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 07 January 2020
in front of a Board of Examiners composed by:

Prof. Nome Cognome (chairman)

Prof. Nome Cognome

Dr. Nome Cognome

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

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

Version: January 6, 2020

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

Dedicated to
Donald Knuth

Abstract

This document is an example which shows the main features of the LaTeX 2_ε class `sapthesis.cls` developed by Francesco Biccari with the help of GuIT (Gruppo Utilizzatori Italiani di TeX).

Acknowledgments

Ho deciso di scrivere i ringraziamenti in italiano per dimostrare la mia gratitudine verso i membri del GuIT, il Gruppo Utilizzatori Italiani di TEX, e, in particolare, verso il prof. Enrico Gregorio.

Contents

Chapter 1

Introduction

Overview of humanoid robots.

Overview of the thesis (aim, results).

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 make to 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 allows 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` [1], 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 TODO and it is used by the footstep planner (Chapter 3) to make NAO climb the stairs.

2.1 Framework

Todo (overview of elevation_mapping). ROS.

Figure 2.1. The ASUS Xtion Pro 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.

2.1.1 Definitions

Todo (RFs).

2.1.2 Map Update

Todo (range measurements + robot motion).

2.1.3 Map Fusion and Dynamic Environments

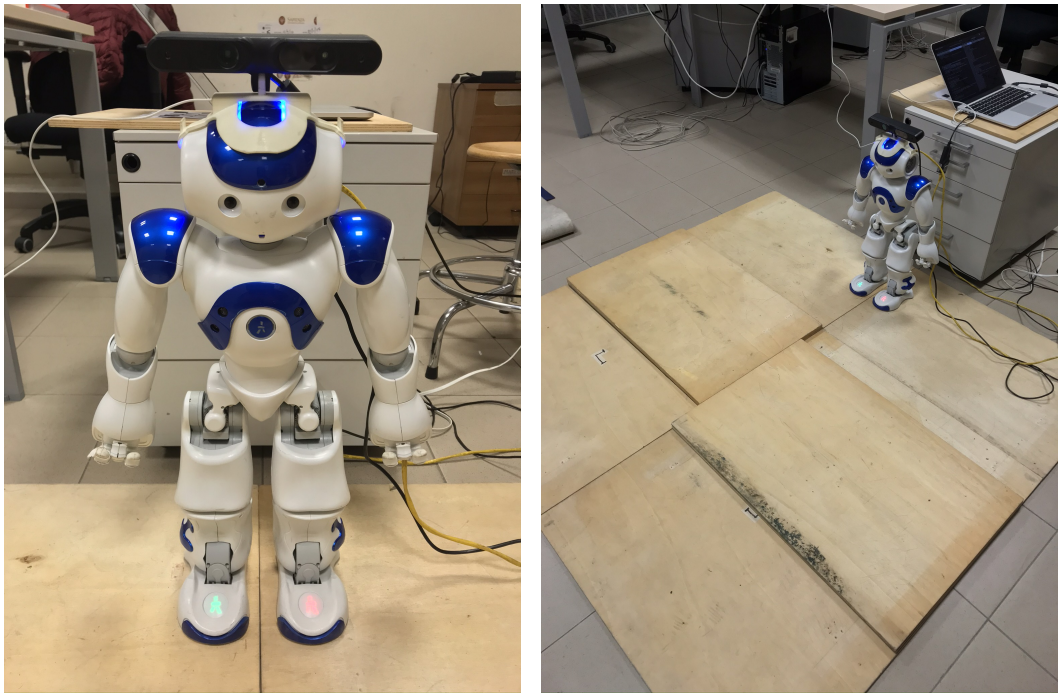
Todo.

2.2 ASUS Xtion Pro

xtion (RGBD camera). ROS.

2.3 World of Stairs

(safe zone)



(a) (b)

Figure 2.2. On the left, NAO humanoid robot with ASUS Xtion Pro placed on top. On the right, NAO humanoid robot in the environment described TODO, right before starting the execution of the experiment.

Figure 2.3. 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.

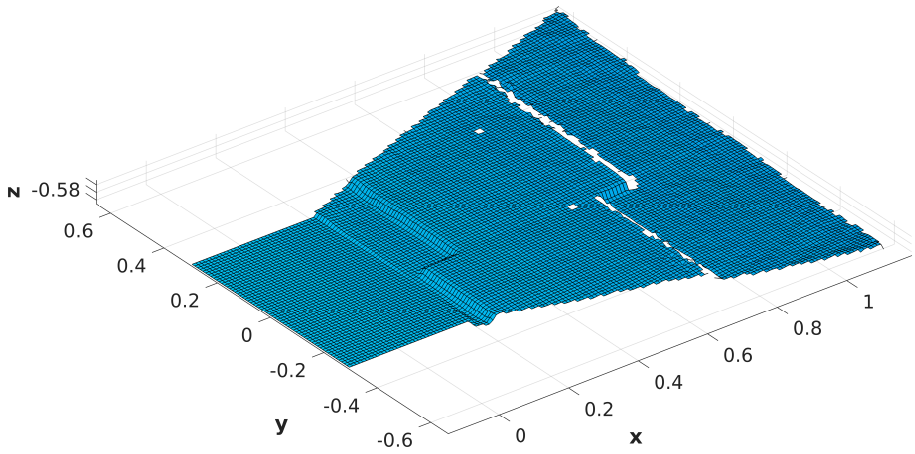


Figure 2.4. Elevation map build by `elevation_mapping` for the environment described TODO.

