the pose of the current support foot $\boldsymbol{f}^{\text{near}}_{\text{sup}}$. The figure shows the case where the left foot is the support foot. The catalogue for the case where the right foot is the support foot is specular. The z coordinate of a candidate footstep can be retrieved directly from the elevation map $\mathcal{M}_z$.

The catalogue of primitives of the right foot with respect the left foot is symmetric and it is shown in Fig. 3.1. The footstep planning hyperparameters have been set in the following way: the goal region has been defined as a circle of radius 0.01m, $\alpha = 1$, $\Delta z_{max} = 0.045$m, $h_{\text{min}} = 0.02$m, $h_{\text{max}} = 0.07$m, $\Delta h = 0.01$m. The resolution of $\mathcal{M}_z$ has been set to 0.01m when using maps generated by `elevation_mapping` and to 0.02m when using maps generated manually.

Chapter 4

Variable Height CoM IS-MPC

Before describing in detail the model [26] that allows humanoid robots to walk on uneven terrain (in this case *World of Stairs*), it is important to introduce the notation and to make an overview of the previous works. Differently from manipulators, which are fixed to the ground, humanoids need to maintain equilibrium while walking. The contact with the ground is, in fact, continuously changed due to walking itself. Usually, this is achieved by making sure that the ZMP (Zero Moment Point) is always within the convex hull of the support polygon. The ZMP, introduced in [34], is the point where the horizontal component of the moment of the ground reaction forces becomes zero. To generate this kind of motion, usually a simplified model which only considers the CoM (center of mass) of the robot is used. In particular, by neglecting the robot angular momentum and by assuming the CoM height is constant, the CoM dynamics can be treated as a LIP (Linear Inverted Pendulum), introduced for the first time in [21]. The Linear Inverted Pendulum model easily allows to design control schemes for the generation of the CoM reference trajectory, like the Preview Control [22] and the Model Predictive Control [23]. Before discussing the LIP and extending it to the Variable Length Inverted Pendulum, the 3D motion model of the CoM is introduced. It is important to remind that the CoP (Center of Pressure) in case of flat ground is the point of application of the ground reaction force.

4.1 3D Motion Model

Let (x_z, y_z, z_z) and (x_c, y_c, z_c) respectively be the position of the ZMP and the position of the CoM. Assuming that the humanoid robot is walking on flat ground (hence the gravity acceleration components on x and y are zero) and neglecting the angular momentum around the CoM, the ZMP can be anywhere along the line connecting the CoP (located on the piece of surface upon which the support foot is

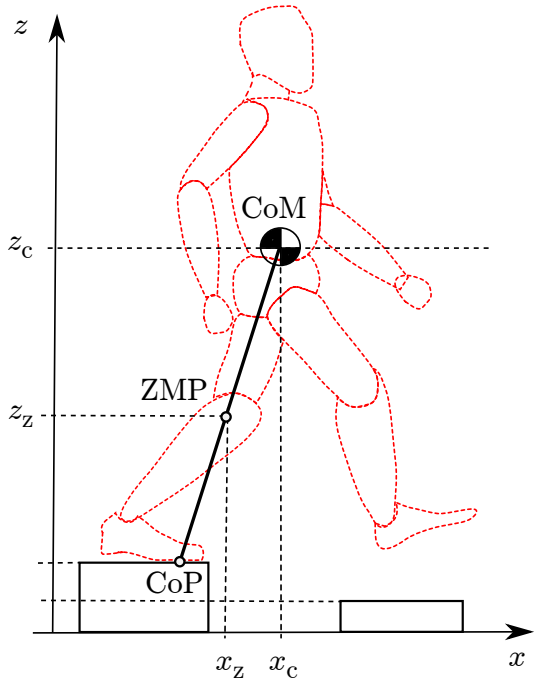


Figure 4.1. When walking on flat ground, or more in general on piecewise-horizontal ground as in the figure, the ZMP can be anywhere between the line connecting the CoP and the CoM.

placed) and the CoM (Fig. 4.1), it is possible [35] to obtain the dynamics of the CoM:

$$\ddot{x}_c = \frac{\ddot{z}_c + g}{z_c - z_z}(x_c - x_z) \tag{4.1}$$

$$\ddot{y}_c = \frac{\ddot{z}_c + g}{z_c - z_z}(y_c - y_z) \tag{4.2}$$

$$\ddot{z}_c = \frac{f_z}{m} - g \tag{4.3}$$

where g is the gravity acceleration, f_z is the z-component of the ground reaction force, acting as an external input, and m is the total mass of the humanoid robot. The condition for maintaining equilibrium is that CoP is internal to the support polygon. Since the CoP, the CoM and the ZMP are colinear, as shown in Fig. 4.2, the condition is equivalent to the ZMP being internal to the polyhedral cone having CoM as vertex and support polygon as cross-section.

4.2 LIP

The above motion model is nonlinear and difficult to use for gait generation. Usually, to make the model linear, it is assumed that the ground is horizontal and the CoM

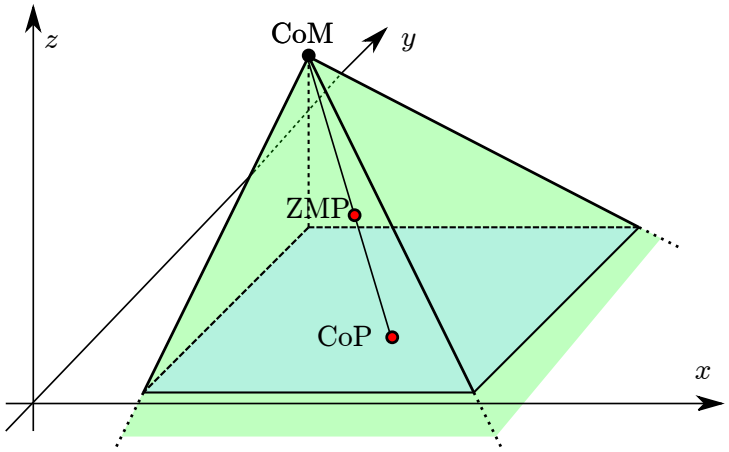


Figure 4.2. The CoP should be internal to the support polygon, which is equivalent to the ZMP being internal to the polyhedral cone with the CoM as a vertex.

has constant elevation with respect to the ground (i.e. $z_c = \bar{z_c}$). As a consequence, it is possible to set $z_z = 0$ making the CoP and the ZMP coincident, hence obtaining the LIP model:

$$\ddot{x}_c = \omega_0^2(x_c - x_z) \tag{4.4}$$

$$\ddot{y}_c = \omega_0^2(y_c - y_z) \tag{4.5}$$

where $\omega_0^2 = g/\bar{z}_c$. This linear model, however, is not suitable for gait generation over uneven terrain.

4.3 Variable Height CoM Motion Model

Requiring the CoM to move at a constant height is not the only way to make the system linear. A more general way is to constraint its vertical motion such that:

$$\frac{\ddot{z}_c + g}{z_c - z_z} = \omega^2 \tag{4.6}$$

with ω arbitrary constant.

Using the above equation, the CoM dynamics become:

$$\ddot{x}_c = \omega^2(x_c - x_z) \tag{4.7}$$

$$\ddot{y}_c = \omega^2(y_c - y_z) \tag{4.8}$$

$$\ddot{z}_c = \omega^2(z_c - z_z) - g \tag{4.9}$$

The above equations are linear and have a LIP-like structure with the ZMP coordinates (x_z, y_z, z_z) acting as control inputs. This 3D model, against the model of

the LIP described in Eqs. (4.4-4.5), allows vertical motion of the CoM, thus, it can be used for gait generation on uneven terrain, considering the balance condition described by Fig. 4.2.

4.4 MPC Formulation

Before describing an MPC scheme for gait generation based on the above equations, it is important to notice that all of them include an unstable subsystem. For example, considering Eq. (4.7), it is possible to decompose the system into a stable and an unstable subsystem by performing the following change of coordinates:

$$x_s = x_c - \dot{x}_c/\omega \tag{4.10}$$

$$x_u = x_c + \dot{x}_c/\omega \tag{4.11}$$

The dynamics of x_u is:

$$\dot{x}_u = \dot{x}_c + \omega(x_c - x_z) \tag{4.12}$$

which is unstable. It is however possible to prove [36] that x_u, and consequently x_c, will not diverge if a certain initial condition is satisfied (discussed in section 4.4.2). The same reasoning applies for the other two variables.

Considering a dynamic extension and choose the control variable as the ZMP velocities $\dot{x}_z, \dot{y}_z, \dot{z}_z$ rather than the ZMP itself, on the x axis, the motion model becomes:

$$\begin{pmatrix} \dot{x}_c \\ \ddot{x}_c \\ \dot{x}_z \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ \omega^2 & 0 & -\omega^2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ \dot{x}_c \\ x_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \dot{x}_z \tag{4.13}$$

The same applies for the other two axes with an additive term g appearing in the second equation of the dynamics along the z axis.

Let the control inputs be piecewise-constant over sampling intervals of duration δ, with a prediction horizon $T_h = N \cdot \delta$, and let t_k be the current time instant and let successive instants within prediction horizon be $t_{k+i}, i = 1, \ldots, N$ by t_{k+i}. At a generic instant t_j:

$$\dot{x}_z(t) = \dot{x}_z^j, \quad t \in [t_j, t_{j+1}) \tag{4.14}$$

hence, the ZMP position along the x axis in the time interval $[t_j, t_{j+1}]$ is:

$$x_z(t) = x_z^j + (t - t_j)\dot{x}_z^j, \quad t \in [t_j, t_{j+1}] \tag{4.15}$$

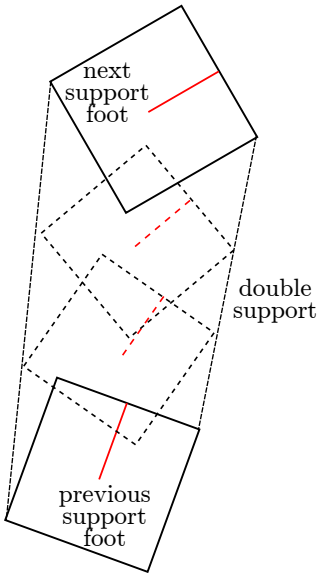


Figure 4.3. The ZMP constraint moves from a support foot to the following one during double support phase.

4.4.1 ZMP constraints

Before describing the ZMP constraints in the 3D case (Fig. 4.2), the 2D case is discussed.

