robotics

MDPI

Article

Humanoid Motion Generation in Complex 3D Environments

Diego Marussi, Michele Cipriano, Nicola Scianca *, Leonardo Lanari and Giuseppe Oriolo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto 25, 00185 Rome, Italy; marussi.1962008@studenti.uniroma1.it (D.M.); cipriano@diag.uniroma1.it (M.C.); lanari@diag.uniroma1.it (L.L.); oriolo@diag.uniroma1.it (G.O.)

* Correspondence: scianca@diag.uniroma1.it

Abstract: We address the problem of humanoid locomotion in 3D environments consisting of planar regions with arbitrary inclination and elevation, such as staircases, ramps, and multi-floor layouts. The proposed framework combines an offline randomized footstep planner with an online control pipeline that includes a model predictive controller for gait generation and a whole-body controller for computing robot torque commands. The planner efficiently explores the environment and returns the highest-quality plan it can find within a user-specified time budget, while the control layer ensures dynamic balance and adequate ground friction. The complete framework was evaluated via dynamic simulation in MuJoCo, placing the JVRC1 humanoid in four scenarios of varying complexity.

Keywords: humanoid; planning; model predictive control

1. Introduction and Related Work

Humanoid robots have seen remarkable advancements in recent years due to improvements in both design and control. They are increasingly being considered for practical applications in settings where human-like mobility is essential. Unlike wheeled robots, which are mostly limited to flat and structured environments, humanoids are designed to traverse complex, human-centric environments, including stairs and ramps.

Despite recent successes demonstrating formidable agility, we have seen less impressive results in the field of the autonomous navigation of complex environments. Indeed, in such environments, it is necessary to combine long-term planning with real-time control.

Moving in complex and cluttered environments demands the ability to plan over a long horizon, ensuring that the robot can reach a distant goal without getting stuck. To make the problem more approachable, the possible paths the robot can take are often reduced to those that can be constructed as concatenations of *primitives*, i.e., elementary motions represented by footstep displacements [1]. Approaches based on grid or graph search (e.g., using A*) proved effective in relatively simple contexts [2–4]. However, it is well known that the performance of discrete planners may deteriorate in complex environments due to insufficient granularity (depending on the resolution of the grid and the size of the primitive catalog) and combinatorial explosion (due to the need to explore a large number of branches in a high-dimensional space). Moreover, designing appropriate heuristics may prove difficult in practice.

The representation of the environment is a crucial aspect, not only for efficiency reasons but also because it can allow or disallow features of the algorithm. A commonly used representation is an *elevation map*, which associates each point in the xy-plane with a corresponding elevation value. Although elevation maps are easy to reconstruct from data, they have limitations, e.g., they do not allow for the representation of multi-floor

check for updates

Academic Editor: Kensuke Harada

Received: 21 April 2025
Revised: 9 June 2025
Accepted: 12 June 2025
Published: 16 June 2025

Citation: Marussi, D.; Cipriano, M.; Scianca, N.; Lanari, L.; Oriolo, G. Humanoid Motion Generation in Complex 3D Environments. *Robotics* **2025**, *14*, 82. https://doi.org/10.3390/robotics14060082

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

environments. In [5,6], the authors showed efficient ways to represent the environment as a set of planar regions and tested online footstep planning using an A* algorithm.

A significant boost in search efficiency can be obtained using sample-based methods, such as those based on Rapidly-exploring Random Trees (RRT) [7]. These methods can address some of the limitations of search techniques because the expansion mechanism is biased towards exploration. RRT in its basic formulation, however, does not account in any way for the quality of the produced plan and can lead to inefficient motions. Furthermore, the use of a primitive catalog results in a certain *granularity* of the final plan, as the number of possible choices is limited by the number of primitives available.

On the control side, many techniques have been proposed. While approaches based on reinforcement learning and imitation learning have proven to produce very agile motions, these are usually not designed to move in crowded environments because high-level commands are given as reference velocities, rather than a precise sequence of footstep positions. Conversely, approaches based on model predictive control can more effectively follow an input reference footstep plan. Classic methods use a Linear Inverted Pendulum (LIP) as a simplified dynamic model [8], which is unsuited to motion in complex environments due to the constant height Center of Mass (CoM) requirement. However, later studies have shown that different simplified models can still be used to generate variable-height CoM trajectories, e.g., see [9].

The problem becomes more complex when the surfaces on which it is possible to step have an arbitrary orientation since the possibility of slipping of the contact feet is significant. Therefore, it is necessary to generate interaction forces that offer sufficient friction. One possibility is to produce a sufficiently robust controller that can walk blindly and adapt to unknown slope changes [10,11]. While this has its advantages, there are cases in which the terrain information is known, and its use undoubtedly produces better performance. Ref. [12] focused on the problem of moving across surfaces with known, different orientations, but the footstep plan was given a priori. In [13], we proposed a method for humanoid gait generation based on an predictive control formulation that included an explicit stability constraint. While this method used the LIP as a template model, we showed that it can be extended in order to generate vertical motions, including running [14]. In [15], we integrated our controller with a footstep planner to realize motion in complex environments constituted by horizontal ground patches at different heights (a *world of stairs*). The exploration was performed by randomly expanding a tree of possible footstep sequences with the goal of reaching a given goal region. The planner was given a time budget, after which it would return the best found plan according to a specified optimality criterion using an RRT*-like strategy [16].

In this paper, we propose a comprehensive framework for footstep planning and control that addresses the unique demands posed by complex environments, including the presence of stairs, inclined planes, and multiple floors. The footstep planner is based on an offline randomized exploration strategy that is continuous in nature (i.e., it uses neither gridmaps nor motion primitives) and finds the best footstep plan in a given time budget. Its goal is to find the sequence that reaches a given goal position in the minimum number of steps. The control pipeline is constituted by a model predictive controller (MPC) and a whole-body controller (WBC) generating torque control commands for the humanoid.

In particular, our unified framework is capable of

- planning a collision-free sequence of footsteps in a *world of ramps*, i.e., an environment constituted by planar regions with different inclinations, without using motion primitives;
- optimizing said footstep plan according to a quality metric and returning the best solution found within the given time budget;

- providing, along with the footstep sequence, variable-height collision-free trajectories for the swing foot in order to overcome small obstacles on the ground or climb at a different height;
- generating a stable 3D CoM trajectory that allows the robot to move along the planned footstep sequence;
- computing torque commands for the humanoid robot in such a way that contact forces provide sufficient friction to avoid slipping.

We validated the footstep planner with an extensive testing campaign in which we evaluated its performance in several different simulated environments and compared results using different time budgets. Furthermore, we tested the full framework, including the control modules, in dynamic simulations in MuJoCo 3.2.7.

The rest of this article is organized as follows. In Section 2, we formulate the motion generation problem, while, in Section 3, we give an overview of the proposed approach. A full description of the introduced footstep planner is given in Section 4, while the control architecture is presented in Section 5. We extensively test the proposed approach on increasingly complex environments through dynamic simulations in MuJoCo in Section 6. Section 7 concludes the article.

2. Problem Formulation

In the situation of interest (Figure 1), a humanoid robot moves in a *world of ramps*, a specific 3D environment consisting of planar regions that are arbitrarily placed and oriented in space. This characterization encompasses office-like scenarios, with rooms connected by staircases or ramps, possibly arranged on several floors and populated by obstacles. Depending on its elevation and/or inclination, a planar region may be accessible for the humanoid to step on or not.

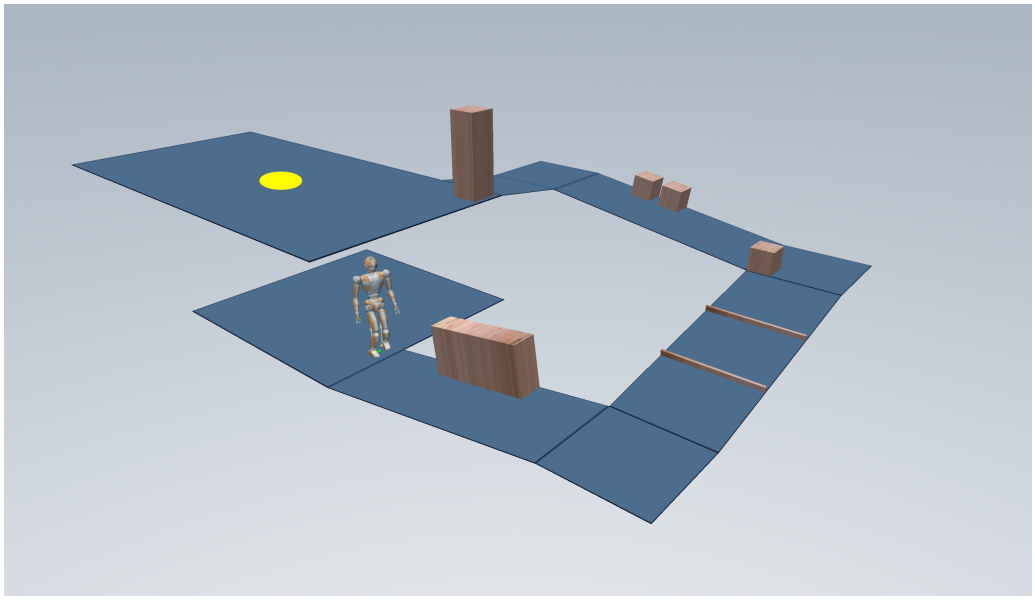

Figure 1. An instance of the considered problem. The robot must reach the goal area (in yellow) by traversing a *world of ramps*.

The mission of the robot is to reach a certain goal area $\mathcal{G}$, assumed to be contained in a single planar region. The locomotion task is accomplished as soon as the robot places a foot inside $\mathcal{G}$.

We wanted to devise a complete framework enabling the humanoid to plan and execute a motion to fulfill the assigned task in the world of ramps. This required addressing two fundamental problems: finding a 3D footstep plan and generating a variable-height gait

consistent with such a plan. Footstep planning consists of finding both footstep placements and swing foot trajectories; overall, the footstep plan needed to be feasible (in a sense to be formally defined later) for the humanoid, given the characteristics of the environment. Gait generation consists of finding a CoM trajectory that realizes the footstep plan while guaranteeing the dynamic balance of the robot at all time instants. Starting from the trajectories of the CoM and the feet, a whole-body controller generates torque commands for the robot joints that comply with the robot kinematic and dynamic limitations.

The problem was solved under the following assumptions.