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.

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's denote the map 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 (offline) 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's denote $\boldsymbol{f}^j = (x^j_{\text{f}}, y^j_{\text{f}}, z^j_{\text{f}}, \theta^j_{\text{f}})^T$ as the pose of the j-th footstep, with $x^j_{\text{f}}, y^j_{\text{f}}, z^j_{\text{f}}$ position of the footstep and θ^j_{f} 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, let's introduce the following requirements:

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

R2 The footstep is fully in contact within 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 $\boldsymbol{p}_{\mathrm{swg}}^j$.

The reference trajectories $\boldsymbol{p}_{\mathrm{swg}}^*, \boldsymbol{p}_{\mathrm{swg}}^j$ are passed to a pseudoinverse-based kinematic controller to compute joint commands $\dot{\boldsymbol{q}}$ for the robot. Proprioceptive feedback is used for both gait generator block and kinematic controller.

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 $\boldsymbol{f}_{\mathrm{sup}}$ and $\boldsymbol{f}_{\mathrm{swg}}$ in v) and $\boldsymbol{p}$, θ_p is the angle between the robot sagittal

axis 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 $\{\boldsymbol{p}_{\text{swg}}^j\}$.

Algorithm 1: Footstep Planner

```
1  root the tree T at v_ini ← (f_L, f_R);
2  i ← 0;
3  repeat
4      i ← i + 1;
5      generate a random point p_rand on the ground;
6      select the closest vertex v_near in T to p_rand according to γ(·, p_rand);
7      randomly select from the primitive catalogue U a candidate footstep
         f^cand;
8      if f^cand is feasible w.r.t. R1–R2 then
9          h ← h_min;
10         p_swg^cand ← BuildTrajectory(f_swg^near, f^cand, h);
11         while h ≤ h_max and Collision(p_swg^cand) do
12             h ← h + Δh;
13             p_swg^cand ← BuildTrajectory(f_swg^near, f^cand, h);
14         if h ≤ h_max then
15             v_new ← (f^cand, f_sup^near);
16             add vertex v_new to T as a child of v_near;
17             compute midpoint m between the feet at v_new;
18 until m ∈ G or i = i_max;
```

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` framwork 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:

$$(x, y, \theta) \in \{-6.0, 0.0, 6.0, 8.0, 10.0\} \times \{13.0, 14.0\} \times \{0, \pi/12\} \tag{3.2}$$

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, $h_{\min} = 0.02$m, $h_{\max} = 0.07$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.

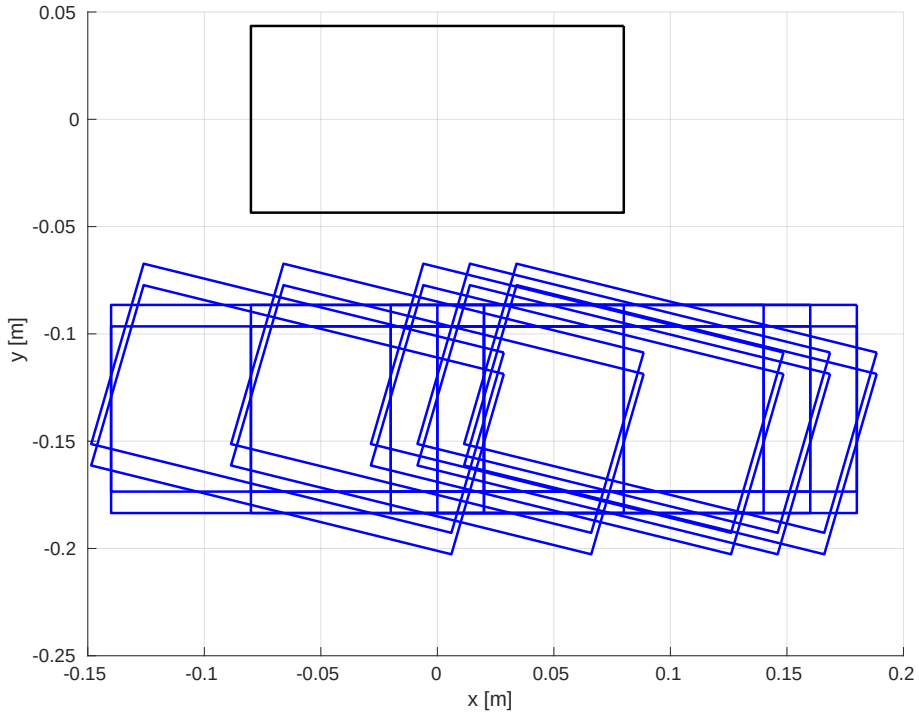


Figure 3.1. The catalogue of primitives (blue color) specifies the possible poses of the candidate footstep $\boldsymbol{f}^{\text{cand}}$ with respect to the pose of the current support foot $\boldsymbol{f}^{\text{near}}_{\text{sup}}$. The figure shows the case where the left foot is the support foot. 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 $\mathcal{M}_z$.