When walking on fully horizontal ground, the robot keeps the equilibrium if the ZMP remains inside the support polygon. Let $(x_f^j, y_f^j, \theta_f^j)$ be the pose of the generic footstep within the given sequence. Let the ZMP constraint be fixed-shape moving [37] to enforce balance. The admissible region for ZMP at t_{k+i} is centered in (x_f^{k+i}, y_f^{k+i}) and has orientation θ_f^{k+i}. In single support case, $(x_f^{k+i}, y_f^{k+i}, \theta_f^{k+i})$ coincide with the pose of the support foot, hence, $(x_f^j, y_f^j, \theta_f^j)$. In double support case, $(x_f^{k+i}, y_f^{k+i}, \theta_f^{k+i})$ gradually slide from the position and orientation of the previous support foot to those of the next, as shown in Fig. 4.3. The expression of the ZMP constraint in 2D can be written as:

$$-\frac{1}{2}\begin{pmatrix} d_x^z \\ d_y^z \end{pmatrix} \leq R_{k+i}^T \begin{pmatrix} x_z^{k+i} - x_f^{k+i} \\ y_z^{k+i} - y_f^{k+i} \end{pmatrix} \leq \frac{1}{2}\begin{pmatrix} d_x^z \\ d_y^z \end{pmatrix} \tag{4.16}$$

where d_x^z and d_y^z are the width and the height of the rectangular constraint region and R_{k+i}^T is the rotation matrix associated to the orientation θ_f^{k+i}. Note that (x_z^{k+i}, y_z^{k+i}) is the predicted position of the ZMP, which can be expressed as a linear combination of the control variables:

$$x_z^{k+i} = x_z^k + \delta \sum_{i=0}^{i-1} \dot{x}_z^{k+j} \tag{4.17}$$

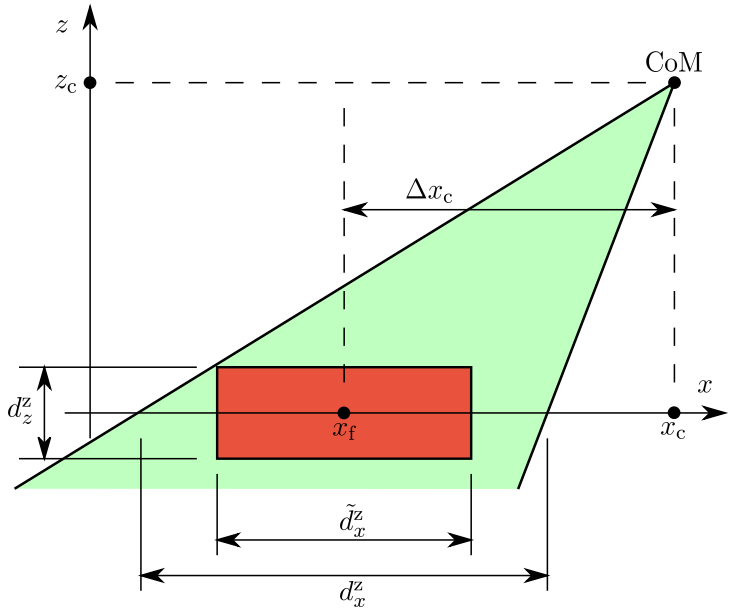


Figure 4.4. The polyhedral cone representing the ZMP constraint and the box used to approximate the constraint (which becomes linear).

Eq. (4.16) must be imposed for $i = 1, \ldots, N$.

In the 3D case, the ZMP is allowed to leave the ground in order to generate CoM motions along the z axis as well. As previously discussed, balance condition requires ZMP to remain inside the polyhedral cone defined by the support polygon and the CoM (Fig. 4.2). When the ZMP is allowed to move vertically, the constraint becomes nonlinear. In order to remove nonlinearity it is possible to consider a subregion of the polyhedral cone, for example a box, as shown in Fig. 4.4:

$$-\frac{1}{2}\begin{pmatrix}\tilde{d}_x^z\\ \tilde{d}_y^z\\ d_z^z\end{pmatrix} \leq R_{k+i}^T \begin{pmatrix}x_z^{k+i} - x_f^{k+i}\\ y_z^{k+i} - y_f^{k+i}\\ z_z^{k+i} - y_f^{k+i}\end{pmatrix} \leq \frac{1}{2}\begin{pmatrix}\tilde{d}_x^z\\ \tilde{d}_y^z\\ d_z^z\end{pmatrix} \tag{4.18}$$

where d_z^z defines the maximum allowed vertical ZMP displacement with respect to the ground. To guarantee that the box is contained in the cone, its x and y dimensions are respectively reduced to $\tilde{d}_x^z, \tilde{d}_y^z$:

$$\tilde{d}_x^z = d_x^z\left(1 - \frac{d_z^z}{2z_c^{\min}}\right) - \frac{d_z^z}{z_c^{\min}}\Delta x_c \tag{4.19}$$

where Δx_c is the maximum expected displacement of the CoM with respect to the center of the support foot and $z_c^{\min}$ is the minimum expected value for CoM height. The same reasoning applies for $\tilde{d}_y^z$. Similarly to the 2D case, the box constraint is kept fixed during single support, while during double support the box slides linearly

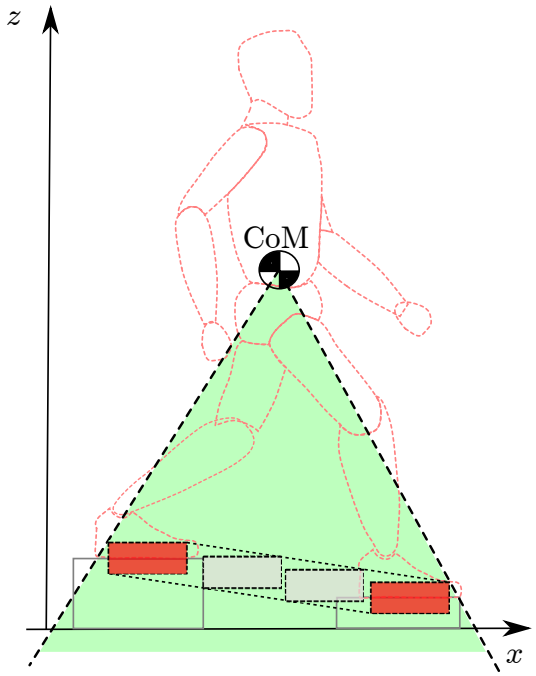


Figure 4.5. The ZMP box constraint moves from a support foot to the following one during double support phase.

from its position around the previous support foot to its position around the next support foot, thus, always remaining within the polyhedral cone which defines the ZMP balance constraint, as shown in Fig. 4.5.

4.4.2 Stability constraint

As previously seen, the motion model (4.7-4.9) is unstable, hence, it is not guaranteed that the CoM position is bounded with respect to the ZMP in general. This could make the generated gait unrealizable. However, as mentioned before and as discussed in [36, 24], it is possible to prove that if the initial condition $(x_c^k, \dot{x}_c^k)$ satisfies:

$$x_c^k + \frac{\dot{x}_c^k}{\omega} = \omega \int_{t_k}^{\infty} e^{-\omega(\tau - t_k)} x_z(\tau) d\tau \tag{4.20}$$

then x_c remains bounded with respect to x_z for all t. An analogous condition can be given for y_c dynamics. Regarding z_c, it is possible to prove its boundedness by using the following initial condition:

$$z_c^k + \frac{\dot{z}_c^k}{\omega} = \frac{g}{\omega^2} + \omega \int_{t_k}^{\infty} e^{-\omega(\tau - t_k)} z_z(\tau) d\tau \tag{4.21}$$

The above stability conditions can be enforced in the MPC formulation by writing them with respect to the control variables $\dddot{x}_z^{k+i}, \dddot{y}_z^{k+i}, \dddot{z}_z^{k+i}$. For x_c, and similarly for

y_c, the initial condition can be writtes as:

$$\frac{1}{\omega}\frac{1-e^{-\delta\omega}}{1-e^{-N\delta\omega}}\sum_{i=0}^{N-1}e^{-i\delta\omega}\dot{x}_z^{k+i} = x_c^k + \frac{\dot{x}_c^k}{\omega} - x_z^k \tag{4.22}$$

which can be obtained from (4.20) by considering that the ZMP trajectory (4.17) is piecewise linear and that the contribution beyond the prediction horizon is computed assuming infinite replication of the control variables within the prediction horizon itself. A similar initial condition can be written for z_c starting from (4.21), where $\dot{z}_z$ is set to zero beyond the prediction horizon (truncated tail):

$$\frac{1-e^{-\delta\omega}}{\omega}\sum_{i=0}^{N-1}e^{-i\delta\omega}\dot{z}_z^{k+i} = z_c^k + \frac{\dot{z}_c^k}{\omega} - z_z^k - \frac{g}{\omega^2} \tag{4.23}$$

A more detailed discussion can be found in [38].

4.5 MPC Algorithm

Now that the constraints have been expressed with respect to the input variables, it is possible to define the MPC scheme used to generate the gait. In particular, the MPC algorithm solves a QP problem at each iteration determining the trajectory of the CoM. Note that the footsteps are assigned in advance.

Considering the decision variable vectors:

$$\dot{X}_{\mathrm{z}}^k = (\dot{x}_{\mathrm{z}}^k, \ldots, \dot{x}_{\mathrm{z}}^{k+N-1})^T \tag{4.24}$$

$$\dot{Y}_{\mathrm{z}}^k = (\dot{y}_{\mathrm{z}}^k, \ldots, \dot{y}_{\mathrm{z}}^{k+N-1})^T \tag{4.25}$$

$$\dot{Z}_{\mathrm{z}}^k = (\dot{z}_{\mathrm{z}}^k, \ldots, \dot{z}_{\mathrm{z}}^{k+N-1})^T \tag{4.26}$$

the QP problem can be defined as:

$$\begin{aligned}\min_{\dot{X}_{\mathrm{z}}^k, \dot{Y}_{\mathrm{z}}^k, \dot{Z}_{\mathrm{z}}^k} \sum_{i=1}^{N} \Big[&(\dot{x}_z^{k+i})^2 + (\dot{y}_z^{k+i})^2 + (\dot{z}_z^{k+i})^2 + \\ &\beta\Big((x_z^{k+i} - x_f^{k+i})^2 + (y_z^{k+i} - y_f^{k+i})^2 + (z_z^{k+i} - z_f^{k+i})^2\Big)\Big]\end{aligned}$$

s.t. ZMP constraint (4.18)

stability constraints (4.22), (4.23)

where the cost function includes the decision variables for regularization purposes and a term which attempts to bring the ZMP to the center of the footstep.