A1 Information about the environment geometry was completely known a priori and collected in a *map* $\mathcal{M}$ expressed as

$$\mathcal{M} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}, \tag{1}$$

where $\mathcal{R}_i$ is an arbitrarily oriented planar region in the 3D space (a *ramp*).

A2 The humanoid was endowed with a localization module which, using the available sensor data, provided an estimate of the robot's current state in terms of its CoM and swing foot pose.

In view of assumption A1, a complete footstep plan leading to the goal area $\mathcal{G}$ could be computed offline before the humanoid started to execute the motion. While we maintained this viewpoint in the present study, an extension to the online case could be devised along the same lines of [15], i.e., by updating the footstep plan on the basis of new information gathered and added to $\mathcal{M}$ during the motion, e.g., using visual information [5,6,15].

3. Proposed Approach

To solve the described problem, we adopted the architecture shown in Figure 2, in which the main components were the footstep planning, gait generation, and whole-body control modules.

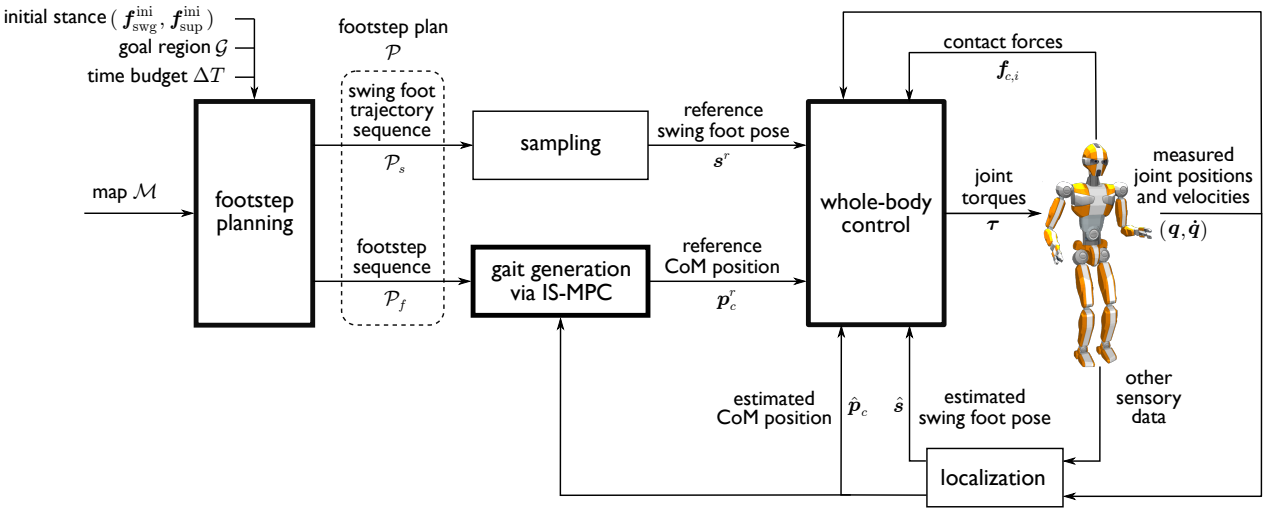

Figure 2. A scheme of the proposed solution approach. Blocks with a thick contour are the subject of specific sections in this paper.

In the following, we denote by $\boldsymbol{f} = (x_f, y_f, z_f, \alpha_f, \beta_f, \gamma_f)$ the *pose* of a footstep, with x_f, y_f, and z_f representing the coordinates of a representative point, henceforth collectively denoted as $\boldsymbol{p}_f$. $\alpha_f, \beta_f, \gamma_f$ is its orientation, expressed via roll–pitch–yaw angles, collectively denoted as $\boldsymbol{\phi}_f$. Moreover, a pair $(\boldsymbol{f}_{\text{swg}}, \boldsymbol{f}_{\text{sup}})$ defines a *stance*, i.e., the feet poses during a double support phase, after which a step is performed by moving the swing foot from $\boldsymbol{f}_{\text{swg}}$ while keeping the support foot at $\boldsymbol{f}_{\text{sup}}$.

The footstep planner receives as input the initial humanoid stance $(\boldsymbol{f}_{\text{swg}}^{\text{ini}}, \boldsymbol{f}_{\text{sup}}^{\text{ini}})$, the goal area $\mathcal{G}$, a time budget ΔT, and the map $\mathcal{M}$ of the environment.

The time budget is the time afforded to the planner for finding a solution. After each iteration, if the elapsed time exceeds this budget, the algorithm is terminated. At this point, the planner either returns a solution or ends with a failure. Including explicitly this parameter as one of the inputs to the algorithm allowed us to compare different budgets to evaluate the performance of the planning module, but it also paved the way for the extension to the online case as, here, the time budget could be set to be equal to the duration of a step to meet the real-time requirement.

The planner works off-line to find, within ΔT, an optimal *footstep plan* $\mathcal{P} = \{\mathcal{P}_f, \mathcal{P}_s\}$ leading to the desired goal area $\mathcal{G}$. In $\mathcal{P}$, we denote by

$$\mathcal{P}_f = \{\boldsymbol{f}^1, \ldots, \boldsymbol{f}^n\} \tag{2}$$

the sequence of footstep placements, whose generic element $\boldsymbol{f}^j$ is the pose of the j-th footstep, with $\boldsymbol{f}^1 = \boldsymbol{f}_{\text{swg}}^{\text{ini}}$ and $\boldsymbol{f}^2 = \boldsymbol{f}_{\text{sup}}^{\text{ini}}$. Also, we denote by

$$\mathcal{P}_s = \{\boldsymbol{s}^1, \ldots, \boldsymbol{s}^{n-2}\} \tag{3}$$

the sequence of associated swing foot trajectories, whose generic element $\boldsymbol{s}^j$ is the j-th *step*, i.e., the trajectory leading the foot from $\boldsymbol{f}^j$ to $\boldsymbol{f}^{j+2}$ over a fixed duration T_s.

Once the footstep plan has been generated, the sequence of footsteps $\mathcal{P}_f$ is sent to a gait generation module based on a 3D extension of our Intrinsically Stable MPC (IS-MPC). This module computes in real time a variable-height CoM trajectory that is compatible with $\mathcal{P}_f$ and guaranteed to be *stable*, i.e., bounded with respect to a specific point called the *Zero Moment Point* (ZMP, more on this in Section 5.1). In particular, we denote by $\boldsymbol{p}_c^r$ the current reference position of the CoM produced by IS-MPC. Also, let $\boldsymbol{s}^r$ be the current reference pose of the swing foot, obtained by sampling the appropriate subtrajectory in $\mathcal{P}_s$.

Finally, the reference trajectories $\boldsymbol{p}_c^r$ and $\boldsymbol{s}^r$ are passed to a WBC, which computes the joint torque commands τ for the robot to track these trajectories while ensuring that the generated contact forces are feasible.

Sensor information, including joint encoder data, is used by the localization module to continuously update the estimate of the CoM position and swing foot pose ($\hat{\boldsymbol{p}}_c$ and $\hat{\mathbf{s}}$, respectively) through kinematic computations. Finally, these estimates are used to provide feedback to both the gait generation and whole-body control modules.

4. Footstep Planner

The footstep planner takes as input the robot initial stance $(\boldsymbol{f}_{\text{swg}}^{\text{ini}}, \boldsymbol{f}_{\text{sup}}^{\text{ini}})$, a goal region $\mathcal{G}$, the time budget ΔT, and the elevation map $\mathcal{M}_z$. Based on the given optimality criterion, this module returns the best footstep plan $\mathcal{P}$ to the goal that is found within ΔT.

The planning algorithm builds a tree $\mathcal{T}$, where each vertex $v = (\boldsymbol{f}_{\text{swg}}, \boldsymbol{f}_{\text{sup}})$ identifies a stance, and an edge between two vertexes v and $v' = (\boldsymbol{f}'_{\text{swg}} = \boldsymbol{f}_{\text{sup}}, \boldsymbol{f}'_{\text{sup}})$ represents a robot step between the two stances, i.e., a collision-free trajectory such that one foot swings from $\boldsymbol{f}_{\text{swg}}$ to $\boldsymbol{f}'_{\text{sup}}$ and the other remains fixed at $\boldsymbol{f}_{\text{sup}}$.

A footstep plan $\mathcal{P}$ to a generic vertex v consists of a branch starting at the root of the tree and connecting it to v. The sequences $\mathcal{P}_f$ and $\mathcal{P}_s$ for this plan are obtained by collecting respectively the support foot poses of all vertexes and the steps corresponding to all edges along $\mathcal{P}$.

4.1. Footstep Feasibility

Footstep $\boldsymbol{f}^j = (\boldsymbol{p}_f^j, \boldsymbol{\phi}_f^j) = (x_f^j, y_f^j, z_f^j, \alpha_f^j, \beta_f^j, \gamma_f^j) \in \mathcal{P}_f$ is *feasible* if it satisfies the following requirements:

R1 $\boldsymbol{f}^j$ is fully contained in a single horizontal region.

In practice, one typically uses an enlarged footprint to ensure that this requirement will still be satisfied in the presence of small positioning errors.

R2 $\boldsymbol{f}^j$ is kinematically admissible from the previous footstep $\boldsymbol{f}^{j-1}$.