Chapter 4

Variable Height CoM IS-MPC

Before describing in detail the model [3] 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 [4], 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 [5]. The Linear Inverted Pendulum model easily allows to design control schemes for the generation of the CoM reference trajectory, like the Preview Control [6] and the Model Predictive Control [7]. Before discussing the LIP and extending it to the Variable Length Inverted Pendulum, let's discuss the 3D motion model of the CoM reminding 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

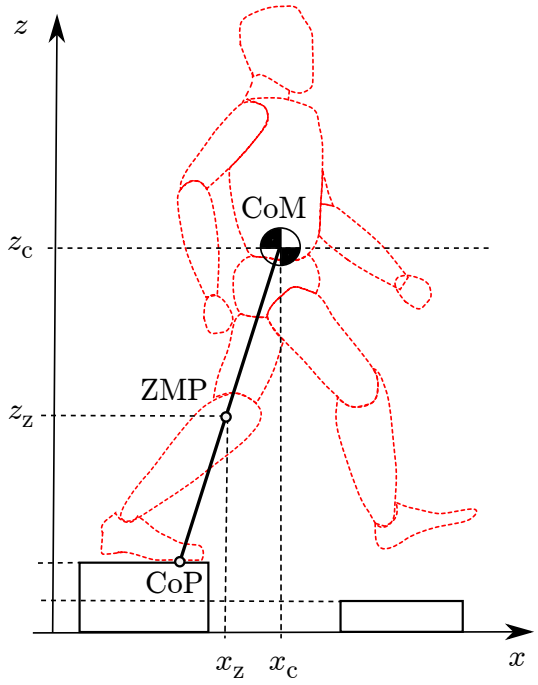


Figure 4.1. When walking on flat ground, or more in general on piecewise-horizontal ground as in the figure, the ZMP can be anywhere between the line connecting the CoP and the CoM.

placed) and the CoM (Fig. 4.1), it is possible [8] 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 has constant elevation with respect to the ground (i.e. $z_c = \bar{z_c}$). As a consequence,

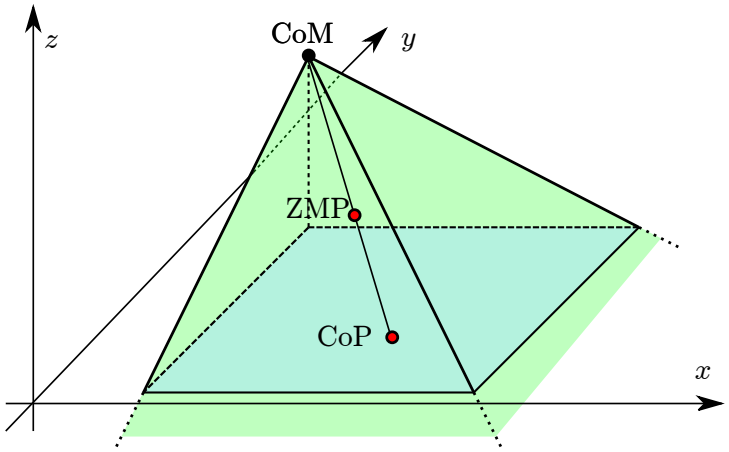


Figure 4.2. The CoP should be internal to the support polygon, which is equivalent to the ZMP being internal to the polyhedral cone with the CoM as a vertex.

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, let's consider 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 [9] 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.

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

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

4.4.1 ZMP constraints

Before describing the ZMP constraints in the 3D case (Fig. 4.2), let's discuss the 2D case.

When walking on fully horizontal ground, the robot keeps the equilibrium if the ZMP remains inside the support polygon. Let's denote by $(x_f^j, y_f^j, \theta_f^j)$ the pose of the generic footstep within the given sequence. Let's use a fixed-shape moving ZMP constraint [10] to enforce balance. The admissible region for ZMP at t_{k+i} is centered

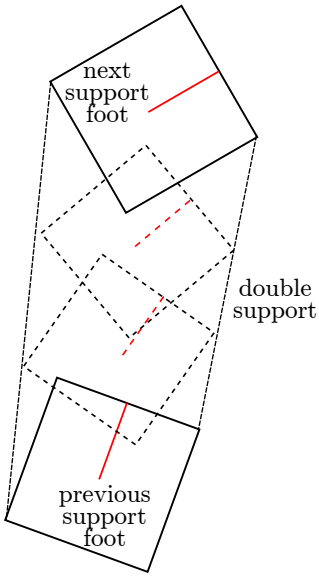


Figure 4.3. The ZMP constraint moves from a support foot to the following one during double support phase.