Each MPC iteration starts at t_k and executes the steps described in Algorithm 2, defining the trajectory of the CoM.

Algorithm 2: MPC iteration

Result: CoM trajectory

1 Compute $\dot{X}_{\mathrm{z}}^k$, $\dot{Y}_{\mathrm{z}}^k$, $\dot{Z}_{\mathrm{z}}^k$ that solve the QP problem;

2 From the solutions, extract the first control samples $\dot{x}_{\mathrm{z}}^k$, $\dot{y}_{\mathrm{z}}^k$, $\dot{z}_{\mathrm{z}}^k$;

3 Set $\dot{x}_z = \dot{x}_{\mathrm{z}}^k$ in (4.13) and integrate from $(x_c^k, \dot{x}_c^k, x_{\mathrm{z}}^k)$ to obtain $x_c(t)$, $\dot{x}_c(t)$, $x_{\mathrm{z}}(t)$ for $t \in [t_k, t_{k+1}]$. The same applies for for y, z.

4.6 BHuman Implementation

The MPC scheme has been implemented in C++ upon the BHuman framework [27] and it has been tested on both dynamic environments and NAO humanoid robot. The QP problem has been solved using qpOASES [39]. The footstep plan is either generated by the footstep planner described in Chapter 3 or manually assigned before the execution of the program. Experiments are described in detail in Chapter 5. Note that to speed up the execution of the code in order to keep computation within the sampling time of the kinematic controller, the rotation matrix of the ZMP constraint described in Eq. (4.18) has been neglected. This does not create any problem if the size of the box is small enough to stay within the poyhedral cone regardless of the rotation of the feet. The MPC hyperparameters have been set in the following way: $\omega = 6.68\mathrm{s}^{-1}$, the step duration $T_s = 0.48\mathrm{s}$ of which $t_{\mathrm{SS}} = 0.30\mathrm{s}$ of single support and $t_{\mathrm{DS}} = 0.18\mathrm{s}$ of double support, $T_h = 0.96$, $N = 16$. The size of the box constraint have been set to $\tilde{d}_x^z = 0.05\mathrm{m}, \tilde{d}_y^z = 0.05\mathrm{m}, \tilde{d}_z^z = 0.05\mathrm{m}$.

Chapter 5

Experiments

To test the behaviour of the whole project, multiple experiments have been performed. In particular, different scenario and test cases have been designed in order to study in detail the functionalities of the robot and each program. As previously introduced, the humanoid robot used for the experiments of this thesis is NAO [5], which has been equipped with an ASUS Xtion Pro RGB-D camera [32], placed on top of its head. As mentioned in the first chapter, the camera communicates with `elevation_mapping` (Chapter 2), which creates a representation of the environment called elevation map, which is used by the footstep planner, described in Chapter 3, to generate a safe and feasible plan for the robot. The footsteps are sent to the robot, which runs an implementation of the Variable Height CoM Intrinsically Stable MPC (Chapter 4) that allows the robot to correctly perform the desired motion, making it able to navigate in a *World of Stairs* environment.

The experiments have been performed by initially checking the real capabilities of the robot, making it first climb a single staircase and then a stairway composed of multiple staircases, in order to better understand the kind of scenario that NAO can handle. At this point only the MPC is used, assigning the footsteps manually. Then, the footstep planner is introduced into the project, making NAO navigate autonomously from its initial position to a goal region, given an elevation map of the environment which has been manually created. In the end, NAO is equipped with an ASUS Xtion Pro camera. This allows to introduce `elevation_mapping` to the project, generating autonomously elevation maps, making NAO able to move in a *World of Stairs* environment which is not known a priori, correctly climbing the stairs in order to reach a desired position.

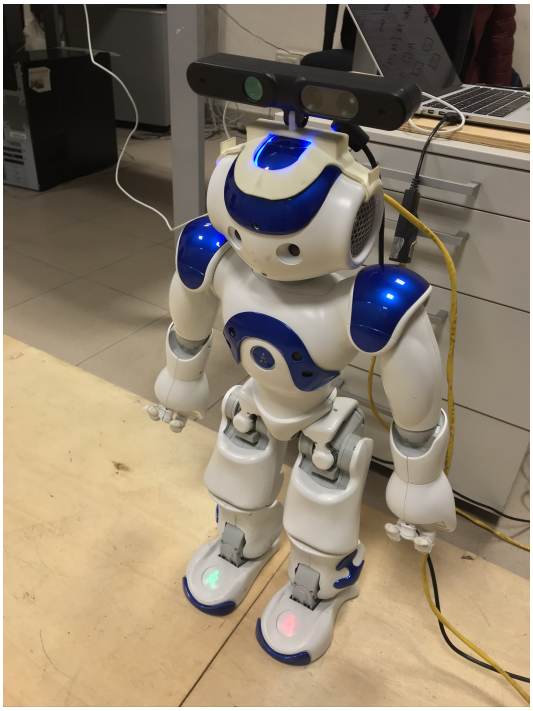


Figure 5.1. The figure shows NAO v5 humanoid robot, the platform used for the experiments of this thesis, equipped with an ASUS Xtion Pro RGB-D camera. The camera is connected via USB to an external computer which runs `elevation_mapping` and the footstep planner. The external computer communicates with NAO through an ethernet cable using TCP.

5.1 NAO and Computer Settings

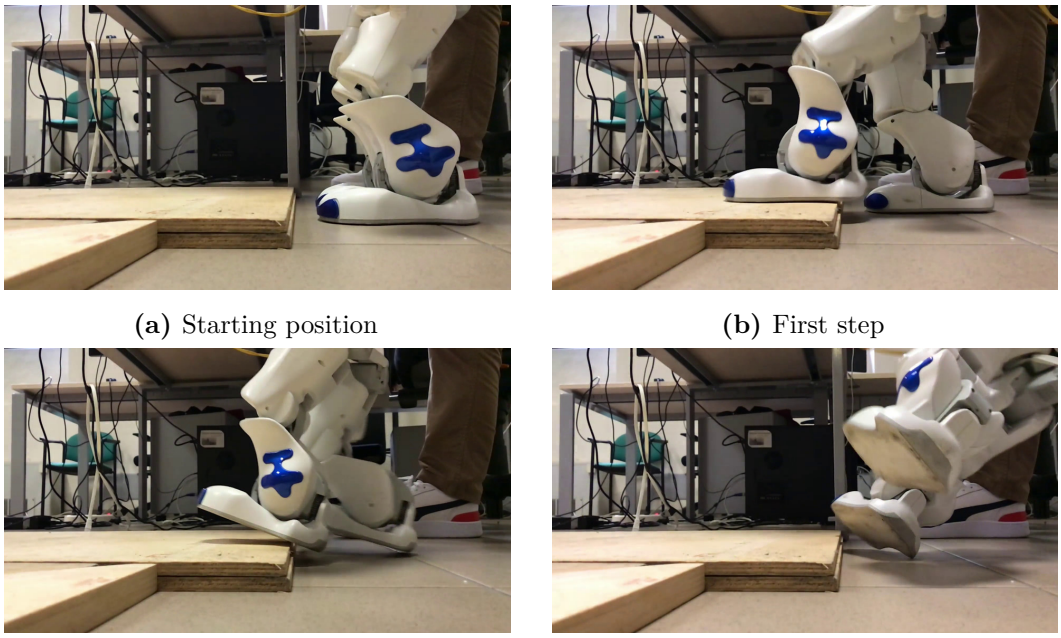
The NAO v5 humanoid robot (Fig: 5.1), developed by Softbank, is 57 cm tall and weights 5.4 kg. It is equipped with an Intel ATOM Z530 processor which has a frequencly of 1.6 Ghz, 1 GB of RAM and 2 GB of flash storage. It has 25 DoF, 7 touch sensors located on the head, the hands and the feet, two internal cameras, 4 microphones, a speaker, sonar and IMU sensors. The framework used for the experiments, which has been installed on the robot, is BHuman [27], which simplifies the development of on board algorithms by organizing the framework in modules. When using the full settings, NAO is equipped with an ASUS Xtion Pro RGB-D camera, which is connected to an external computer via a USB cable. This external computer communicates with NAO via an ethernet cable through TCP. Both `elevation_mapping` and the footstep planner run on the external computer, which is equipped with an Intel Core i5-5257U with 2.70 GHz base frequency and 16 GB of RAM. The Variable Height CoM IS-MPC runs on the robot at 100 Hz.

5.2 Simple Staircase

As mentioned before, the aim of the first experiment is to understand the kind of scenario that the robot can handle. In this first setting, only the robot with the Variable Height CoM IS-MPC is used. The planner and the mapper will be introduced later. The footsteps are, hence, assigned manually.

The first scenario, called "Normal Staircase" consists in placing the robot in front of a staircase of 2 cm in order to make the robot climb it. To consider the experiment successful, NAO must climb the staircase putting the whole foot on the ground at each step. Increasing the length of the footstep increases the footprint area that is in contact with the ground at each step during climbing. However, while shorter footsteps are easier to perform, longer footsteps need to deal with the kinematic limitations of the robot. It is clear from the experiment (Fig. 5.2) that the larger the footstep is, the harder it is for the robot to make a step. In particular, in Fig. 5.2a the robot performs a footstep of size 16 cm; the foot is not placed entirely on the staircase (Fig. 5.2b), causing the robot to be unstable. This causes the robot to slip (Fig. 5.2c) while attempting to do the second step. In the end, the robot falls on the floor (Fig. 5.2d) failing the assigned task. A longer footstep is, hence, needed in order to correctly perform the motion. However, due to the short legs of the robot, longer footsteps cause the robot to immediately fall. This suggests that NAO is not able to climb this kind of stairway due to its kinematics and that a special stairway needs to be built in order to test the behaviour of the Variable Height CoM IS-MPC.