Given $\boldsymbol{f}^{j-1}$, the *kinematically admissible region* for $\boldsymbol{f}^j$ is the submanifold of $\boldsymbol{R}^3 \times SO(3)$, defined as

$$\mathcal{K}(\boldsymbol{f}_{j-1}) = \mathcal{K}_{\text{pos}}(\boldsymbol{p}_{j-1}, \boldsymbol{\phi}_{j-1}) \times \mathcal{K}_{\text{ang}}(\boldsymbol{\phi}_{j-1}). \tag{4}$$

Here, $\mathcal{K}_{\text{pos}}(\boldsymbol{p}_{j-1}, \boldsymbol{\phi}_{j-1})$ is the set of kinematically admissible positions of the j-th footstep, defined by the following constraints:

$$-\begin{pmatrix} \Delta_x^- \\ \Delta_y^- \\ \Delta_z^- \end{pmatrix} \leq \boldsymbol{R}^T(\boldsymbol{\phi}_f^{j-1}) \begin{pmatrix} x_f^j - x_f^{j-1} \\ y_f^j - y_f^{j-1} \\ z_f^j - z_f^{j-1} \end{pmatrix} \pm \begin{pmatrix} 0 \\ \ell \\ 0 \end{pmatrix} \leq \begin{pmatrix} \Delta_x^+ \\ \Delta_y^+ \\ \Delta_z^+ \end{pmatrix}, \tag{5}$$

where $\boldsymbol{R}(\boldsymbol{\phi}_f^{j-1})$ is the rotation matrix associated with $\boldsymbol{\phi}_f^{j-1}$ and the Δ symbols denote lower and upper maximum increments (see Figure 3).
The set $\mathcal{K}_{\text{ang}}(\boldsymbol{\phi}_{j-1})$ of kinematically admissible orientations of the j-th footstep is defined by the following constraints:

$$\begin{aligned} -\Delta_\alpha^- \leq \alpha_f^j \leq \Delta_\alpha^+ \\ -\Delta_\beta^- \leq \beta_f^j \leq \Delta_\beta^+ \\ -\Delta_\gamma^- \leq \gamma_f^j - \gamma_f^{j-1} \leq \Delta_\gamma^+. \end{aligned} \tag{6}$$

Note that the constraints on the roll and pitch angles α_f and β_f are bounds on their value, whereas the constraint on the yaw angle γ_f only limits its variation with respect to the previous footstep. This is because yaw can assume any value, whereas pitch and roll are subject to kinematic limitations that are impossible to evaluate at the planning stage and must then be accounted for conservatively.

R3 $\boldsymbol{f}^j$ is reachable from $\boldsymbol{f}^{j-2}$ through a collision-free motion.

This requirement can only be tested conservatively because information about the whole-body motion of the robot is not yet defined during footstep planning (it will be generated later, in the gait generation phase). In particular, we say that R3 is satisfied if (*i*) a collision-free swing foot trajectory $\boldsymbol{s}^{j-2}$ from $\boldsymbol{f}^{j-2}$ to $\boldsymbol{f}^j$ exists and (*ii*) a suitable volume $\mathcal{B}$ accounting for the maximum occupancy of the upper body of the humanoid at stance $(\boldsymbol{f}^{j-1}, \boldsymbol{f}^j)$ is collision-free. More precisely, $\mathcal{B}$ is a vertical cylinder whose base has radius r_b, centered at the midpoint $\boldsymbol{m}$ between the footsteps, and is raised from the ground by z_b, which represents the average distance between the ground and the hip (Figure 4).

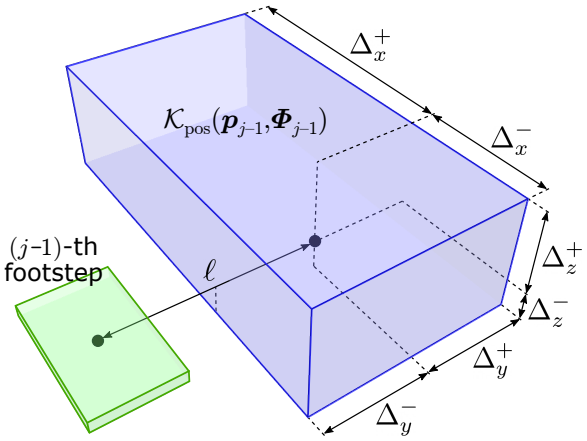

Figure 3. The kinematically admissible region $\mathcal{K}_{\text{pos}}(\boldsymbol{p}_{j-1}, \boldsymbol{\phi}_{j-1})$ for the position of the j-th footstep is defined relative to the position and orientation of the $(j-1)$-th footstep.

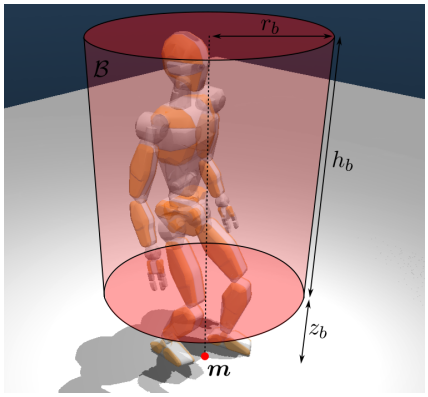

Figure 4. The robot volume occupancy (in red) used for checking requirement R3.

4.2. Algorithm

At the start, the tree $\mathcal{T}$ is rooted at $(\boldsymbol{f}_{\text{swg}}^{\text{ini}}, \boldsymbol{f}_{\text{sup}}^{\text{ini}})$, i.e., the initial stance of the humanoid. During the exploration, $\mathcal{T}$ is expanded by means of an RRT*-like strategy, whose generic iteration entails the following steps: selecting a vertex for expansion, generating a candidate vertex, choosing a parent for the new vertex and rewiring the tree. We will describe these individual steps below, right after some quick preliminaries.

4.2.1. Preliminaries

We assume that an edge between two vertexes v^a and v^b of $\mathcal{T}$ has a cost $l(v^a, v^b)$. The *cost* of a vertex v is then defined as

$$c(v) = c(v^{\text{parent}}) + l(v^{\text{parent}}, v) \tag{7}$$

and represents the cost of reaching v from the root of $\mathcal{T}$. Then, the cost of a plan $\mathcal{P}$ ending at a vertex v is $c(\mathcal{P}) = c(v)$. In this paper, we let $l(v^a, v^b) = 1$ for all edges in $\mathcal{T}$. Correspondingly, the cost of a vertex is the length of the corresponding plan in terms of the number of edges (i.e., steps). See [15] for other interesting definitions of the cost of a plan, such as the total height variation or the minimum obstacle clearance.

The algorithm will make use of the set of *neighbors* of a vertex $v = (\boldsymbol{f}_{\text{swg}}, \boldsymbol{f}_{\text{sup}})$, defined as

$$\mathcal{N}(v) = \{v' = (\boldsymbol{f}'_{\text{swg}}, \boldsymbol{f}'_{\text{sup}}) \in \mathcal{T} : d_1(\boldsymbol{f}_{\text{sup}}, \boldsymbol{f}'_{\text{sup}}) \leq \bar{d}_1\}, \tag{8}$$

where

$$d_1(\boldsymbol{f}, \boldsymbol{f}') = \|\boldsymbol{p}_f - \boldsymbol{p}'_f\| + \sigma|\boldsymbol{\phi}_f - \boldsymbol{\phi}'_f| \tag{9}$$

is the footstep-to-footstep metric, with $\sigma \geq 0$, and $\bar{d}_1$ is a threshold distance.

4.2.2. Selecting a Vertex for Expansion

A point $\boldsymbol{p}^{\text{rand}}$ is randomly selected in $\mathbb{R}^3$, and the vertex v^{near} that is closest to $\boldsymbol{p}^{\text{rand}}$ is identified according to the following vertex-to-point metric:

$$d_2(v, \boldsymbol{p}) = \|\boldsymbol{m}(v) - \boldsymbol{p}\|_{\boldsymbol{W}} + |\psi(v, \boldsymbol{p})|, \tag{10}$$

where $\boldsymbol{m}(v)$ is the midpoint between the feet at stance v, $\boldsymbol{W}$ is a weight matrix, and $\psi(v, \boldsymbol{p})$ is the angle between the ground projection of the robot sagittal axis (whose orientation is the average of the yaw angles of the two footsteps) and the ground projection of the line joining $\boldsymbol{m}_{xy}(v)$ to $\boldsymbol{p}_{xy}$. Matrix $\boldsymbol{W}$ can be used to weigh differently vertical and horizontal displacements.

4.2.3. Generating a Candidate Vertex

After identifying the vertex $v^{\text{near}} = (\boldsymbol{f}^{\text{near}}_{\text{swg}}, \boldsymbol{f}^{\text{near}}_{\text{sup}})$, a candidate footstep is generated in the kinematically admissible region associated to $\boldsymbol{f}^{\text{near}}_{\text{sup}}$. For simplicity, denote by $(\boldsymbol{p}^{\text{near}}, \boldsymbol{\phi}^{\text{near}})$ the pose of $\boldsymbol{f}^{\text{near}}_{\text{sup}}$. We first compute the *steppable region* $\mathcal{S}$ from $\boldsymbol{f}^{\text{near}}_{\text{sup}}$, i.e., the intersection between $\mathcal{K}_{\text{pos}}(\boldsymbol{p}^{\text{near}}, \boldsymbol{\phi}^{\text{near}})$ and the collection $\mathcal{M}$ of planar regions that constitute the world:

$$\mathcal{S} = \{\mathcal{K}_{\text{pos}}(\boldsymbol{p}^{\text{near}}, \boldsymbol{\phi}^{\text{near}}) \cap \mathcal{M}\}. \tag{11}$$

By construction, $\mathcal{S}$ itself will be a collection of planar regions. One of these regions is randomly selected, and a sample point p^{cand} is extracted from it. Also, a random yaw angle γ^{cand} is generated in $[\gamma^{\text{near}} - \Delta^-_\gamma, \gamma^{\text{near}} + \Delta^+_\gamma]$. Finally, we perform *foot snapping*, a process that correctly places the footstep at p^{cand}, with yaw angle γ^{cand}, by matching its pitch and roll angles to those of the associated planar region of $\mathcal{S}$.

Once the candidate footstep $\boldsymbol{f}^{\text{cand}}$ has been generated, it can be checked with respect to requirement R1 and R2. For the latter, note that the position constraint (5) and the last of the angular constraints (6) are automatically satisfied by the above procedure and, therefore, only the pitch and roll rows of (6) must be explicitly checked.

In order to check requirement R3, we invoke an engine that generates a swing foot trajectory $\boldsymbol{s}^{\text{near}}$ from $\boldsymbol{f}^{\text{near}}_{\text{swg}}$ to $\boldsymbol{f}^{\text{cand}}_{\text{sup}}$. Such an engine uses a Bezier curve as a parameterized trajectory; given the endpoints, such a trajectory can be deformed through control points to change the apex height h along the motion. Once a a swing foot trajectory has been generated, it is checked for collision with the environment: if a collision is detected, h is increased and the process is repeated. If a maximum height $h_{\max}$ is reached without success, the candidate footstep is discarded.

If R1–R3 are satisfied, the candidate vertex is generated as $v^{\text{cand}} = (\boldsymbol{f}^{\text{cand}}_{\text{swg}}, \boldsymbol{f}^{\text{cand}}_{\text{sup}})$, with $\boldsymbol{f}^{\text{cand}}_{\text{swg}} = \boldsymbol{f}^{\text{near}}_{\text{sup}}$; however, adding v^{cand} to $\mathcal{T}$ is postponed, as the planner first has to choose a parent for it. If, instead, one of the requirements is violated, we abort the current expansion and start a new iteration.