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:

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

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 polyhedral cone defined by the support polygon and CoM (Fig. 4.2. When ZMP is allowed to move vertically, the constraint becomes nonlinear. In order to remove nonlinearity it is possible to consider a subregion of

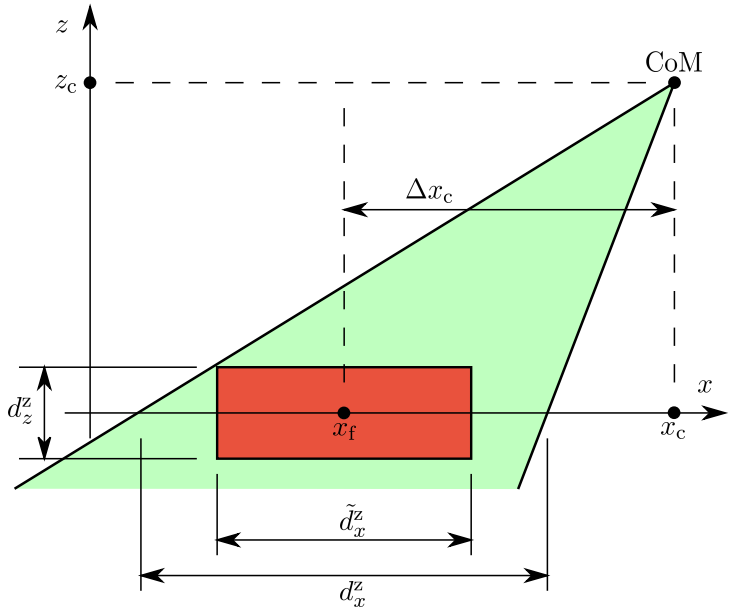


Figure 4.4. The polyhedral cone representing the ZMP constraint and the box used to approximate the constraint (which becomes linear).

the polyhedral cone, for example a box constraint, 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 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

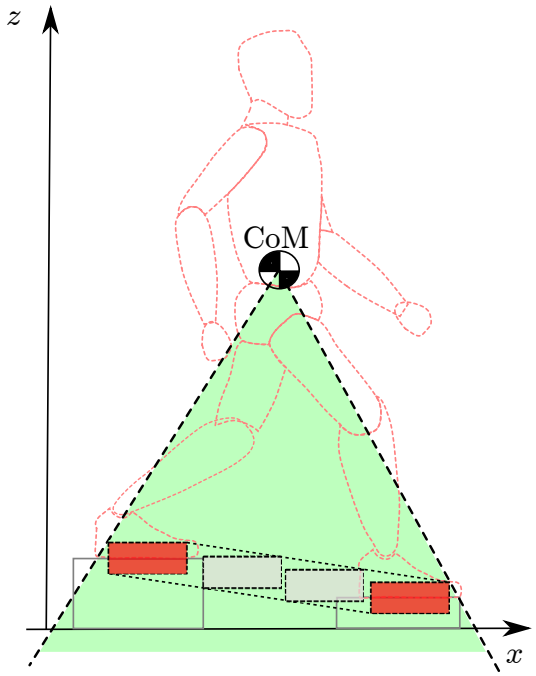


Figure 4.5. The ZMP box constraint moves from a support foot to the following one during double support phase.

make the generated gait unrealizable. However, as mentioned before and as discussed in [9, 11], 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 $\dot{x}_{\mathrm{z}}^{k+i}, \dot{y}_{\mathrm{z}}^{k+i}, \dot{z}_{\mathrm{z}}^{k+i}$. 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):

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

A more detailed discussion can be found in [12].

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 [13] and it has been tested on both dynamic environments and NAO humanoid robot. The QP problem has been solved using qpOASES [14]. 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: $\omega = 6.68\text{s}^{-1}$, the step duration $T_s = 0.48\text{s}$ of which $t_{\text{SS}} = 0.30\text{s}$ of single support and $t_{\text{DS}} = 0.18\text{s}$ of double support. The size of the box constraint have been set to $\tilde{d}_x^z = 0.05\text{m}, \tilde{d}_y^z = 0.05\text{m}, \tilde{d}_z^z = 0.05\text{m}$.

Chapter 5

Experiments

5.1 NAO

Brief description of NAO v5 (which is running BHuman).

5.2 Simple Staircase

Staircase from 1cm to 5cm. Limitations of the robot.

5.3 Multiple Staircases

Two consecutive staircases of 2cm.

5.3.1 Upstairs

Upstairs.

5.3.2 Downstairs

Downstairs.

5.4 Obstacle Avoidance

Footstep planning with manually generated elevation mapping. Map contains an obstacle that can not be climbed by NAO because of short legs.

5.5 Stair Climbing in Unknown Environment

Stair climbing (unknown environment). Planner + `elevation_mapping`.