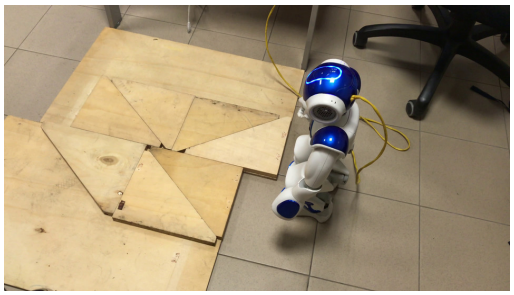
The scenario "Simple Staircase" is similar to the previous one. The only exception is the structure of the stairs, which has been put as in Fig. 5.3 in order to give NAO some space to place the foot before climbing the staircase. Placing the staircases in this way allows the robot to perform climbing with shorter footsteps, avoiding kinematic limitations and correctly making it complete the assigned task. With these settings, NAO is able to climb staircases up to 4 cm. Fig. 5.3 shows NAO climbing a staircase of 3 cm. Fig. 5.4 shows how the CoM and the ZMP varies through time with respect to the position of the support foot, when dealing with a staircase of 4 cm. It is important to notice that both the CoM and the ZMP change their height when the robot performs a step. This is possible because of the the Variable Height CoM IS-MPC described in Chapter 4, which allows the two variables to move along the z axis while keeping a linear motion model like the Linear Inverted Pendulum [24]. This characteristic is important not just in terms of the analysis of the system, but also in terms of computational resources, which can be kept low as for the case of NAO.



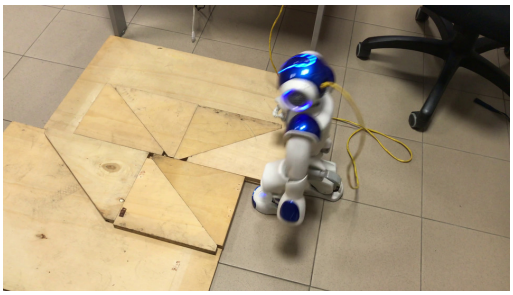
(a) Starting position **(b)** First step

(c) Second step **(d)** Robot falling

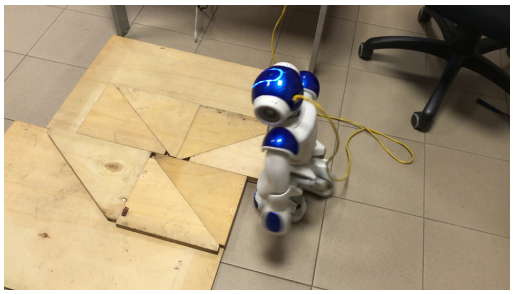
Figure 5.2. The figures show the motion of the robot for the scenario "Normal Staircase". The robot starts just in front of the stairs (Fig. 5.2a). The motion starts by placeing the first footstep on the staircase (Fig. 5.2b). Here, it is already clear that the motion could be unsuccessful. The foot is, in fact, not entirely in contact with the staircase. Longer footsteps make the robot immediately fall. Shorter footsteps are safer but fail to correctly place the foot. In Fig. 5.2c the robot attempts to place the other foot on the platform but it falls (Fig. 5.2d), making the robot fail its task. Each staircase has a height of 2 cm.



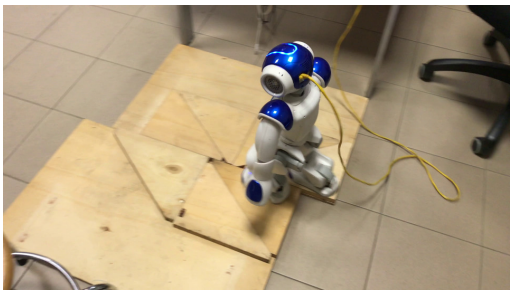
(a) Starting position



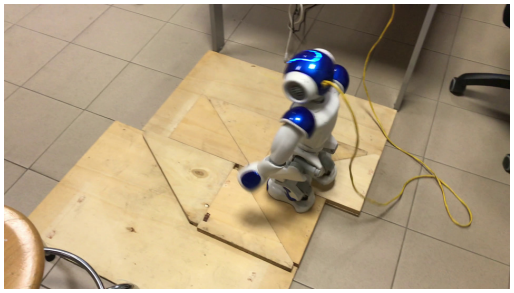
(b) First step



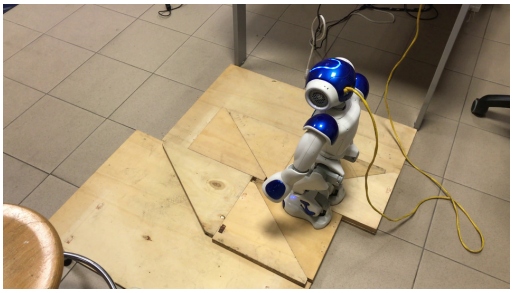
(c) Second step



(d) Third step



(e) Fourth step



(f) Final position

Figure 5.3. The figures show the motion of the robot for the scenario "Simple Staircase (3 cm)". The robot start just in front of the stairs (Fig. 5.3a), then it places each step one in front of the other without colliding with the staircases, safely climbing the stairway. Each staircase has a height of 3 cm.

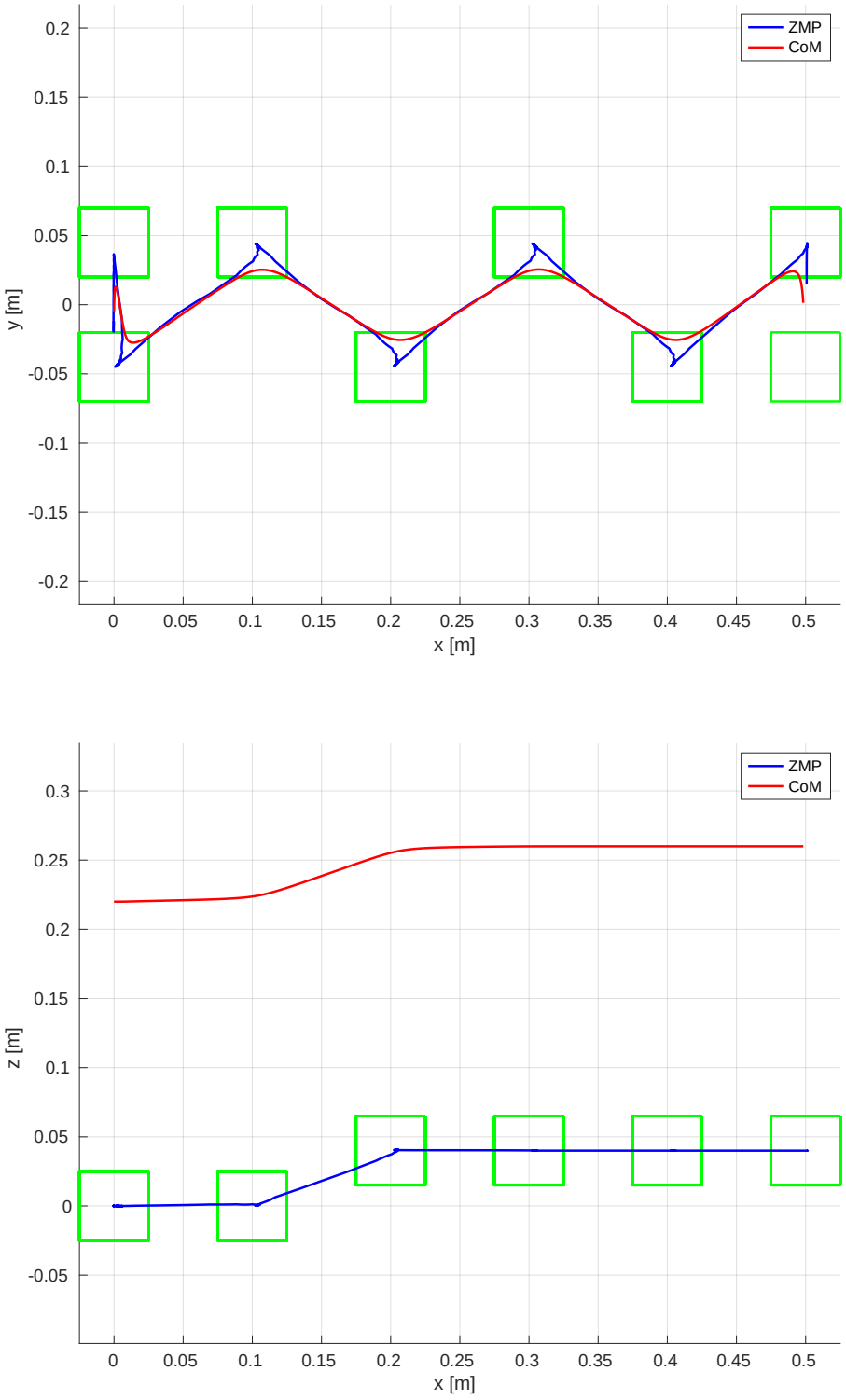


Figure 5.4. The plots show how the CoM and the ZMP vary with respect to the footsteps in the scenario "Simple Staircase (4 cm)". The green boxes represent the footsteps.

5.3 Multiple Staircases

This second experiment consists in making NAO climb a stairway composed of two staircases of 2 cm both upstairs and downstairs. The sequence of footsteps is manually assigned as in the previous experiment.

5.3.1 Upstairs

The following scenario is called "Multiple Staircases (Upstairs)" and it consists in making NAO climb a stairway composed of two consecutive staircases of 2 cm by moving upstairs. Fig. 5.5 shows the behaviour of the robot for this setting. NAO correctly completes the assigned task reaching the top of the stairway. Fig. 5.6 show how the CoM and the ZMP changes through time during this experiment. Again, it is important to notice how both variables change in the z axis, allowing the robot to safely climb the stairway.

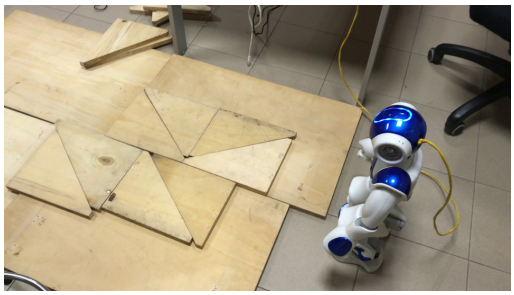
5.3.2 Downstairs

The following scenario is called "Multiple Staircases (Downstairs)" and it consists in making NAO climb a stairway composed of two consecutive staircases of 2 cm by moving downstairs. Fig. 5.7 shows the behaviour of the robot for this setting. NAO correctly completes the assigned task reaching ground level. Fig. 5.8 show how the CoM and the ZMP changes through time during this experiment.