4.2.4. Choosing a Parent

Although the procedure for generating v^{cand} originated at v^{near}, a different vertex in the tree might lead to v^{cand} with a lower cost. To identify such a vertex, the algorithm examines each vertex $v' = (\boldsymbol{f}'_{\text{swg}}, \boldsymbol{f}'_{\text{sup}}) \in \mathcal{N}(v^{\text{cand}})$ to determine whether choosing v' as a parent of v^{cand} satisfies requirements R2-R3 and whether this new connection decreases the cost of v^{cand}, that is

$$c(v') + l(v', v^{\text{cand}}) < c(v^{\text{near}}) + l(v^{\text{near}}, v^{\text{cand}}). \tag{12}$$

The new parent of v^{cand} is chosen as the vertex $v^{\min} = (f_{\text{swg}}^{\min}, f_{\text{sup}}^{\min})$, which allows v^{cand} to be reached with minimum cost. If $v^{\min} \neq v^{\text{near}}$, the candidate vertex v^{cand} must be modified by relocating its swing footstep to the support footstep of $v^{\min}$: in particular, a new vertex $v^{\text{new}} = (f_{\text{swg}}^{\text{new}}, f_{\text{sup}}^{\text{new}})$ with $f_{\text{swg}}^{\text{new}} = f_{\text{sup}}^{\min}$ and $f_{\text{sup}}^{\text{new}} = f_{\text{sup}}^{\text{cand}}$ is generated and added to $\mathcal{T}$ as a child of $v^{\min}$. The edge joining $v^{\min}$ to v^{new} represents the swing foot trajectory $\boldsymbol{s}^{\min}$.

4.2.5. Rewiring

The last step of each iteration is rewiring, i.e., verifying whether the new node v^{new} allows some parts of $\mathcal{T}$ to be reached with a lower cost and modifying the tree in accordance. In particular, for each $v' = (f'_{\text{swg}}, f'_{\text{sup}}) \in \mathcal{N}(v^{\text{new}})$, the algorithm checks whether choosing v^{new} as the parent of v' satisfies requirements R2–R3 and whether this connection results in a cost reduction for v', that is

$$c(v^{\text{new}}) + l(v^{\text{new}}, v') < c(v'). \tag{13}$$

In this case, v' is modified (similarly to what was done in Section 4.2.4) by relocating its swing footstep to $f_{\text{sup}}^{\text{new}}$ and then reconnected to $\mathcal{T}$ as a child of v_{new}, with the corresponding edge representing the new swing foot trajectory. Finally, for each child $v'' = (f''_{\text{swg}}, f''_{\text{sup}})$ of v', the swing foot trajectory from the relocated f'_{swg} to f''_{sup} is generated and the edge between v' and v'' is updated.

4.3. Pseudocode

For convenience, the pseudocode of the footstep planning algorithm so far described is given in Algorithm 1.

Algorithm 1 FootstepPlanner()

Require: $f_{\text{swg}}^{\text{ini}}, f_{\text{sup}}^{\text{ini}}, \Delta T, \mathcal{M}$
  Initialize($\mathcal{T}, (f_{\text{swg}}^{\text{ini}}, f_{\text{sup}}^{\text{ini}})$)
  while TimeNotExpired(ΔT) **do**
    $\boldsymbol{p}^{\text{rand}} \leftarrow$ RandomPointGenerator()
    $v^{\text{near}} \leftarrow$ NearestNeighborVertex($\mathcal{T}$, $\boldsymbol{p}^{\text{rand}}$)
    $f^{\text{cand}} \leftarrow$ GenerateCandidateFootstep($f_{\text{sup}}^{\text{near}}, \mathcal{M}$)
    if R1(f^{cand}) AND R2(f^{cand}) **then**
      $\boldsymbol{s}^{\text{near}} \leftarrow$ SwingTrajectoryGenerator($f_{\text{swg}}^{\text{near}}, f_{\text{sup}}^{\text{cand}}$)
      if R3($\boldsymbol{s}^{\text{near}}$) **then**
        $v^{\text{cand}} \leftarrow (f_{\text{sup}}^{\text{near}}, f_{\text{sup}}^{\text{cand}})$
        $\mathcal{N} \leftarrow$ Neighbors($\mathcal{T}, v^{\text{cand}}$)
        $(v^{\min}, \boldsymbol{s}^{\min}) \leftarrow$ ChooseParent($\mathcal{T}, \mathcal{N}, v^{\text{near}}, v^{\text{cand}}, \boldsymbol{s}^{\text{near}}$)
        $v^{\text{new}} \leftarrow (f_{\text{sup}}^{\min}, f_{\text{sup}}^{\text{cand}})$
        $\mathcal{T} \leftarrow$ AddVertex($v^{\min}, v^{\text{new}}, \boldsymbol{s}^{\min}$)
        Rewire($\mathcal{T}, \mathcal{N}, v^{\text{new}}$)
      end if
    end if
  end while
  return

When the time budget ΔT expires, we stop tree expansion and retrieve $\mathcal{V}_{\text{goal}}$, the set of vertexes v such that $\boldsymbol{p}_{f,\text{sup}} \in \mathcal{G}$. From this, we extract the vertex with minimum cost and obtain the optimal footstep plan $\mathcal{P}$ as the branch of $\mathcal{T}$ joining the root to v^*.

4.4. Footstep Planning: Numerical Results

The proposed footstep planner was implemented in C++ 17 on a laptop equipped with an AMD Ryzen 7 6800HS CPU and an NVIDIA GeForce RTX 3050 GPU. The chosen humanoid was JVRC1, an open-source virtual robot with a total of 44 joints created for the Japan Virtual Robotics Challenge [17]. JVRC1 is similar to the HRP series of humanoids in terms of mechanical design, thus representing a realistic yet convenient simulation platform.

A campaign of planning experiments took place over four scenarios of varying complexity (see Figure 5):

- *Single Floor*. This environment included elevation changes, ramps (both ascending and descending), and obstacles.
- *Multi-Floor with Stairs*. This was an environment with two floors connected by 23 steps.
- *Spiral Staircase*. A spiral staircase with 26 steps connected two floors.
- *Multi-Floor with Ramps*. In this environment, the two floors were connected by four inclined ramps and three landings.

In all scenarios, the robot had to reach a circular goal region with a radius of 0.3 m. In the footstep planner, we set $h_{\max} = 0.19$ m, $\Delta_x^- = 0.05$ m, $\Delta_x^+ = 0.30$ m, $\Delta_y^- = 0.20$ m, $\Delta_y^+ = 0.30$ m, $\Delta_z^- = \Delta_z^+ = 0.12$ m, $\Delta_\alpha^- = \Delta_\alpha^+ = 0.175$ rad, $\Delta_\beta^- = \Delta_\beta^+ = 0.175$ rad, $\Delta_\gamma^- = \Delta_\gamma^+ = 0.35$ rad, $z_b = 0.3$ m, $h_b = 1.2$ m, and $r_b = 0.25$ m.

Figure 5 shows some examples of successful footstep plans obtained in the four scenarios using the proposed algorithm. Table 1 gives details about the performance of the planner in each scenario, for different values of the time budget, with each row reporting average results over 30 runs of the planner (the first plan was the time needed by the footstep planner to generate the first plan that reached the goal). A run was considered unsuccessful if the planner terminated without placing any footstep in the goal region. Note that increasing the time budget consistently improved the success rate, suggesting that the proposed footstep planner is probabilistically complete.

Overall, the results highlight the effectiveness of the proposed planner in rather complex 3D scenarios, even in the presence of stairs and ramps. See the concluding section for some hints at possible improvements of our planner.

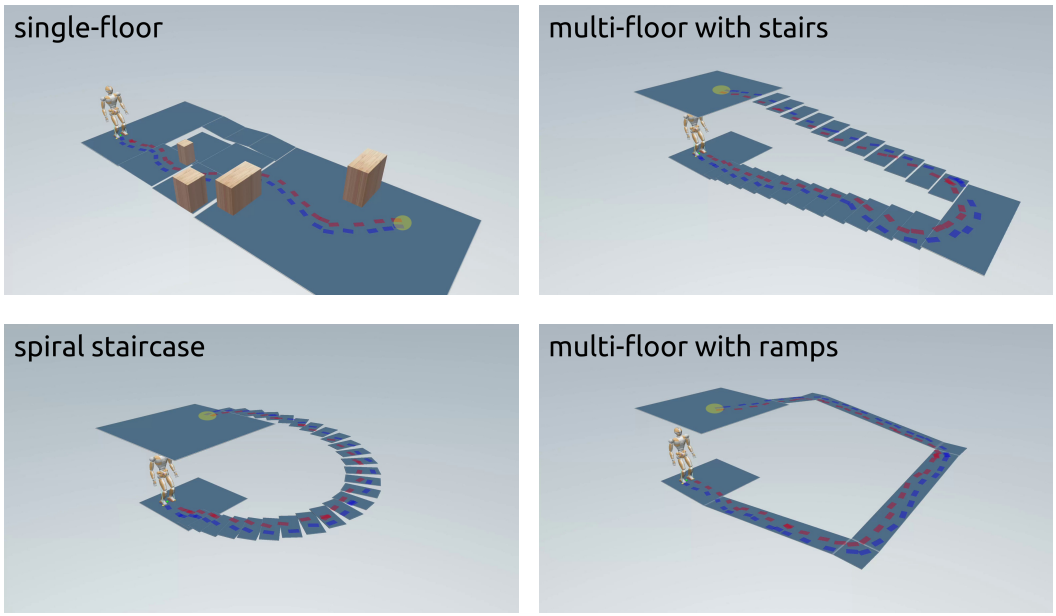

Figure 5. Examples of footstep plans found in the four considered scenarios.

Table 1. Performance of the footstep planner in the four scenarios.

