2019 18th European Control Conference (ECC)
Napoli, Italy, June 25-28, 2019

An Integrated Motion Planner/Controller for Humanoid Robots on Uneven Ground

Paolo Ferrari, Nicola Scianca, Leonardo Lanari, Giuseppe Oriolo

***Abstract*— We consider a situation in which a humanoid robot must reach a goal region (*walk-to* task) walking in an environment consisting of horizontal patches located at different heights (*world of stairs*). To solve this problem, the paper proposes an integrated motion planner/controller working in two stages: off-line footstep planning and on-line gait generation. The planning stage is based on a randomized algorithm that efficiently searches for a feasible footstep sequence. The gait generation uses an intrinsically stable MPC-based control scheme which computes CoM trajectories that are suitable for walking on uneven ground. The proposed framework was implemented in the V-REP environment for the HRP4 humanoid robot and successfully tested via simulations.**

I. Introduction

With the interest in humanoids still growing in the robotics community, the skills of these robotic platforms continue to improve as a consequence of hardware and control advances. In principle, one of the advantages of humanoids is the possibility of moving through complex environments, e.g., by stepping over or onto obstacles. However, while there are a large number of locomotion techniques for flat ground, walking on uneven surfaces poses additional challenges which make it largely an open field of research.

The complexity of humanoids makes it difficult to devise control strategies based on accurate dynamic models. On flat ground, many researchers adopt a simplified model known as the Linear Inverted Pendulum (LIP) [1], in which the robot is considered a single point-mass pivoting on a foot and moving at constant height above the ground. The linearity of this model has been exploited to design preview controllers [2] first, and — to include constraints in the formulation — Model Predictive Control (MPC) schemes [3] later.

When the robot moves on uneven ground one must remove the constant height hypothesis, leading to a nonlinear inverted pendulum model. The nonlinear problem was addressed in [4] using a dual MPC scheme and, more recently, in [5] by extending capturability concepts to the 3D case.

However, keeping the linearity of the system is still an attractive option in view of the simplicity of the associated controllers and, ultimately, the possibility of an efficient real-time implementation. One way to do this consists in assuming a predefined vertical trajectory of the Center of Mass (CoM), so that the robot can be described by a time-varying LIP, as done in [6], [7] and in [8] for transitions from bipedal to quadrupedal locomotion. Another, less restrictive possibility [9] is to maintain the time-invariant LIP structure by constraining the CoM vertical motion to satisfy a certain differential equation. Related approaches [10], [11] lead to a 3D model with LIP dynamics on all three axes.

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Italy. E-mail: *lastname*@diag.uniroma1.it. This work was supported by the EU H2020 COMANOID project.

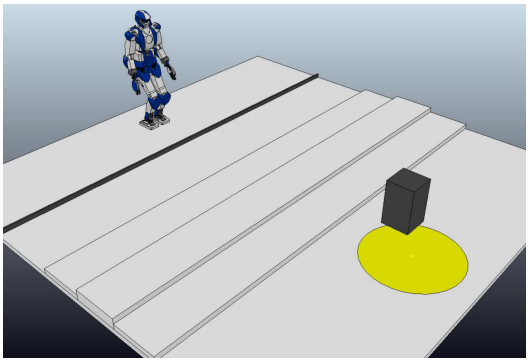

Fig. 1. An instance of the considered problem. To reach the goal region $\mathcal{G}$ (in yellow), the humanoid must go over the black bar obstacle, climb and descend the staircase, and avoid the black box.

In [12] we have introduced an MPC scheme for gait generation incorporating a novel constraint that guarantees stable CoM trajectories regardless of the length of the prediction horizon. This scheme was then extended in [13] to the case of a 3D LIP model where the CoM height can vary under the appropriate differential constraint, thereby obtaining a scheme for walking along a given footstep sequence in a world of stairs. In this paper, we integrate that gait generation scheme in a complete framework which includes a planner for generating a footstep sequence in this kind of scenario.

The paper is structured as follows. Section II defines the problem and provides an overview of the solution approach. Section III describes the footstep planning algorithm, while Sect. IV recalls the main features of the 3D gait generation method. Section V presents simulations on the HRP-4 humanoid and Sect. VI offers a few concluding remarks.

II. Formulation and Approach

The situation of interest in this paper is shown in Fig. 1. A humanoid robot is assigned a *walk-to* locomotion task to a desired goal region $\mathcal{G}$ in a *world of stairs*, a specific kind of uneven ground composed by horizontal patches located at different heights. Depending on its elevation with respect to the neighboring areas, a patch may be accessible for the humanoid to climb on from an appropriate direction, or else represent an obstacle to be avoided. Some low-height patches may be shaped in such a way that the humanoid may decide to go over (rather then stepping on) them; for example, this is the case of the long bar obstacle in Fig. 1.