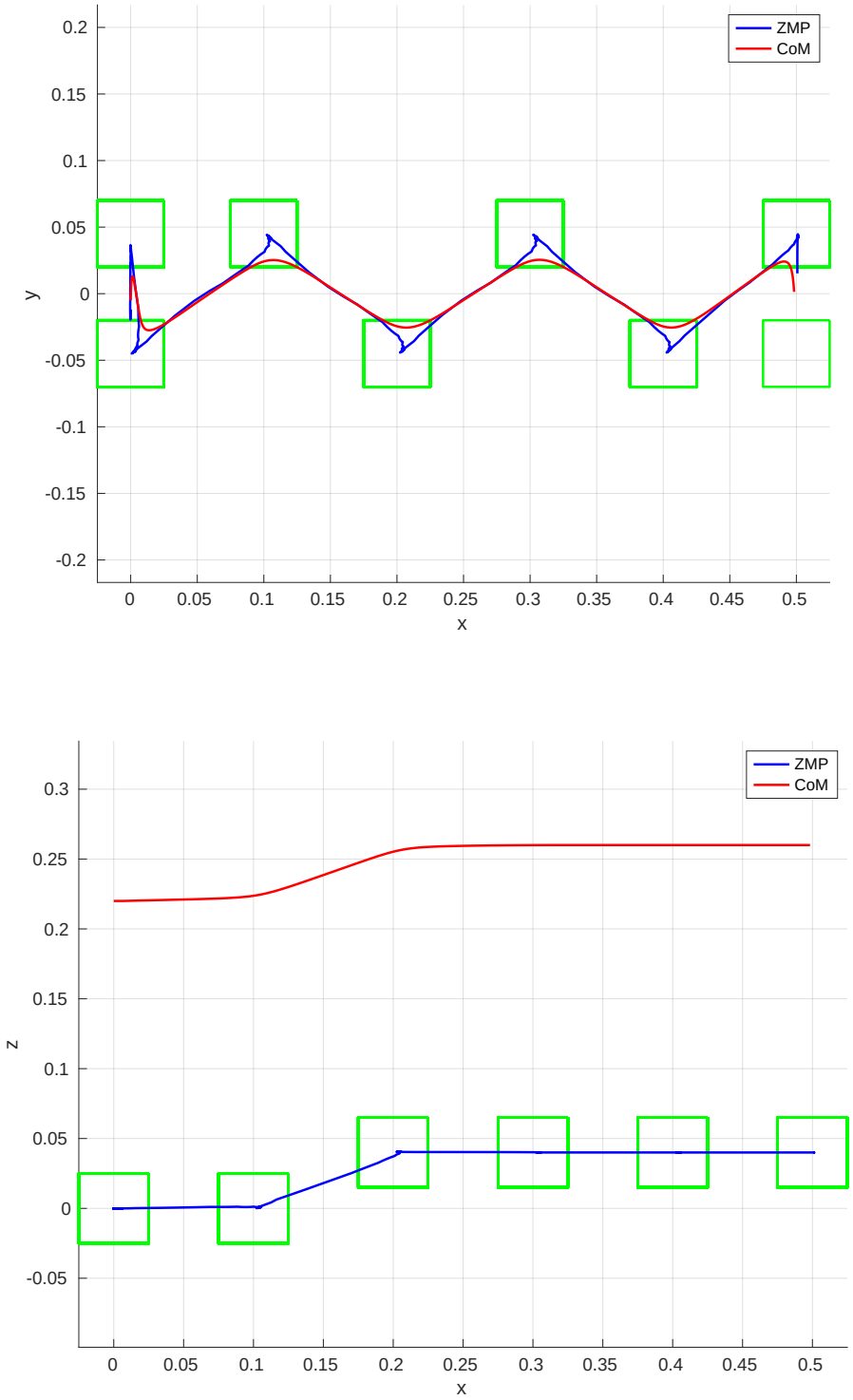
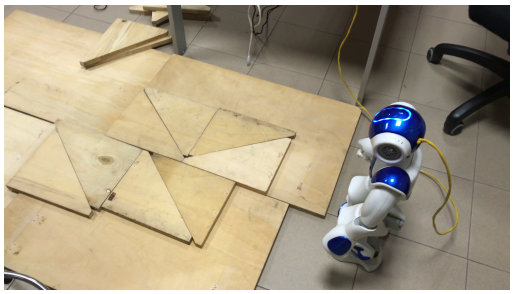
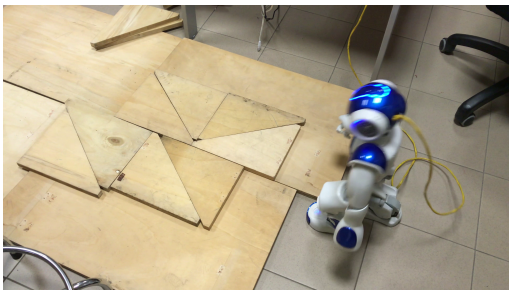


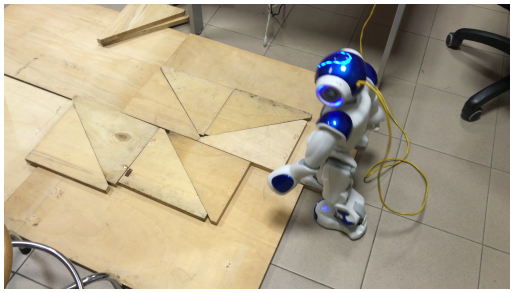
Figure 5.1. The plots show how the CoM and the ZMP vary with respect to the footsteps in the scenario “Simple Staircase”. The green boxes represent the footsteps.