Scenario	Time Budget [s]	Avg Cost	Min Cost	Max Cost	Iters	Tree Size	1st Plan [s]	Successes
Single-Floor	10	57.3	46.0	67.0	10,685	7392	4.0	19/30
	30	61.5	49.0	83.0	18,458	12,890	11.22	28/30
	60	61.3	46.0	75.0	26,370	18,457	13.67	29/30
Multi-Floor with Stairs	10	94.0	86.0	104.0	11,290	7026	6.27	14/30
	30	94.2	85.0	102.0	19,297	12,500	8.94	27/30
	60	93.3	78.0	103.0	27,933	18,119	12.51	29/30
Spiral Staircase	10	73.3	68.0	114.0	13,316	6654	3.98	24/30
	30	72.8	59.0	108.0	21,771	12,291	5.37	29/30
	60	66.9	59.0	73.0	29,247	17,571	5.59	30/30
Multi-Floor with Ramps	10	103.7	100.0	111.0	10,382	7997	8.64	3/30
	30	110.0	96.0	123.0	17,765	13,876	19.73	19/30
	60	107.5	94.0	130.0	25,924	19,695	21.88	29/30

5. Control Architecture

This section describes the control pipeline, which consisted of the IS-MPC module followed by the WBC module (see Figure 2).

5.1. IS-MPC Gait Generation

Gait generation was the process of generating a feasible trajectory for the CoM position $\boldsymbol{p}_c = (x_c, y_c, z_c)$, such that the robot realized the assigned footstep plan and maintained balance. This last requirement could be formulated as a constraint on the position $\boldsymbol{p}_z = (x_z, y_z, z_z)$ of the Zero Moment Point (ZMP), which was the point of application of the equivalent ground reaction force.

To generate the CoM trajectory, we used MPC, which optimized trajectories over a certain horizon, using a model to predict the evolution of the system.

5.1.1. Prediction Model

Our prediction model is a 3D extension of the classic LIP, which has the ZMP as input and allows for vertical motion of the CoM [14]. By neglecting the internal angular momentum variation, we have

$$\begin{aligned} (z_c - z_z)\,\ddot{x}_c &= (x_c - x_z)(\ddot{z}_c + g) \\ (z_c - z_z)\,\ddot{y}_c &= (y_c - y_z)(\ddot{z}_c + g), \end{aligned}$$

where g is the gravity acceleration. To remove the nonlinear coupling between the horizontal and vertical components, we impose $(\ddot{z}_c + g)/(z_c - z_z) = \eta^2$, where η is a constant parameter. The resulting model is expressed as

$$\ddot{\boldsymbol{p}}_c = \eta^2(\boldsymbol{p}_c - \boldsymbol{p}_z) + \boldsymbol{g}, \tag{14}$$

where $\boldsymbol{g} = (0, 0, -g)$ is a constant drift. This means that the 3D LIP is at an equilibrium when the CoM and ZMP are vertically displaced by η/g; this should be taken into account when choosing the value of η. In our formulation, we dynamically extended (14) so that the input was the ZMP velocity $\dot{\boldsymbol{p}}_z$, rather than the position $\boldsymbol{p}_z$, in order to generate smoother trajectories.

5.1.2. ZMP Constraints

In the classic LIP, the condition for balance requires the ZMP to stay within of the support polygon of the robot, which, however, is no more defined in a 3D setting. A sufficient

condition for ensuring balance in this case is that the ZMP must belong to a pyramidal region $\mathcal{Z}$, with apex at the robot CoM and edges going through the vertexes of the convex hull of the contact surfaces. Since this condition is nonlinear, we resorted to the following conservative approximation.

Consider a *moving box* $\mathcal{B}$ with constant dimensions (d_x, d_y, d_z) along the three axes. Its center $\boldsymbol{p}_{\text{mc}} = (x_{\text{mc}}, y_{\text{mc}}, z_{\text{mc}})$ moves in the following way: during single support phases, it coincides with the position of the support foot, while, during double support phases, it performs a roto-translation from the position of the previous support foot to the position of the next. By properly choosing d_x, d_y, and d_z, one may guarantee that $\mathcal{B}$ is always contained in $\mathcal{Z}$ [15].

At a generic time instant t_j, a linear ZMP constraint can then be written as

$$-\frac{1}{2}\begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} \leq \boldsymbol{R}_j^T(\boldsymbol{p}_z^j - \boldsymbol{p}_{\text{mc}}^j) \leq \frac{1}{2}\begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}, \tag{15}$$

where $\boldsymbol{R}_j$ is the rotation matrix associated with the orientation of the moving box at t_j and the j superscript denotes the variable at t_j.

5.1.3. Stability Constraint

While keeping the ZMP inside $\mathcal{B}$ (and therefore, inside $\mathcal{Z}$) is sufficient for balance, it does not by itself guarantee that the COM trajectories are bounded with respect to the ZMP. This is due to the well-known fact that the LIP model includes one unstable mode, which can be isolated by defining the variable $\boldsymbol{p}_u = \boldsymbol{p}_c + \dot{\boldsymbol{p}}_c/\eta$.

To avoid divergence, IS-MPC enforces the following *stability condition*:

$$\boldsymbol{p}_u^k = \eta \int_{t_k}^{\infty} e^{-\eta(\tau - t_k)} \boldsymbol{p}_z(\tau) d\tau + \frac{\boldsymbol{g}}{\eta}. \tag{16}$$

This is a relationship between the current value of $\boldsymbol{p}_u$ and the future of the ZMP trajectory and is, therefore, noncausal. To obtain a causal condition, we split the integral in two parts, the first from t_k to t_{k+C} (the final instant of the prediction horizon) and the second from t_{k+C} to infinity. While the first integral could be expressed in terms of the MPC decision variables $\dot{\boldsymbol{p}}_z^k, \ldots, \dot{\boldsymbol{p}}_z^{k+C-1}$, the second could be computed by conjecturing a trajectory $\tilde{\boldsymbol{p}}_z$ of $\boldsymbol{p}_z$ (*anticipative tail*) on the basis of information contained in the footstep plan:

$$\boldsymbol{c}_u^k = \eta \int_{t_{k+C}}^{\infty} e^{-\eta(\tau - t_k)} \tilde{\boldsymbol{p}}_z(\tau) d\tau. \tag{17}$$

The stability constraint could then be written as

$$\boldsymbol{p}_u^k = \eta \int_{t_k}^{t_{k+C}} e^{-\eta(\tau - t_k)} \boldsymbol{p}_z(\tau) d\tau + \boldsymbol{c}_u^k + \frac{\boldsymbol{g}}{\eta}. \tag{18}$$

5.1.4. IS-MPC

At each iteration, IS-MPC solved the following quadratic program:

$$\begin{aligned} \min_{\dot{\boldsymbol{p}}_z^k, \ldots, \dot{\boldsymbol{p}}_z^{k+C-1}} \quad & \sum_{i=k}^{k+C-1} ||\dot{\boldsymbol{p}}_z^i||^2 + \lambda ||\boldsymbol{p}_z^i - \boldsymbol{p}_{\text{mc}}^i||^2 \\ \text{subject to:} \quad & \text{ZMP constraints (15), for } j = k, \ldots, k+C \\ & \text{stability constraint (18)}, \end{aligned} \tag{19}$$

where λ is a positive weight. After solving the quadratic program, the first optimal sample $\dot{\boldsymbol{p}}_z^k$ was used to integrate the dynamically extended model (14) and obtain the next sample of the reference CoM position $\boldsymbol{p}_c^*$ and velocity $\dot{\boldsymbol{p}}_c^*$, as well as of the corresponding ZMP position $\boldsymbol{p}_z^*$. The corresponding CoM acceleration could then be computed from (14). All these terms were sent to the WBC, where they were used to set up the CoM tracking task.

5.2. Whole-Body Controller

Commands to be sent to the robot joints were generated by a whole-body controller that worked in two stages. The first stage determined appropriate joint accelerations and contact forces by solving a constrained quadratic program and the second stage computed the associated joint torques via inverse dynamics.

5.2.1. Task References

The input to the WBC is a set of reference trajectories, one for each task. The set of tasks was $\mathcal{T} = \{com, left, right, torso, joint\}$, which respectively represented

- *com*: the CoM reference, generated by the IS-MPC module;
- *left/right*: foot pose references, generated by the footstep planner;
- *torso*: a reference torso orientation, typically chosen to be upright;
- *joint*: a reference joint posture (for solving kinematic redundancy).

We denoted the generic task variable by $\boldsymbol{t}_i$, $i = 1, \ldots, n_t$, and its reference position, velocity, and acceleration by $\boldsymbol{t}_i^*$, $\dot{\boldsymbol{t}}_i^*$, and $\ddot{\boldsymbol{t}}_i^*$, respectively (with a little abuse as we also used a time derivative notation for angular task variables). The desired acceleration $\ddot{\boldsymbol{t}}_i$ was then defined as

$$\ddot{\boldsymbol{t}}_i = \ddot{\boldsymbol{t}}_i^* + k_p(\boldsymbol{t}_i^* - \boldsymbol{t}_i) + k_d(\dot{\boldsymbol{t}}_i^* - \dot{\boldsymbol{t}}_i), \tag{20}$$

where the reference acceleration was used as feedforward and the PD feedback was aimed at robustifying kinematic inversion. The desired acceleration $\ddot{\boldsymbol{t}}_i$ was realized provided that the following relationship held:

$$\boldsymbol{J}_i\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}_i\boldsymbol{\nu} = \ddot{\boldsymbol{t}}_i^* + k_p(\boldsymbol{t}_i^* - \boldsymbol{t}_i) + k_d(\dot{\boldsymbol{t}}_i^* - \dot{\boldsymbol{t}}_i), \tag{21}$$

where $\boldsymbol{J}_i$ is the Jacobian of the i-th task.

5.2.2. Robot Dynamics

The robot dynamics can be written as

$$\boldsymbol{M}(\boldsymbol{q})\,\dot{\boldsymbol{\nu}} + \boldsymbol{n}(\boldsymbol{q}, \boldsymbol{\nu}) = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\tau} \end{pmatrix} + \sum_{i=1}^{n_c} \boldsymbol{J}_{c,i}^T(\boldsymbol{q})\boldsymbol{f}_{c,i}. \tag{22}$$

Here, $\boldsymbol{M}$ is the inertia matrix; the configuration $\boldsymbol{q} = (\boldsymbol{q}_u, \boldsymbol{q}_a)$ is the stack of the floating base (unactuated) variables and the joint (actuated) variables; $\boldsymbol{\nu} = (\boldsymbol{\nu}_u, \dot{\boldsymbol{q}}_a)$ contains the corresponding pseudovelocities; $\boldsymbol{n}$ collects Coriolis, centrifugal, and gravitational terms; $\boldsymbol{\tau}$ are the joint torques; and $\boldsymbol{J}_{c,i}$ denotes the Jacobian of the point of application of the i-th contact force $\boldsymbol{f}_{c,i}$, with $i = 1, \ldots, n_c$.