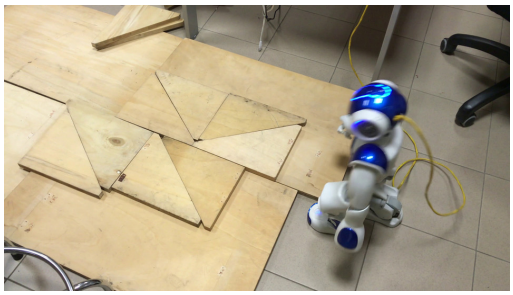
5.4 Obstacle Avoidance

The third experiment introduces the footstep planner described in Chapter 3 into the project. The aim is to test the behaviour of the planner in a simple *World of Stairs* scenario that contains a platform put between the initial position of the robot and the goal region that can not be climbed by NAO. The goal of the footstep planner is those of finding a feasible plan not just avoiding the obstacle, but also by not stepping onto it. The robot would, in fact, not climb it correctly because of its physical limitations. The elevation map used in this experiment has been manually generated.

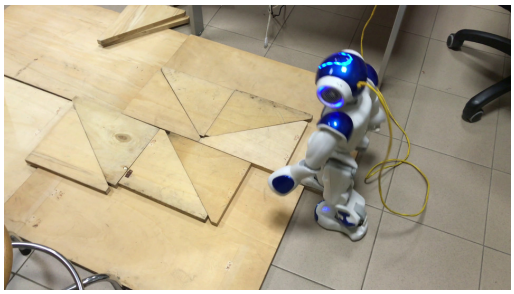
The planner generates a plan of 31 steps (Fig. 5.10) in 70 ms (Table 5.1). The corresponding tree in shown in Fig. 5.11 and it has size 488. Fig. 5.9 shows the robot moving inside the environment from its initial position to the goal region (a circle of radius 10 cm with center at the position of the ball).



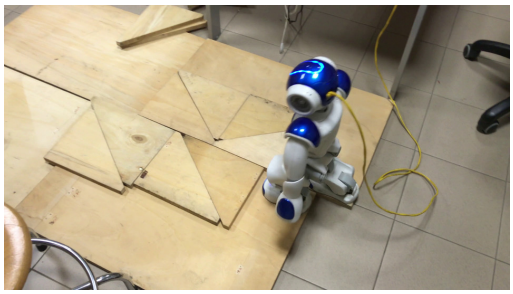
(a) Starting position



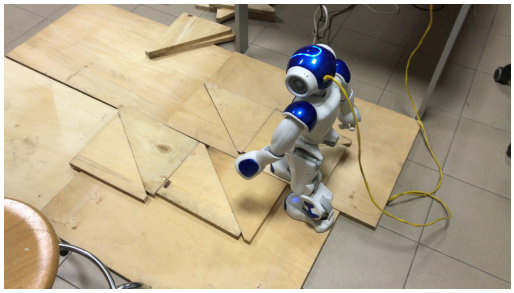
(b) First step



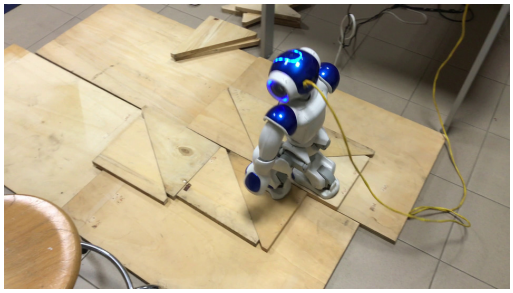
(c) Second step



(d) Third step



(e) Fourth step



(f) Fifth step

Figure 5.5. The figures show the motion of the robot for the scenario "Multiple Staircases (Upstairs)". The robot starts just in front of the stairs (Fig. 5.5a), then it places each step one in front of the other without colliding with the staircases, safely climbing the stairway. Each staircase has a height of 2 cm.

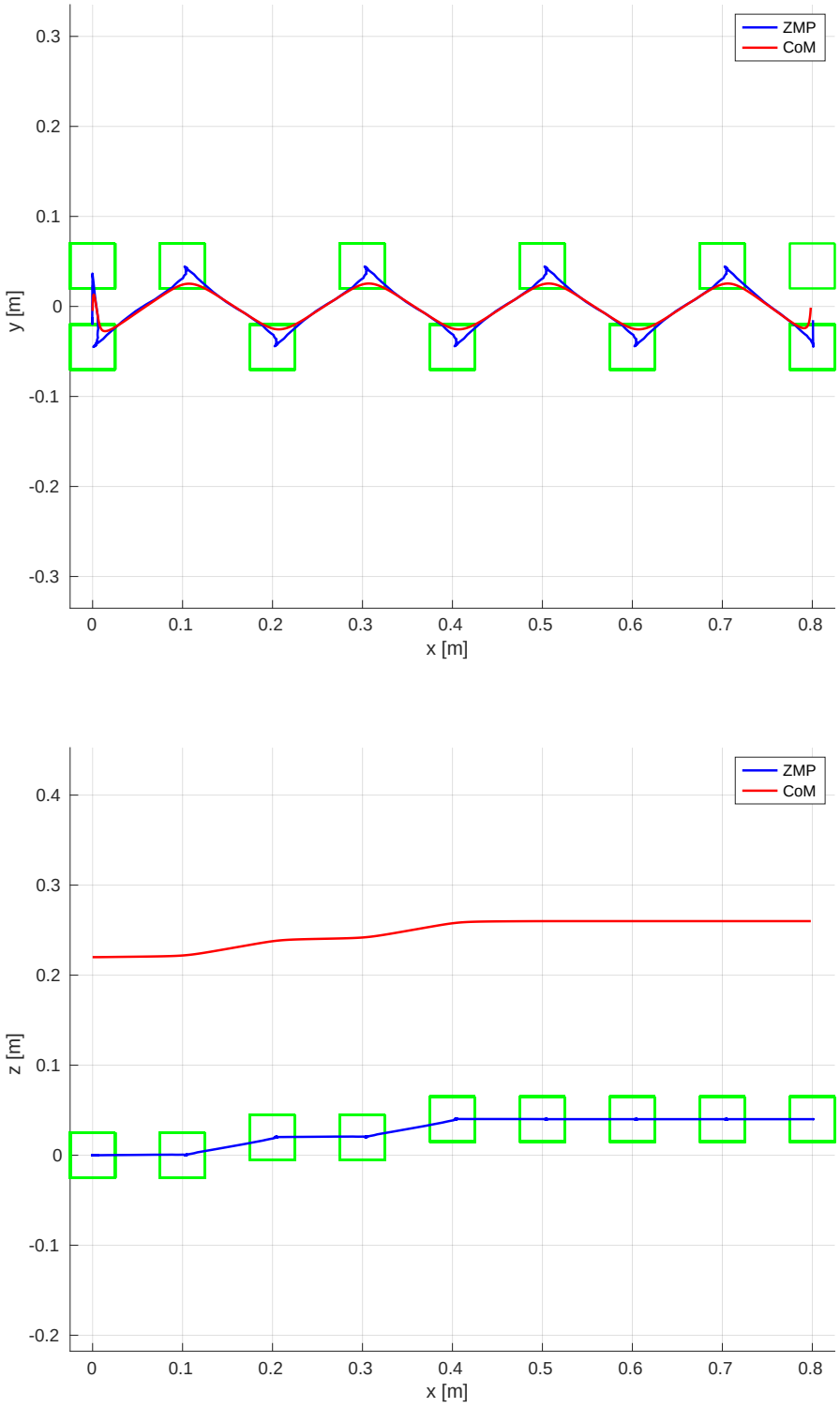
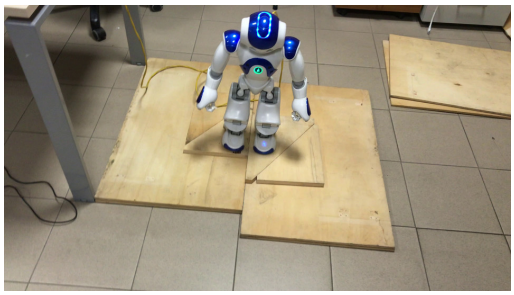
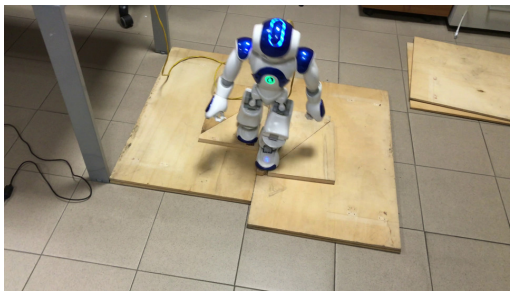


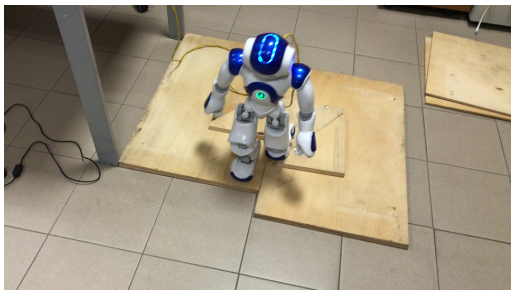
Figure 5.6. The plots show how the CoM and the ZMP vary with respect to the footsteps in the scenario "Multiple Staircases (Upstairs)". The green boxes represent the footsteps.



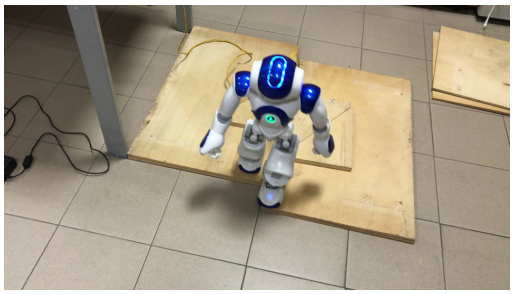
(a) Starting position



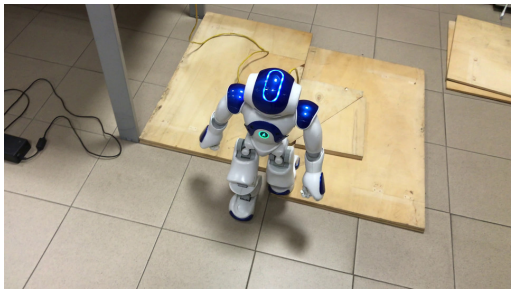
(b) First step



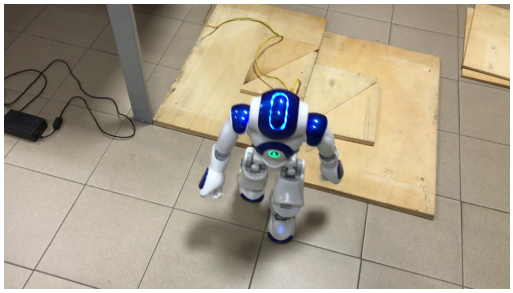
(c) Second step



(d) Third step



(e) Fourth step



(f) Fifth step

Figure 5.7. The figures show the motion of the robot for the scenario "Multiple Staircases (Downstairs)". The robot starts on top of the stairway (Fig. 5.7a), then it places each step one in front of the other without colliding with the staircases, safely reaching ground level. Each staircase has a height of 2 cm.