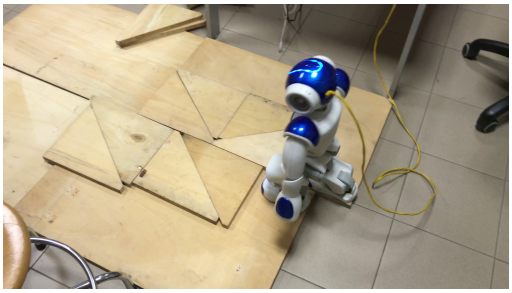
(a) Starting position



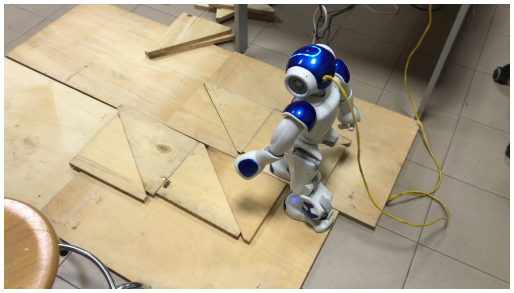
(b) First step



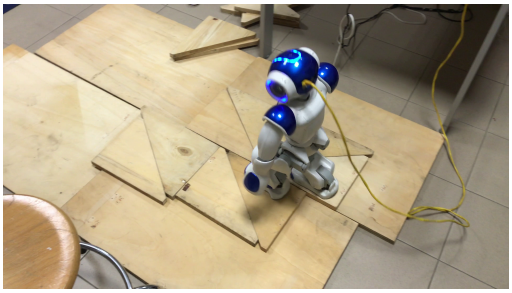
(c) Second step



(d) Third step



(e) Fourth step



(f) Fifth step

Figure 5.2. The figures show the motion of the robot for the scenario "Multiple Staircases (Upstairs)". The robot starts just in front of the stairs (Fig. 5.2a), 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.

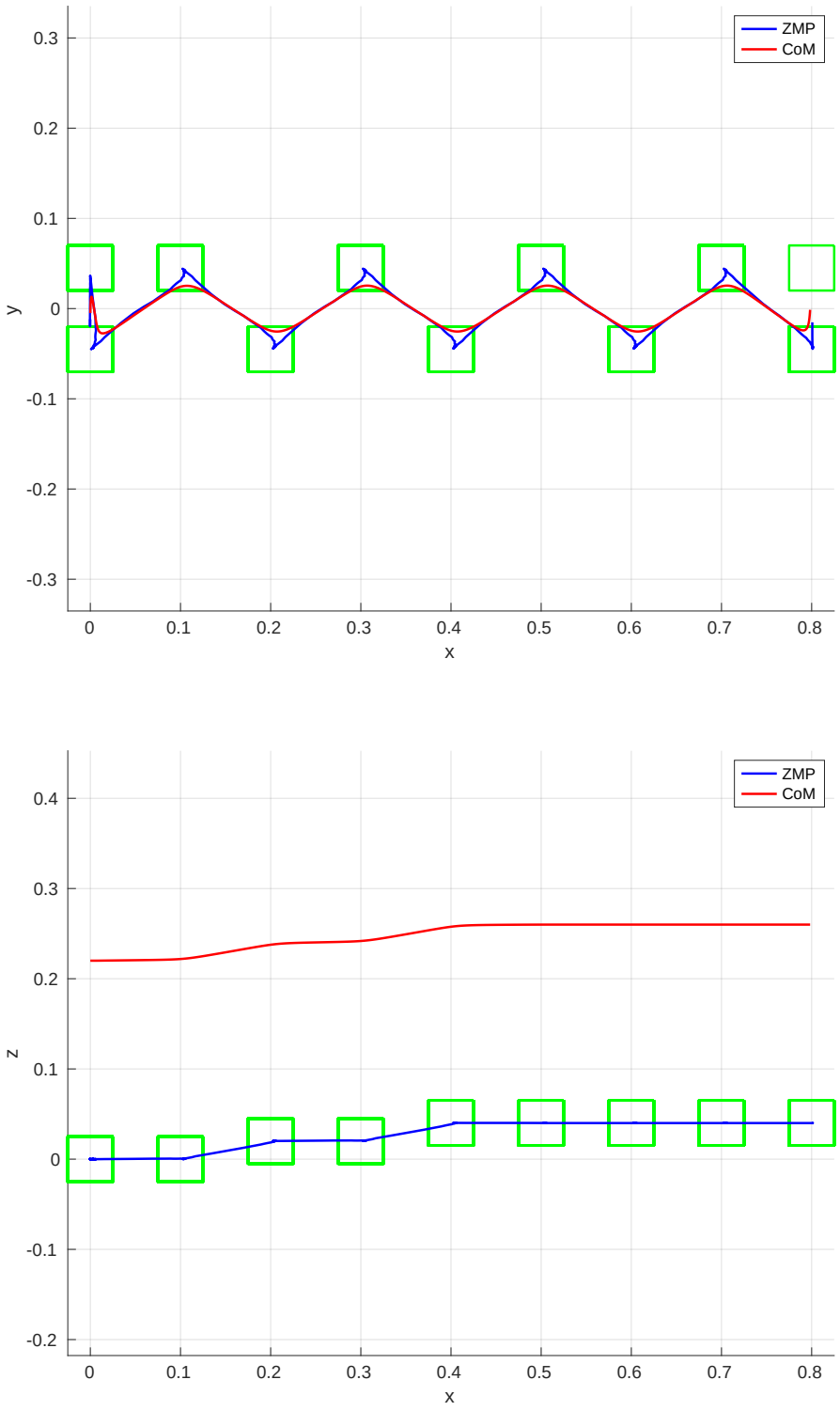


Figure 5.3. The plots show how the CoM and the ZMP vary with respect to the footsteps in the scenario "Multiple Staircases (Upstairs)". The green boxes represent the footsteps.