Equation (22) can be partitioned to highlight the unactuated and the actuated dynamics:

$$\begin{pmatrix} \boldsymbol{M}_u \\ \boldsymbol{M}_a \end{pmatrix}\dot{\boldsymbol{\nu}} + \begin{pmatrix} \boldsymbol{n}_u \\ \boldsymbol{n}_a \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\tau} \end{pmatrix} + \sum_{i=1}^{n_c} \begin{pmatrix} \boldsymbol{J}_{ci,u}^T \\ \boldsymbol{J}_{ci,a}^T \end{pmatrix} \boldsymbol{f}_{c,i}, \tag{23}$$

We omit the dependence on $\boldsymbol{q}$ and $\boldsymbol{\nu}$ for compactness.

5.2.3. Constraints

The first constraint enforces the unactuated part of (23):

$$\boldsymbol{M}_u\,\dot{\boldsymbol{\nu}} + \boldsymbol{n}_u = \sum_{i=1}^{n_c} \boldsymbol{J}_{ci,u}^T \boldsymbol{f}_{c,i}. \tag{24}$$

We did not directly enforce the actuated dynamics (which would have required adding $\boldsymbol{\tau}$ as a decision variable) because $\boldsymbol{\tau}$ did not explicitly appear in our quadratic program; therefore, it was computed a posteriori from the inverse dynamics without affecting the optimization process (if one wants to minimize torque effort or impose torque limits, $\boldsymbol{\tau}$ must be included in the decision variables, and then the actuated dynamics should be added as a constraint).

The second set of constraints guarantee stationary contacts by imposing zero acceleration to the left and/or right foot, depending on the specific support phase:

$$\boldsymbol{J}_{\text{left/right}}\,\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}_{\text{left/right}}\,\boldsymbol{\nu} = \boldsymbol{0}. \tag{25}$$

The third set of constraints enforce limits on joint velocities and positions:

$$\dot{\boldsymbol{q}}_{a,\min} \leq \dot{\boldsymbol{q}}_a + \delta\,\ddot{\boldsymbol{q}}_a \leq \dot{\boldsymbol{q}}_{a,\max} \tag{26}$$

$$\boldsymbol{q}_{a,\min} \leq \boldsymbol{q}_a + \delta\,\dot{\boldsymbol{q}}_a + \tfrac{\delta^2}{2}\,\ddot{\boldsymbol{q}}_a \leq \boldsymbol{q}_{a,\max}, \tag{27}$$

where δ is the duration of a sampling interval.

The last set of constraints require each contact force $\boldsymbol{f}_{c,i}$ to be contained in the friction cone, ensuring that no slipping occurs:

$$-\mu\,f_{ci,n} \leq f_{ci,x} \leq \mu\,f_{ci,n} \tag{28}$$

$$-\mu\,f_{ci,n} \leq f_{ci,y} \leq \mu\,f_{ci,n}, \tag{29}$$

where μ is the friction coefficient, $f_{ci,n}$ is the normal component of $\boldsymbol{f}_{c,i}$, and $f_{ci,x}, f_{ci,y}$ are its tangential components along the x and y axes of the foot frame at the contact. Note that *(i)* a pyramidal approximation of the friction cone was used to maintain linearity and *(ii)* these constraints automatically implied $f_{ci,n} > 0$, i.e., unilaterality of the contact forces.

5.2.4. Quadratic Program

At each iteration, the first stage of the WBC solved the following QP problem:

$$\begin{aligned} \min_{\dot{\boldsymbol{\nu}},\,f_1,\ldots,f_{n_c}} \quad & \sum_{i=1}^{n_t} w_i \left\| \boldsymbol{J}_i\dot{\boldsymbol{\nu}} + \dot{\boldsymbol{J}}_i\,\boldsymbol{\nu} - \ddot{\boldsymbol{t}}_i^{*} - k_p(\boldsymbol{t}_i^{*} - \boldsymbol{t}_i) - k_d(\dot{\boldsymbol{t}}_i^{*} - \dot{\boldsymbol{t}}_i) \right\|^2 \\ \text{subject to} \quad & \text{unactuated dynamics (24)}, \\ & \text{stationary contact constraints (25)}, \\ & \text{joint velocity constraints (26)}, \\ & \text{joint position constraints (27)}, \\ & \text{contact force constraints (28), (29)}, \; i = 1, \ldots, n_c. \end{aligned} \tag{30}$$

5.2.5. Inverse Dynamics

Once $\dot{\nu}$ and the contact forces $f_{c,i}$ were obtained by solving the above QP, the joint torques could be computed using the actuated dynamics in (23), i.e.,

$$\tau = M_a \dot{\nu} + n_a - \sum_{i=1}^{n_c} J_{ci,a}^T f_{c,i}. \tag{31}$$

6. Simulations

We now complement the footstep planning results of Section 4.4 by presenting dynamic simulations of the JVRC1 robot executing the planned motions in MuJoCo, which guaranteed high physical accuracy. IS-MPC ran at 100 Hz and used a sampling interval of 0.1 s and a control horizon of 2 s, whereas the WBC ran at 1000 Hz (i.e., $\delta = 1$ ms). Contacts were modeled with four forces per foot, so that $n_c = 4$ or 8 depending on the support phase. Finally, all quadratic programs (QPs) were solved with HPIPM [18], which required under 1 ms to compute a solution, ensuring real-time capability.

The simulation results are shown via successive snapshots in Figures 6–10 and in the form of video clips at https://youtu.be/RFPrQjY2tD8 (accessed on 11 June 2025). As expected, the robot was able to efficiently and robustly realize all footstep plans, even in the presence of stairs or inclined ramps.

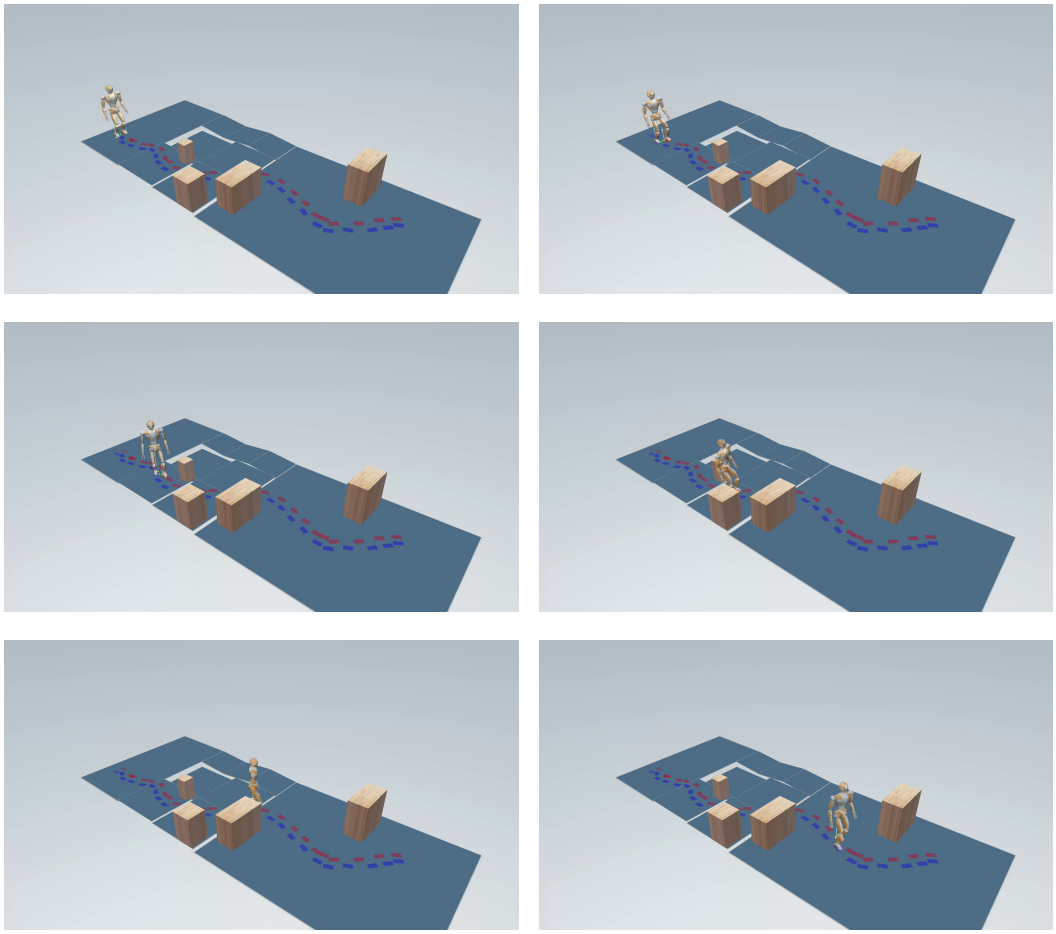

Figure 6. *Cont.*

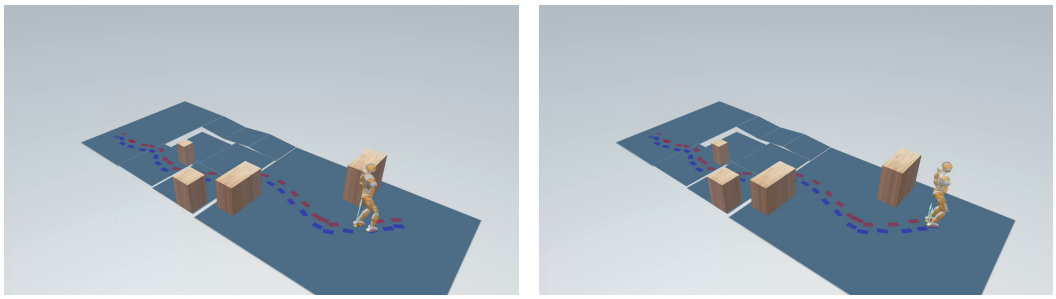

Figure 6. Single-Floor: The robot traversed some horizontal patches at different levels while treating others as obstacles due to their height. For this figure, as for all successive figures, the snapshots were hand-picked to adequately illustrate the generated motion in a compact format.