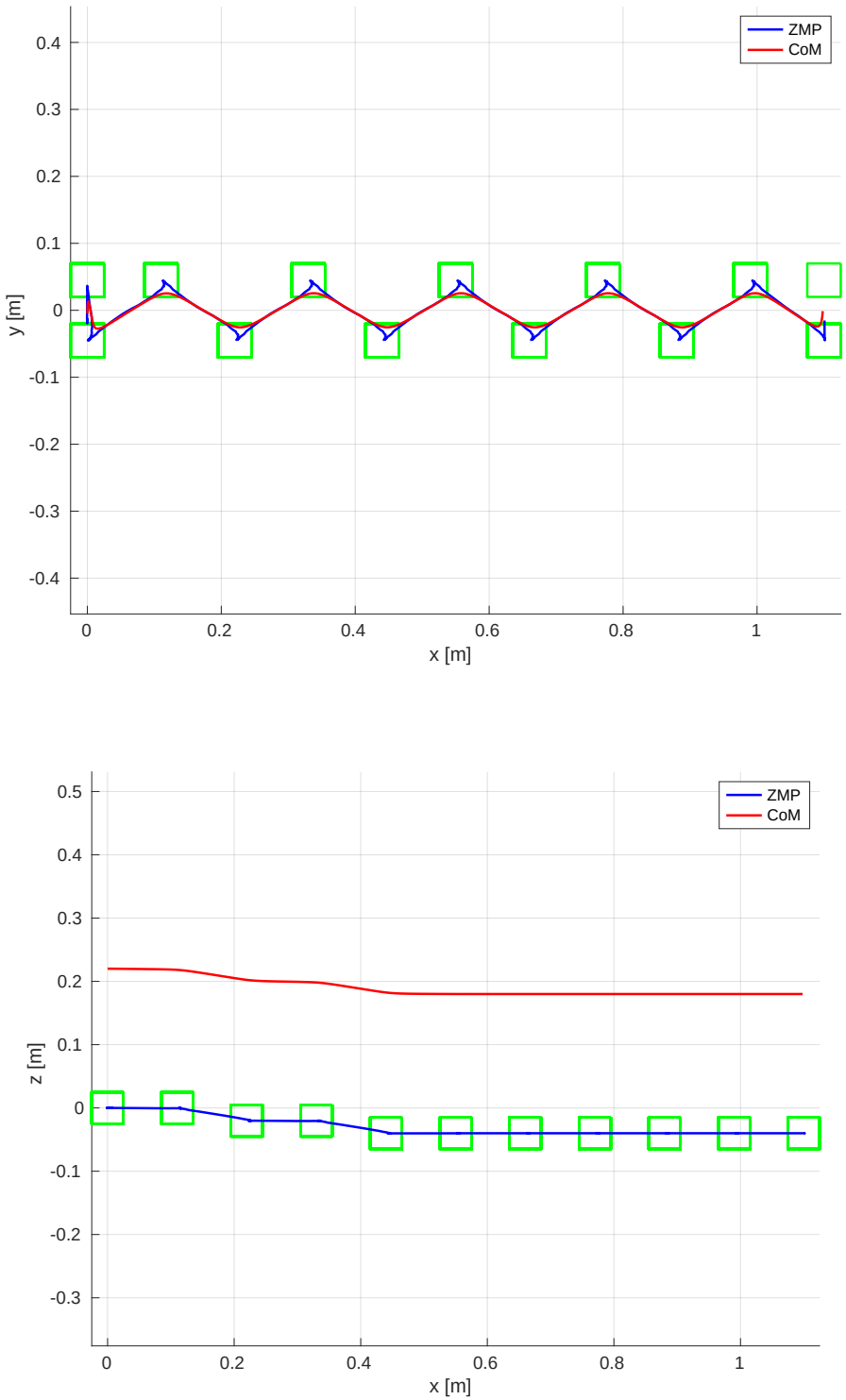


Figure 5.8. The plots show how the CoM and the ZMP vary with respect to the footsteps in the scenario "Multiple Staircases (Downstairs)". The green boxes represent the footsteps.

Scenario	Tree Size	Solution Size	Runtime
Obtacle Avoidance	488	31	70 ms
Stair Climbing in Unknown Environment	454	10	331 ms

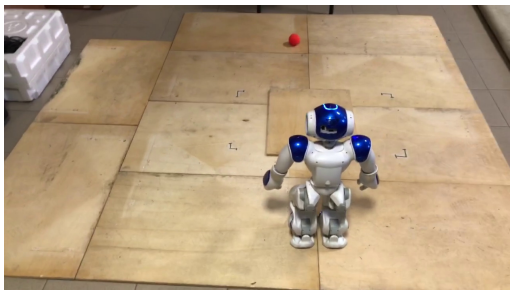
Table 5.1. Performances of the planner on different scenarios. Note that even if the size of the trees are similar, the runtime of "Stair Climbing in Unknown Environment" is slower due to noise in the elevation map. All the experiment have been performed on an Intel Core i5-5257U @ 2.70 Ghz.

Since the planner is based on RRT, the generated plan is not optimal. It is, in fact, possible to notice how the robot does not move fluidly, sometimes putting the foot back and forth before changing position. This is a characteristic of RRT that could be changed by building the algorithm on top of RRT* [40].

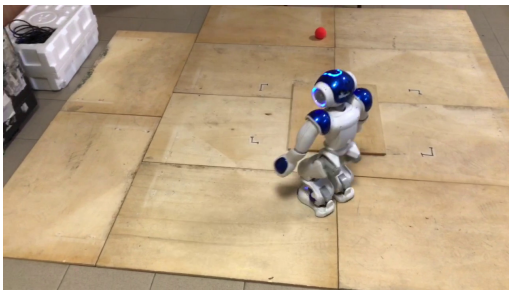
5.5 Stair Climbing in Unknown Environment

The last experiment introduces `elevation_mapping` (Chapter 2) into the project. The aim is to test the behaviour of the framework in an unknown *World of Stairs* environment. The idea is to generate a map autonomously exploiting the depth sensor of the ASUS Xtion Pro camera, which has been put on top of NAO humanoid robot. The generated map is sent to the footstep planner described in Chapter 3 which is in charge of generating a plan that brings the robot to a predefined goal region. This goal region is placed on top of a stairway, so that the robot must climb the stairs in order to complete the task. The generation of the map has been already described in Chapter 2 and it is shown in Fig. 2.5. As already discussed, once the footstep plan is generated, it is sent to the robot which executes a proper motion using the Variable Height CoM IS-MPC (Chapter 4).

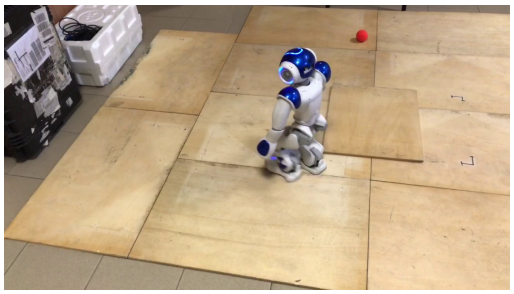
The planner generates a plan of 10 steps (Fig. 5.13) in 0.331s (Table 5.1). Note that the runtime of this experiment is slower than the previous one because of the noise contained in the elevation map, which causes the planner to discard many configurations since they do not satisfy requirement R2 of the footstep planner (Chapter 3). In this experiment the requirement R2 has been relaxed allowing a maximum difference of 7 mm between the z coordinate of the footstep and the height of a cell in the map. The corresponding tree is shown in Fig. 5.14 and it has size 454. Fig. 5.12 shows the robot moving inside the environment from its initial position to the goal region (a circle of radius 10 cm with center at the position of the ball).



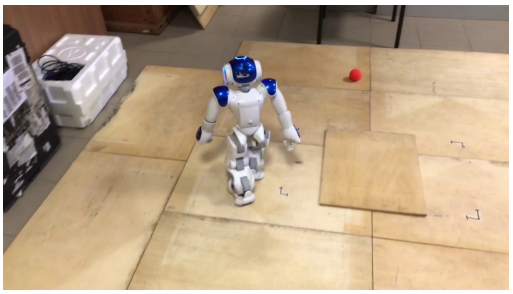
(a) Starting position



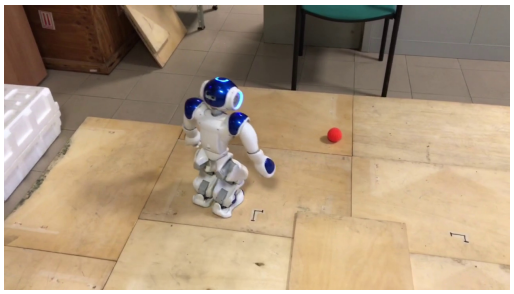
(b) Moving to the left



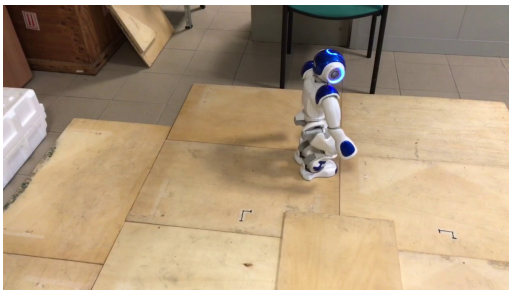
(c) Avoiding the obstacle



(d) Moving forward



(e) Moving towards the goal



(f) Goal region reached

Figure 5.9. The figures show the motion of the robot for the scenario "Obstacle Avoidance". The robot starts in front of the obstacle (Fig. 5.9a), then it moves to its left correctly avoiding it (Fig. 5.9b-5.9e) until it reaches the goal region (Fig. 5.9f), whose center is represented by the ball. The goal region is a circle with a radius of 10 cm. Note that even if the obstacle has a height of 2 cm, NAO can not climb it because of its kinematic limitations, hence, the planner takes into account the mechanical capabilities of the robot, generating a safe and realizable footstep plan.