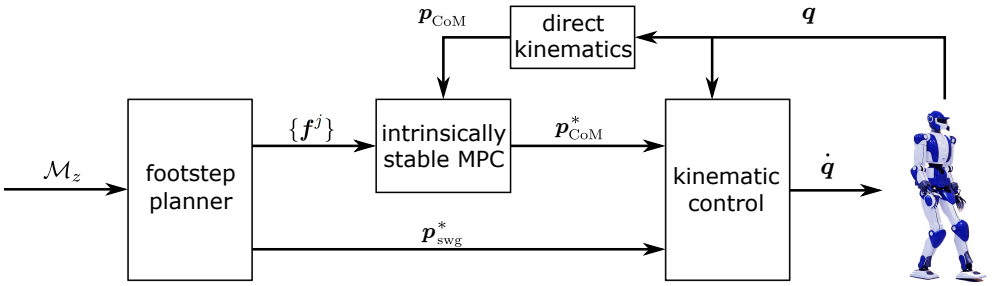

Fig. 2. Block scheme of the proposed approach.

A natural choice for representing the considered kind of ground is a 2.5D grid map of equally-sized cells, also called *elevation map* [14]. We will denote this map by $\mathcal{M}_z$, and assume that it is known in advance so that whenever needed it can be queried as $z = \mathcal{M}_z(x, y)$, to provide the height of the ground at the cell having coordinates (x, y).

To solve the given problem, we propose an integrated motion planner/controller whose block scheme is shown in Fig. 2. An off-line footstep planner is in charge of finding first an appropriate sequence of footsteps $\{\boldsymbol{f}^j\}$ leading to the desired location $\mathcal{G}$, together with the associated swing foot trajectories $\{\boldsymbol{p}^j_{\mathrm{swg}}\}$ between consecutive footsteps. Here, $\boldsymbol{f}^j = (x^j_{\mathrm{f}}, y^j_{\mathrm{f}}, z^j_{\mathrm{f}}, \theta^j_{\mathrm{f}})^T$ is the pose of the j-th footstep, with x^j_{f}, y^j_{f}, and z^j_{f} representing its position and θ^j_{f} its orientation[1].

For the plan to be feasible, each element in the sequences $\{\boldsymbol{f}^j\}$ and $\{\boldsymbol{p}^j_{\mathrm{swg}}\}$ must satisfy the following requirements:

- R1 The height variation w.r.t. the previous footstep is within a maximum range, i.e., $|z^j_{\mathrm{f}} - z^{j-1}_{\mathrm{f}}| \leq \Delta z_{\max}$.
- R2 The footstep is fully in contact within a single horizontal patch, i.e., each cell of $\mathcal{M}_z$ belonging to the footprint has the same height z^j_{f}.
- R3 Apart from the ground contact at the start and at the end, the swing foot trajectory $\boldsymbol{p}^j_{\mathrm{swg}}$ is collision-free.

Once the footstep sequence has been generated, it is passed to an on-line gait generation block based on intrinsically stable MPC, which computes a CoM trajectory $\boldsymbol{p}^*_{\mathrm{CoM}}$ compatible with the sequence. The swing foot trajectory $\boldsymbol{p}^*_{\mathrm{swg}}$ at any instant is defined by the appropriate subtrajectory $\boldsymbol{p}^j_{\mathrm{swg}}$.

Finally, the reference trajectories $\boldsymbol{p}^*_{\mathrm{CoM}}$ and $\boldsymbol{p}^*_{\mathrm{swg}}$ are used in a standard pseudoinverse-based kinematic controller to compute joint commands $\dot{\boldsymbol{q}}$ for the robot. Proprioceptive feedback is used for both MPC and kinematic control.

In the next sections we describe separately the modules for footstep planning and gait generation.

III. Footstep Planning

The footstep planning module is in charge of finding a feasible (in the sense of the requirements R1-R3) sequence of footsteps which leads the robot to the goal region $\mathcal{G}$.

A. Previous work

One possible approach to plan footsteps (on flat or uneven ground) is based on continuous optimization techniques, which do not restrict steps to a finite set [15]. However, these methods require an expensive pre-computation phase aimed at finding a convex approximation of the free configuration space. An efficient alternative is to consider a finite set of possible foot displacements, so that the solution will consist of a particular sequence of such elements. To search among all possible sequences, both deterministic and randomized approaches have been proposed.

Examples of the first kind are A^*-based techniques [16], [17], whose performance strongly depends on the chosen heuristic, which is often difficult to design. Moreover, node expansion in these techniques can be very expensive in the presence of several constraints to be verified (e.g., collision avoidance, kinematic reachability, and so on).