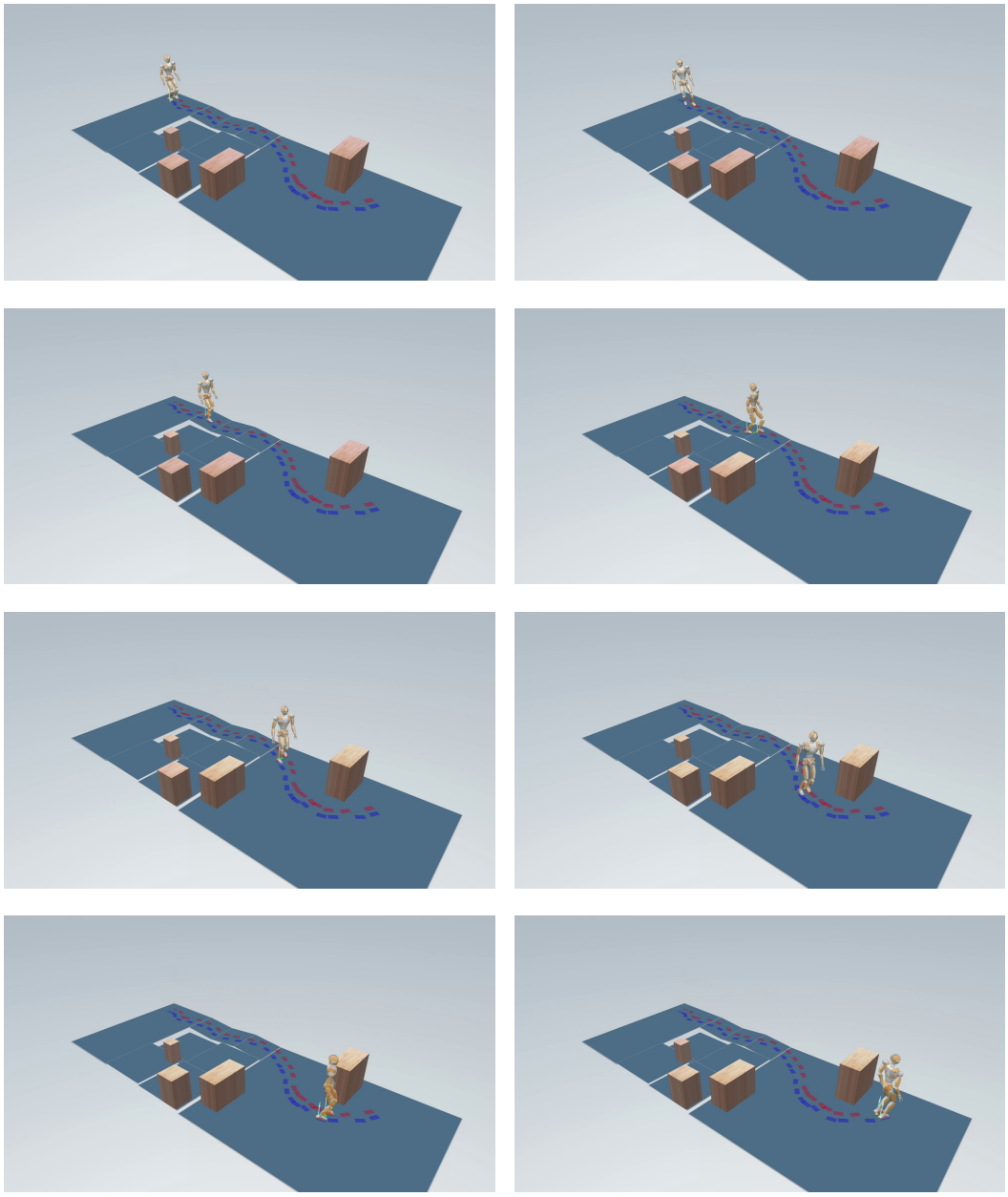

Figure 7. Single-Floor with a different start: The robot took a different route, going over some inclined ramps.

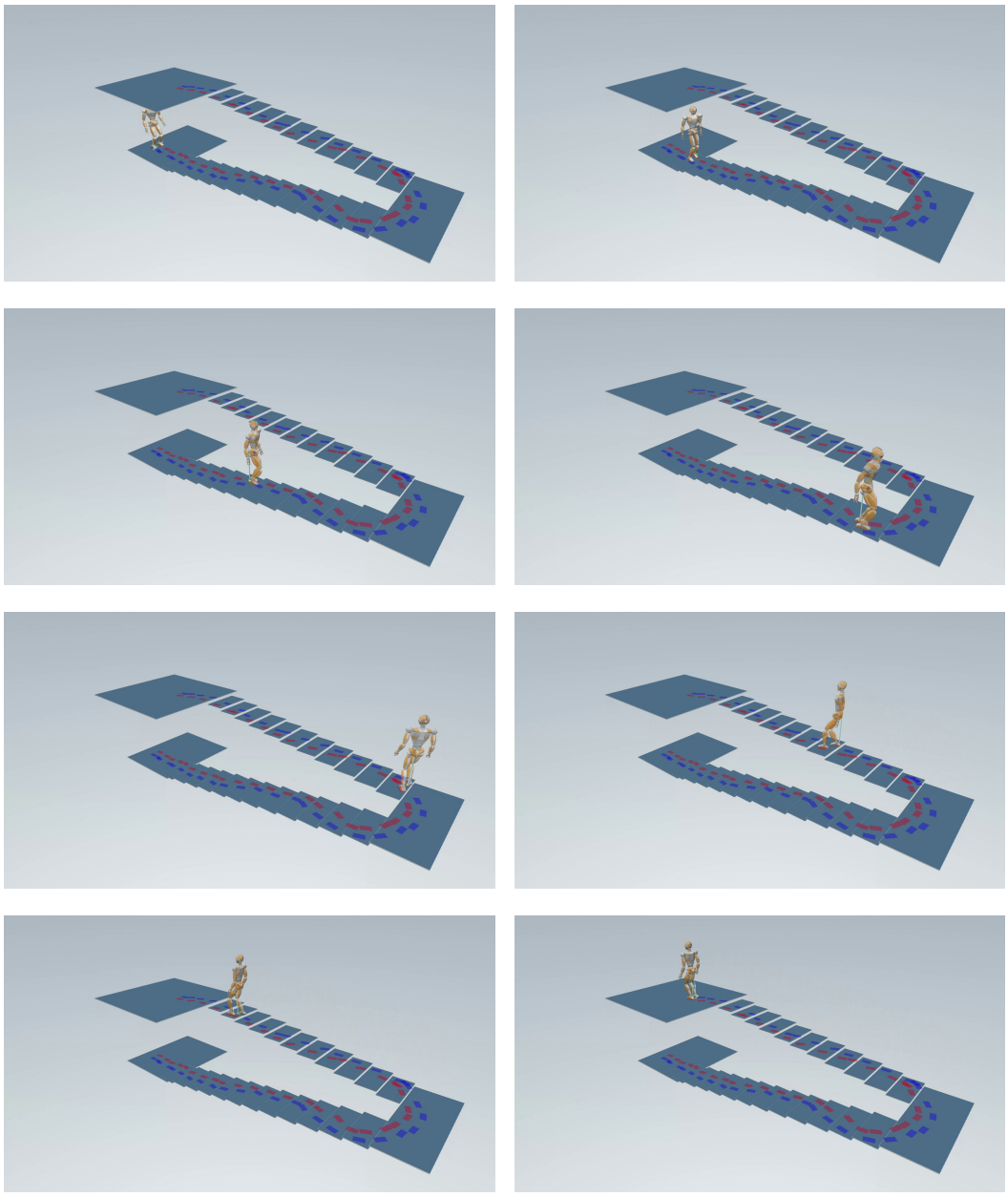

Figure 8. Multi-Floor with Stairs: Since the start and the goal were on different floors, the robot climbsed the whole staircase.

Figure 9. *Cont*.

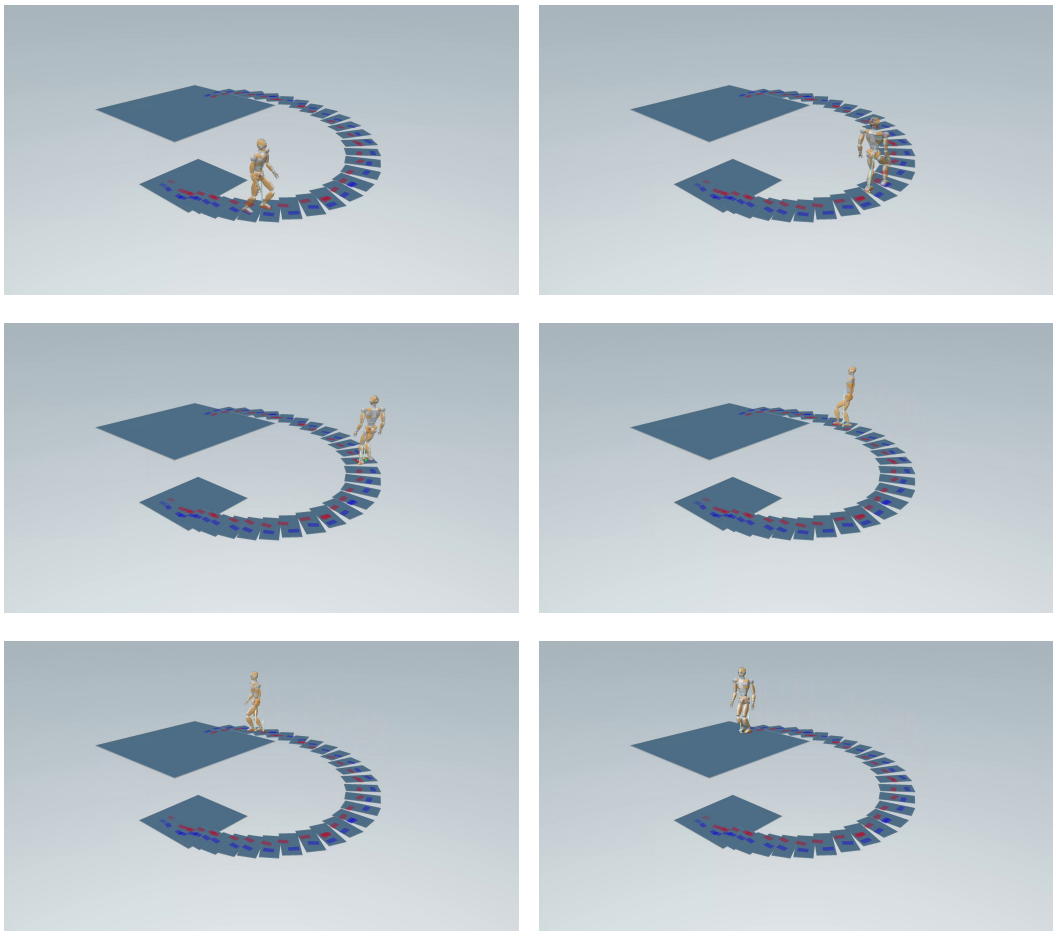

Figure 9. Spiral Staircase: As before, the robot climbed the whole spiral staircase to reach the upper floor.