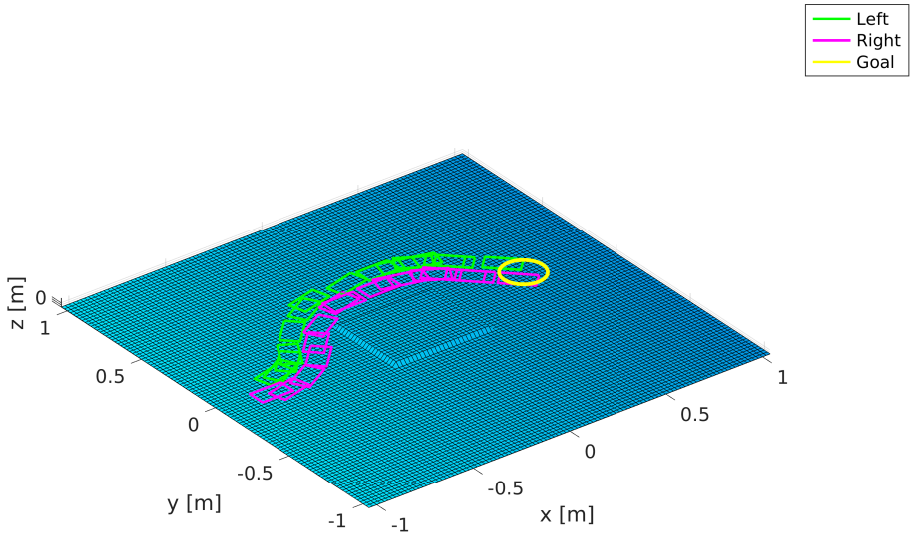


Figure 5.10. Footstep plan generated for the scenario “Obstacle Avoidance”.

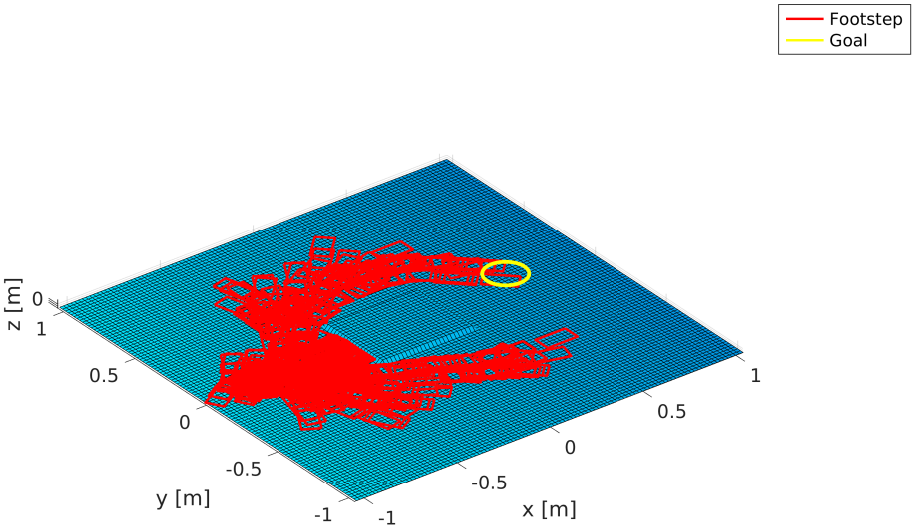
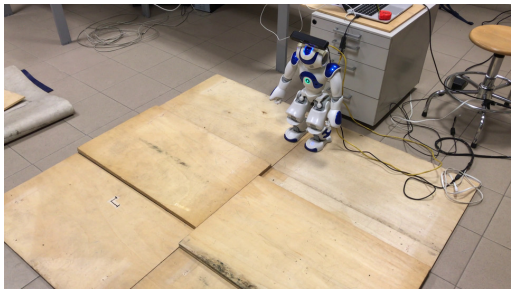
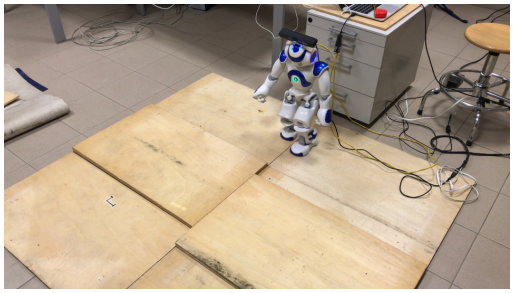


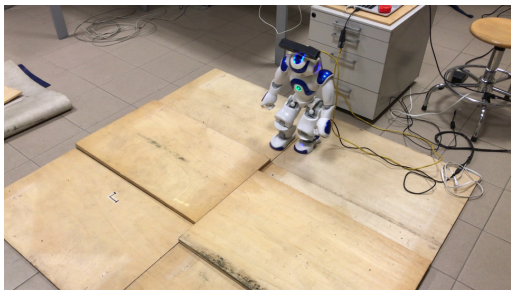
Figure 5.11. Tree generated for the scenario “Obstacle Avoidance”.



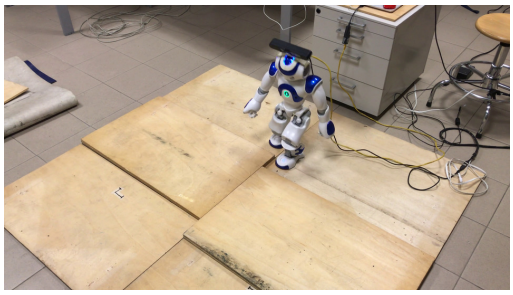
(a) Starting position



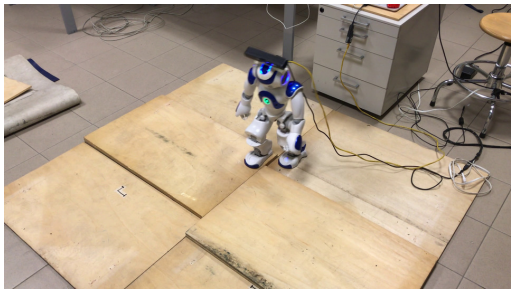
(b) First step



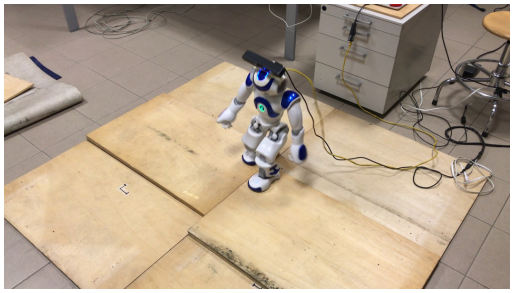
(c) Second step



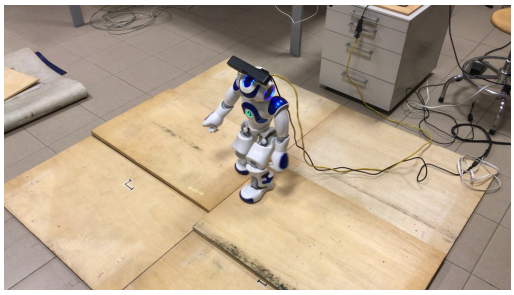
(d) Third step



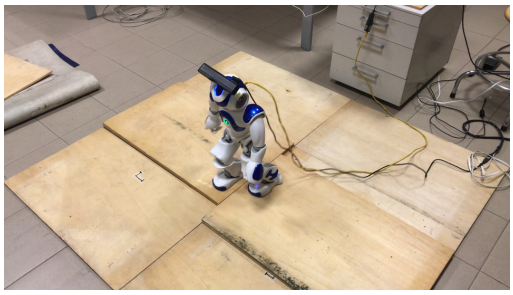
(e) Fourth step



(f) Fifth step



(g) Seventh step



(h) Eighth step

Figure 5.12. The figures show the motion of the robot for the scenario "Stair Climbing in Unknown Environment". The robot starts at 20 cm from the first staircase (Fig. 5.12a). Before starting the motion (Fig. 5.12b), an elevation map is built by the `elevation_mapping` framework (Chapter 2), which receives the depth frames from the ASUS Xtion Pro placed on top of the robot. The elevation map is sent to the planner, which generates a footstep plan (Chapter 3). The footstep plan is then sent to the robot, which executes the motion by using the Variable Height CoM IS-MPC (Chapter 4). The robot correctly manages to climb the stairs without colliding with the staircases. Each staircase has a height of 2 cm.

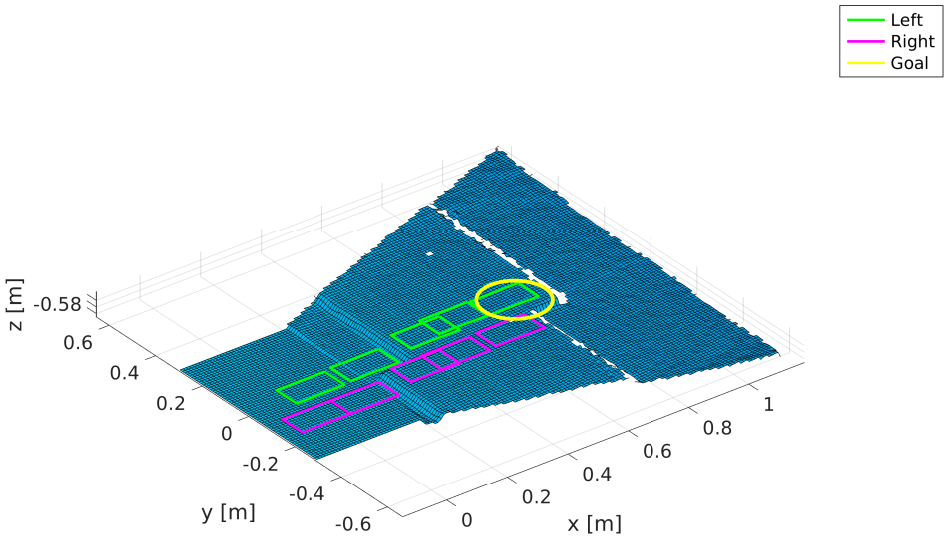


Figure 5.13. Footstep plan generated for the scenario "Stair Climbing in Unknown Environments".