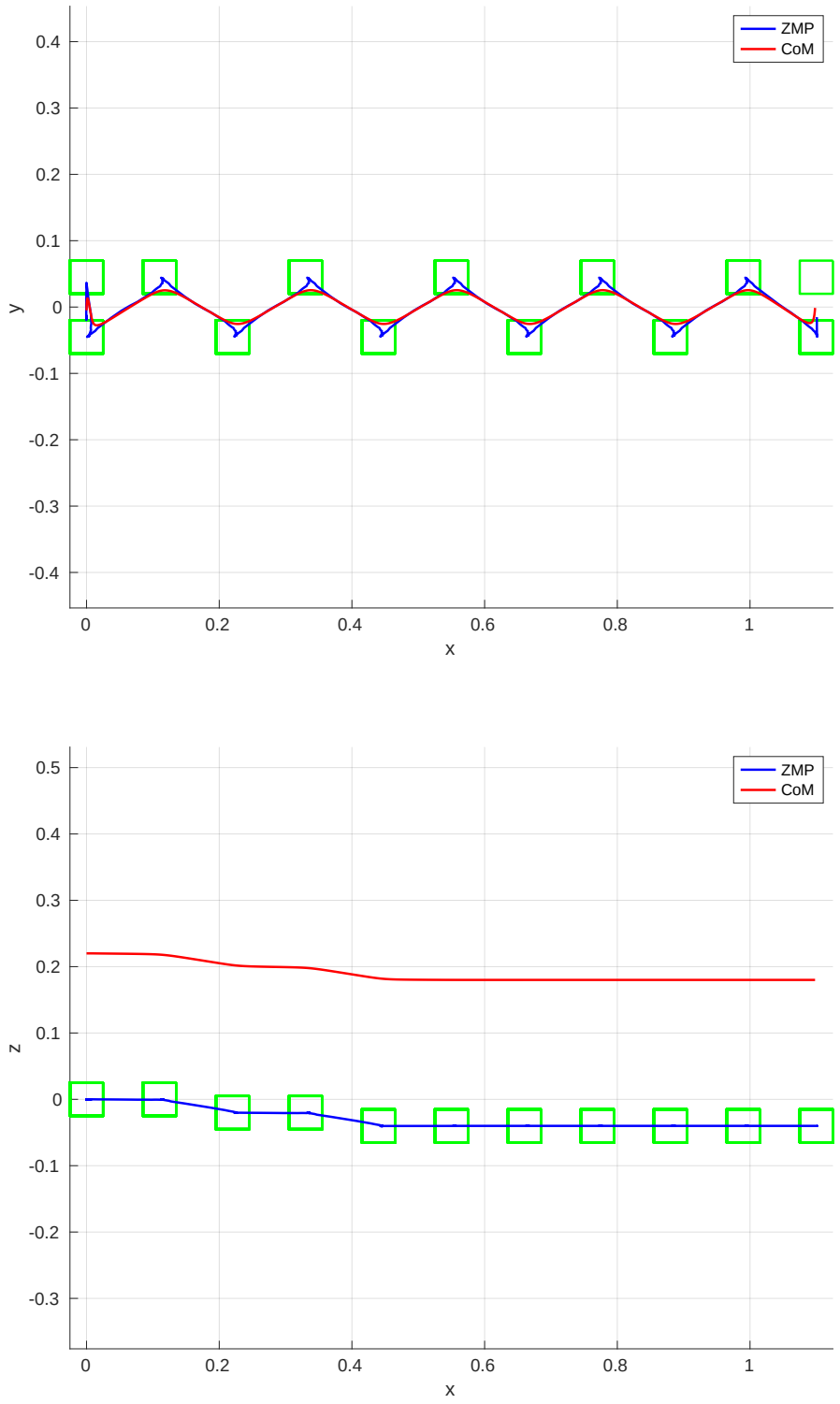


Figure 5.4. The plots show how the CoM and the ZMP vary with respect to the footsteps in the scenario "Multiple Staircases (Downstairs)". The green boxes represent the footsteps.

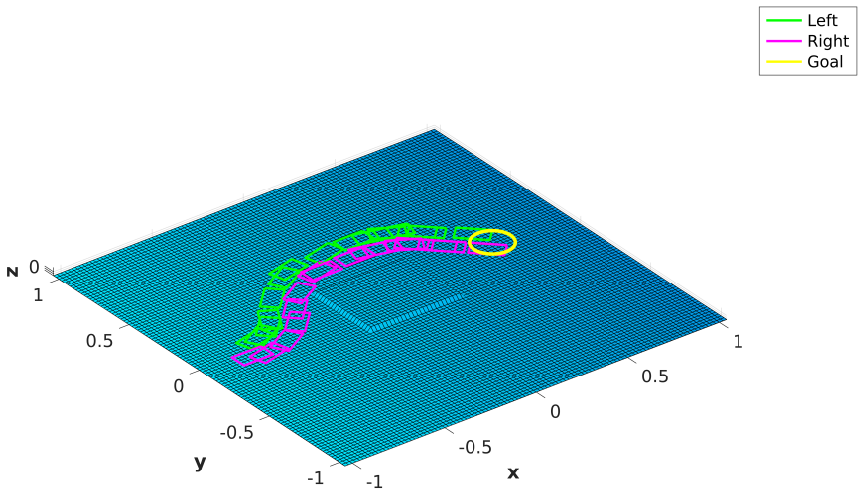


Figure 5.5. Footstep plan generated for the scenario “Obstacle Avoidance”.