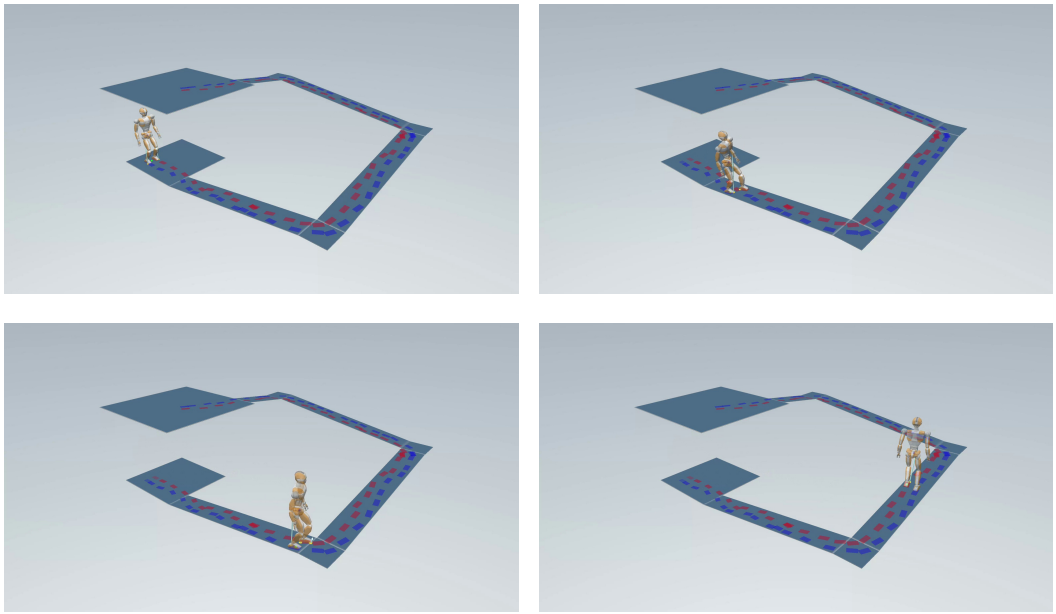

Figure 10. *Cont*.

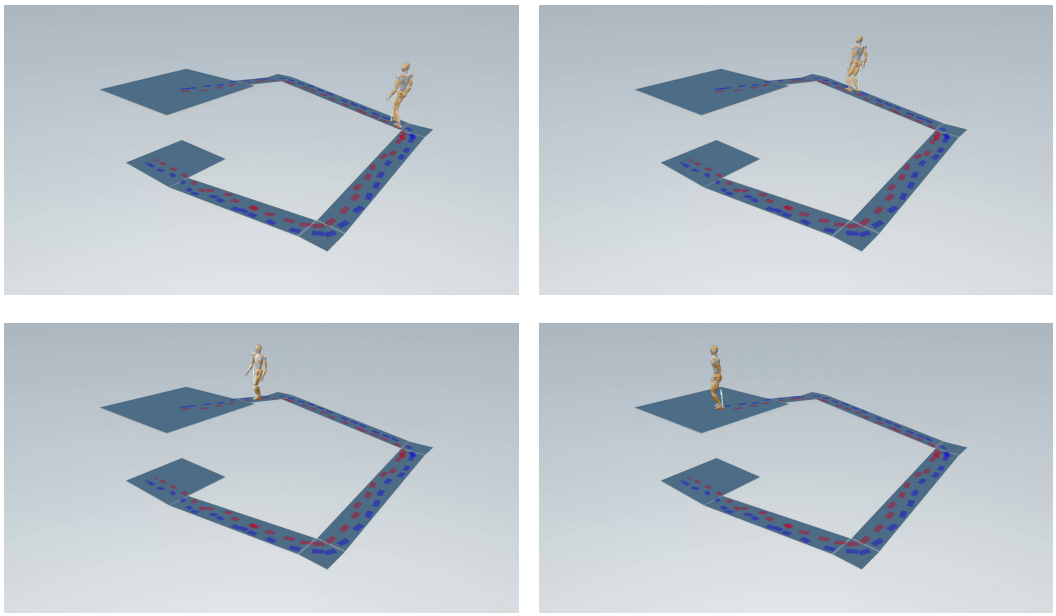

Figure 10. Multi-Floor with Ramps: In this last scenario, the robot reached the upper floor, going over a sequence of inclined ramps.

7. Conclusions

The proposed framework represents a humanoid motion generation scheme that enables navigation in complex 3D environments, including multiple floors and inclined surfaces. We ran an extensive campaign in order to validate the performance of the pipeline (footstep planner, gait generation, and whole body control) in several different scenarios via dynamic simulations on a JVRC1 robot in MuJoCo. Our results show that our framework is effective and robust in very complex environments, paving the way for further promising research.

Future work will explore several directions, including

- implementing and testing the proposed method on the Unitree G1 humanoid available in our lab;
- speeding up the footstep planner by adopting a k-d tree [16] data structure and introducing waypoints (intermediate goals);
- devising an online version of the planner that can be used as the robot moves and gathers new information about the environment;
- using a whole-body MPC for enhancing agility and allowing more complex movements.

Author Contributions: Conceptualization, M.C., N.S., L.L. and G.O.; methodology, D.M., M.C., N.S., L.L. and G.O.; software, D.M., M.C. and N.S.; validation, D.M., M.C. and N.S.; formal analysis, D.M., M.C., N.S., L.L. and G.O.; investigation, D.M., M.C., N.S., L.L. and G.O.; resources, M.C., N.S., L.L. and G.O.; data curation, D.M., M.C., N.S., L.L. and G.O.; writing—original draft preparation, D.M., M.C., N.S., L.L. and G.O.; writing—review and editing, M.C., N.S., L.L. and G.O.; visualization, D.M., M.C., N.S., L.L. and G.O.; supervision, M.C., N.S., L.L. and G.O.; project administration, M.C., N.S., L.L. and G.O.; funding acquisition, L.L. and G.O. All authors have read and agreed to the published version of the manuscript.

Funding: Nicola Scianca was fully supported by the PNRR MUR project PE0000013-FAIR.

Data Availability Statement: Data will be made available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hauser, K.; Bretl, T.; Latombe, J.C.; Harada, K.; Wilcox, B. Motion Planning for Legged Robots on Varied Terrain. *Int. J. Robot. Res.* **2008**, *27*, 1325–1349. [CrossRef]
2. Chestnutt, J.; Lau, M.; Cheung, G.; Kuffner, J.; Hodgins, J.; Kanade, T. Footstep Planning for the Honda ASIMO Humanoid. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 629–634.
3. Gutmann, J.S.; Fukuchi, M.; Fujita, M. Real-time path planning for humanoid robot navigation. In Proceedings of the 19th International Joint Conference on Artificial Intelligence, Edinburgh, UK, 30 July–5 August 2005; pp. 1232–1237.
4. Griffin, R.J.; Wiedebach, G.; McCrory, S.; Bertrand, S.; Lee, I.; Pratt, J. Footstep Planning for Autonomous Walking Over Rough Terrain. In Proceedings of the 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), Toronto, ON, Canada, 15–17 October 2019; pp. 9–16.
5. Mishra, B.; Calvert, D.; Bertrand, S.; McCrory, S.; Griffin, R.; Sevil, H.E. GPU-accelerated rapid planar region extraction for dynamic behaviors on legged robots. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 8493–8499.
6. Mishra, B.; Calvert, D.; Bertrand, S.; Pratt, J.; Sevil, H.E.; Griffin, R. Efficient Terrain Map Using Planar Regions for Footstep Planning on Humanoid Robots. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024; pp. 8044–8050.
7. Liu, H.; Sun, Q.; Zhang, T. Hierarchical RRT for Humanoid Robot Footstep Planning with Multiple Constraints in Complex Environments. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 3187–3194.
8. Wieber, P.B. Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, 4–6 December 2006; pp. 137–142.
9. Englsberger, J.; Ott, C.; Albu-Schäffer, A. Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion. *IEEE Trans. Robot.* **2015**, *31*, 355–368. [CrossRef]
10. Han, Y.H.; Cho, B.K. Slope walking of humanoid robot without IMU sensor on an unknown slope. *Robot. Auton. Syst.* **2022**, *155*, 104163. [CrossRef]
11. Wiedebach, G.; Bertrand, S.; Wu, T.; Fiorio, L.; McCrory, S.; Griffin, R.; Nori, F.; Pratt, J. Walking on partial footholds including line contacts with the humanoid robot Atlas. In Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), Cancun, Mexico, 15–17 November 2016; pp. 1312–1319.
12. Caron, S.; Pham, Q.C.; Nakamura, Y. ZMP support areas for multicontact mobility under frictional constraints. *IEEE Trans. Robot.* **2016**, *33*, 67–80. [CrossRef]
13. Scianca, N.; De Simone, D.; Lanari, L.; Oriolo, G. MPC for Humanoid Gait Generation: Stability and Feasibility. *IEEE Trans. Robot.* **2020**, *36*, 1171–1178. [CrossRef]
14. Smaldone, F.M.; Scianca, N.; Lanari, L.; Oriolo, G. From walking to running: 3D humanoid gait generation via MPC. *Front. Robot. AI* **2022**, *9*, 876613. [CrossRef] [PubMed]
15. Cipriano, M.; Ferrari, P.; Scianca, N.; Lanari, L.; Oriolo, G. Humanoid motion generation in a world of stairs. *Robot. Auton. Syst.* **2023**, *168*, 104495. [CrossRef]
16. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
17. Okugawa, M.; Oogane, K.; Shimizu, M.; Ohtsubo, Y.; Kimura, T.; Takahashi, T.; Tadokoro, S. Proposal of inspection and rescue tasks for tunnel disasters—Task development of Japan virtual robotics challenge. In Proceedings of the 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), West Lafayette, IN, USA, 18–20 October 2015; pp. 1–2.
18. Frison, G.; Diehl, M. HPIPM: A high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine* **2020**, *53*, 6563–6569. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.