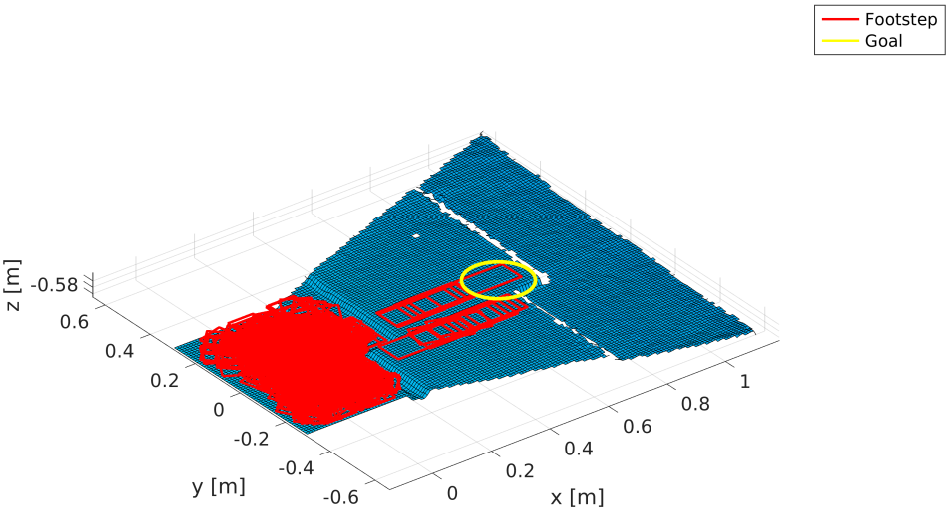


Figure 5.14. Tree generated for the scenario “Stair Climbing in Unknown Environments”.

Chapter 6

Conclusion

6.1 Results

This thesis presents an architecture that integrates mapping, planning and control to generate humanoid gaits in *World of Stairs* unknown environments. The use of `elevation_mapping` [25] together with a depth sensor allows the robot to represent the surrouding environment as a map that can be used by a RRT-based footstep planning module [2] to generate a sequence of footsteps. The Variable-Height CoM IS-MPC [26] has been implemented on NAO humanoid robot upon the BHuman framework and the whole architecture has been tested on multiple scenarios.

6.2 Future Works

The current architecture does not include a localization module, limiting the potentialities of the robot. Localizing the robot inside the environment would, in fact, provide a precise configuration of the humanoid, enabling `elevation_mapping` to continuously build the map during locomotion. A possible extension could be to develop such module using a Kalman filter [41] or a factor graph [42]. Another possible extension of this work could be that of developing a replanning phase [43], allowing the robot to work in dynamic environments. The combination of these two could give even more autonomy to humanoid robots, further advancing current technology and allowing their introduction into our society.

Bibliography

[1] Michael Moran. The da vinci robot. *Journal of endourology / Endourological Society*, 20:986–90, 01 2007.

[2] Paolo Ferrari, Nicola Scianca, Leonardo Lanari, and Giuseppe Oriolo. An integrated motion planner/controller for humanoid robots on uneven ground. In *18th European Control Conference, ECC 2019, Naples, Italy, June 25-28, 2019*, pages 1598–1603, 2019.

[3] Ichiro Kato. The "wabot-1" an information-powered machine with senses and limbs. 1973.

[4] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo FujiMura. The intelligent asimo: System overview and integration. volume 3, pages 2478 – 2483 vol.3, 02 2002.

[5] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jérôme Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, and Bruno Maisonnier. Mechatronic design of nao humanoid. pages 769 – 774, 06 2009.

[6] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 340–347, New York, NY, USA, 1997. Association for Computing Machinery.

[7] Giulio Sandini, Giorgio Metta, and David Vernon. *The ICub Cognitive Humanoid Robot: An Open-System Research Platform for Enactive Cognition*, pages 358–369. Springer-Verlag, Berlin, Heidelberg, 2007.

[8] Christopher Atkeson, Benzun Pious Wisely Babu, Nandan Banerjee, Dmitry Berenson, Christoper Bove, Xiongyi Cui, Mathew Dedonato, Ruixiang Du, Siyuan Feng, Perry Franklin, M. Gennert, Joshua Graff, Peng He, Aaron Jaeger, Joohyung Kim, Kevin Knoedler, Lening Li, Chenggang Liu, Xianchao Long,

and X. Xinjilefu. *What Happened at the DARPA Robotics Challenge Finals*, pages 667–684. 04 2018.

[9] Nicolaus A. Radford, Philip Strawser, Kimberly Hambuchen, Joshua S. Mehling, William K. Verdeyen, A. Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, and et al. Valkyrie: Nasa's first bipedal humanoid robot. *J. Field Robot.*, 32(3):397–419, May 2015.

[10] Claudio Gaz, Fabrizio Flacco, and Alessandro Luca. Identifying the dynamic model used by the kuka lwr: A reverse engineering approach. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1386–1392, 09 2014.

[11] Marco Hutter, Philipp Leemann, Stefan Stevsiic, Andreas Michel, Dominic Jud, Ruedi Figi, Christian Caduff, Markus Loher, Stefan Tagmann, Mark Hoepflinger, and Roland Siegwart. Towards optimal force distribution for walking excavators. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 295 – 301, Piscataway, NJ, 2015. IEEE. 2015 International Conference on Advanced Robotics (ICAR); Conference Location: Istanbul, Turkey; Conference Date: July 27-31, 2015.

[12] A. Hornung, Kai Wurm, and Maren Bennewitz. Humanoid robot localization in complex indoor environments. pages 1690 – 1695, 11 2010.

[13] Giuseppe Oriolo, Antonio Paolillo, Lorenzo Rosa, and Marilena Vendittelli. Humanoid odometric localization integrating kinematic, inertial and visual information. *Autonomous Robots*, 40, 09 2015.

[14] Arnaud Tanguy, Daniele De Simone, Andrew I. Comport, Giuseppe Oriolo, and Abderrahmane Kheddar. Closed-loop MPC with dense visual SLAM - stability through reactive stepping. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 1397–1403, 2019.

[15] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer Handbooks. Springer, 2016.

[16] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`.

[17] Joel Chestnutt, Manfred Lau, G. Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. volume 2005, pages 629 – 634, 05 2005.

[18] Armin Hornung, Andrew Dornbush, Maxim Likhachev, and Maren Bennewitz. Anytime search-based footstep planning with suboptimality bounds. In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Osaka, Japan, November 29 - Dec. 1, 2012*, pages 674–679, 2012.

[19] Andrei Herdt, Holger Diedam, Pierre-Brice Wieber, Dimitar Dimitrov, Katja Mombaur, and Moritz Diehl. Online walking motion generation with automatic foot step placement. *Advanced Robotics*, 24:719–737, 04 2010.

[20] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. 2015:279–286, 02 2015.

[21] Shuuji Kajita and Kazuo Tanie. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1405–1411 vol.2, 1991.

[22] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 2:1620–1626 vol.2, 2003.

[23] Pierre-Brice Wieber. Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In *IEEE-RAS International Conference on Humanoid Robots*, Genova, Italy, 2006.

[24] Nicola Scianca, Marco Cognetti, Daniele De Simone, Leonardo Lanari, and Giuseppe Oriolo. Intrinsically stable MPC for humanoid gait generation. In *16th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2016, Cancun, Mexico, November 15-17, 2016*, pages 601–606, 2016.

[25] Péter Fankhauser, Michael Bloesch, and Marco Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):3019–3026, 2018.

[26] Alessio Zamparelli, Nicola Scianca, L. Lanari, and Giuseppe Oriolo. Humanoid Gait Generation on Uneven Ground using Intrinsically Stable MPC. *IFAC-PapersOnLine*, 51:393–398, 01 2018.

[27] Thomas Röfer, Tim Laue, Arne Hasselbring, Jannik Heyen, Bernd Poppinga, Philip Reichenberg, Enno Roehrig, and Felix Thielke. B-Human team report and code release 2018, 2018. Only available online: `http://www.b-human.de/downloads/publications/2018/CodeRelease2018.pdf`.

[28] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[29] Péter Fankhauser and Marco Hutter. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In Anis Koubaa, editor, *Robot Operating System (ROS) âĂŞ The Complete Reference (Volume 1)*, chapter 5. Springer, 2016.

[30] Péter Fankhauser, Michael Bloesch, Diego Rodriguez, Ralf Kaestner, Marco Hutter, and Roland Y. Siegwart. Kinect v2 for mobile robot navigation: Evaluation and modeling. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 388 – 394, Piscataway, NJ, 2015-09-10. IEEE. International Conference on Advanced Robotics (ICAR 2015); Conference Location: Istanbul, Turkey; Conference Date: July 27-31, 2015.

[31] Péter Fankhauser, Marko Bjelonic, Dario Bellicoso, Takahiro Miki, and Marco Hutter. Robust rough-terrain locomotion with a quadrupedal robot. 05 2018.

[32] ASUS Xtion Pro. `https://www.asus.com/3D-Sensor/Xtion_PRO/`.

[33] `camera_calibration`. `http://wiki.ros.org/camera_calibration`.

[34] M. Vukobratović and J. Stepanenko. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15(1):1 – 37, 1972.

[35] Shuuji Kajita, Hirohisa Hirukawa, Kensuke Harada, and Kazuhito Yokoi. *Introduction to Humanoid Robotics*. Springer Publishing Company, Incorporated, 2014.

[36] Leonardo Lanari, Seth Hutchinson, and Luca Marchionni. Boundedness issues in planning of locomotion trajectories for biped robots. *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 951–958, 2014.

[37] Ahmed Aboudonia, Nicola Scianca, Daniele De Simone, L. Lanari, and Giuseppe Oriolo. Humanoid gait generation for walk-to locomotion using single-stage mpc. pages 178–183, 11 2017.

[38] Nicola Scianca, Daniele De Simone, Leonardo Lanari, and Giuseppe Oriolo. MPC for humanoid gait generation: Stability and feasibility. *CoRR*, abs/1901.08505, 2019.

[39] Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6, 12 2014.

[40] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Rob. Res.*, 30(7):846–894, June 2011.

[41] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart. State estimation for legged robots on unstable and slippery terrain. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6058–6064, Nov 2013.

[42] David Wisth, Marco Camurri, and Maurice Fallon. Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry, 10 2019.

[43] Robert Griffin, Georg Wiedebach, Stephen McCrory, Sylvain Bertrand, Inho Lee, and Jerry Pratt. Footstep planning for autonomous walking over rough terrain, 07 2019.