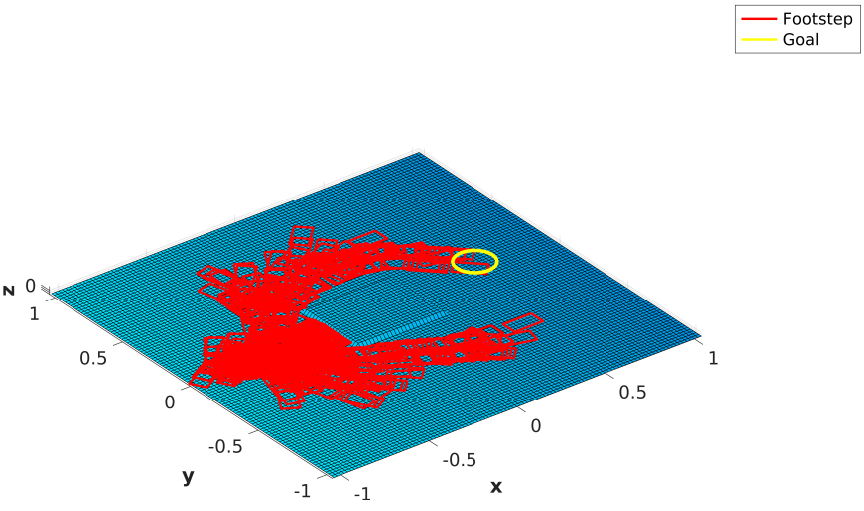


Figure 5.6. Tree generated for the scenario “Obstacle Avoidance”.

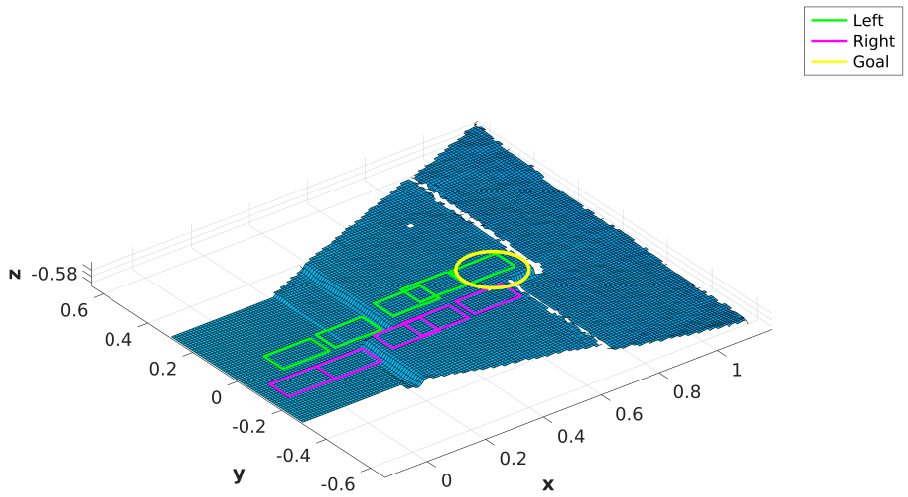


Figure 5.7. Footstep plan generated for the scenario "Stair Climbing in Unknown Environments".

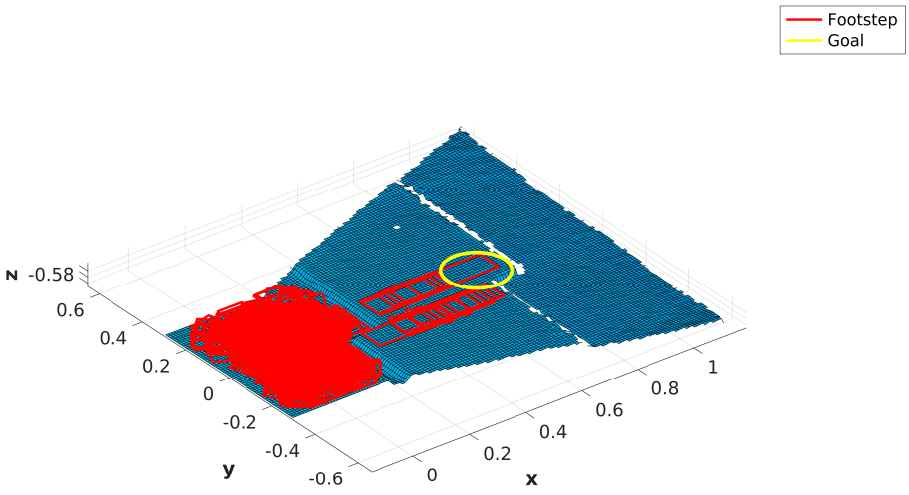


Figure 5.8. Tree generated for the scenario "Stair Climbing in Unknown Environments".

Chapter 6

Conclusion

What's next: tracker -> makes it possible to continuously build the map with elevation_mapping -> replanning -> makes it possible to move in whatever environment (World of Stairs).

Rough environments?

Bibliography

[1] 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.

[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] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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`.

[14] 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.