Randomized approaches include [18] where, to ensure goal-directedness, all possible node expansions must be evaluated at each iteration, at the expense of efficiency. To overcome such a problem, an approximate swept volume is precomputed in [19] for each possible foot displacement, thus speeding up collision checking for footsteps and swinging trajectories. This approach, however, cannot be extended to the case of variable height steps.

B. The proposed planner

In the following, we a propose a simple randomized algorithm which does not require any kind of pre-processing of the environment and allows to embed a feasibility check directly in the expansion mechanism.

Our footstep planner, whose pseudocode is given in Algorithm 1, iteratively builds a tree $\mathcal{T}$ in the search space. A vertex $v = (\boldsymbol{f}_{\mathrm{sup}}, \boldsymbol{f}_{\mathrm{swg}})$ specifies the poses $\boldsymbol{f}_{\mathrm{sup}}$ and $\boldsymbol{f}_{\mathrm{swg}}$ of the two feet during a double support phase; in all steps originating from v, the support foot will remain at $\boldsymbol{f}_{\mathrm{sup}}$ while the swinging foot will move from $\boldsymbol{f}_{\mathrm{swg}}$. An edge exists between two vertices when there exists a collision-free trajectory of the swing foot connecting the two.

The algorithm starts by rooting $\mathcal{T}$ at $v_{\mathrm{ini}} = (\boldsymbol{f}_L, \boldsymbol{f}_R)$, where $\boldsymbol{f}_L$ and $\boldsymbol{f}_R$ are the foot poses at the starting configuration. The initial support foot can be chosen arbitrarily.

The generic iteration of the algorithm begins by selecting a sample point $\boldsymbol{p}_{\mathrm{rand}}$ on the ground. We allow the planner to randomly choose between exploration and exploitation, to bias the growth of the tree towards, respectively, unexplored regions and the goal. In the first case, $\boldsymbol{p}_{\mathrm{rand}}$ is generated by randomly choosing its x, y coordinates and retrieving the

[1]To represent the footstep orientation we only use yaw, as roll and pitch are always zero thanks to the piecewise-horizontal ground assumption.

Algorithm 1: Footstep Planner

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

corresponding z coordinate from $\mathcal{M}_z$. In the second case, $\boldsymbol{p}_{\text{rand}}$ is sampled from the goal region $\mathcal{G}$.

Then, the vertex v_{near} of $\mathcal{T}$ that is closest to $\boldsymbol{p}_{\text{rand}}$ is selected for an expansion attempt. The following metric is used for evaluating the distance between a certain vertex v and a point $\boldsymbol{p} \in I\!R^3$:

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

Here, $d(\boldsymbol{m}, \boldsymbol{p})$ is the Euclidean distance of the midpoint $\boldsymbol{m}$ between the feet (at the poses $\boldsymbol{f}_{\text{sup}}$ and $\boldsymbol{f}_{\text{swg}}$ specified in v) from $\boldsymbol{p}$; θ_p is the angle between the robot sagittal axis (defined as the axis passing through $\boldsymbol{m}$ with orientation equal to the average of the orientations of the two footsteps) and the line joining $\boldsymbol{m}$ to $\boldsymbol{p}$; α is a positive scalar.

Once v_{near} has been identified, the foot poses $\boldsymbol{f}_{\text{sup}}^{\text{near}}$ and $\boldsymbol{f}_{\text{swg}}^{\text{near}}$ are extracted from it. A candidate footstep $\boldsymbol{f}^{\text{cand}}$ is generated by randomly selecting a final pose of the swinging foot from the catalogue U of primitives, which is defined with respect to $\boldsymbol{f}_{\text{sup}}^{\text{near}}$ (see Fig. 3). The z coordinate of $\boldsymbol{f}^{\text{cand}}$ is retrieved from the elevation map $\mathcal{M}_z$.

At this point, the candidate footstep is checked for feasibility with respect to requirements R1–R2. If the outcome is positive, the algorithm verifies whether a collision-free trajectory $\boldsymbol{p}_{\text{swg}}^{\text{cand}}$ exists that brings the swinging foot from $\boldsymbol{f}_{\text{swg}}^{\text{near}}$ to $\boldsymbol{f}^{\text{cand}}$ (requirement R3). This is done by using a parametrized trajectory which, once the endpoints are selected, can be deformed[2] by changing the maximum height h along the trajectory. Starting from a lower bound $h_{\min}$, the algorithm iteratively checks increasing values for h up to an upper bound $h_{\max}$. If a collision-free trajectory is found,

[2]As a deformable trajectory we used a polynomial, but other choices (e.g., B-splines and Bezier curves) are also possible.

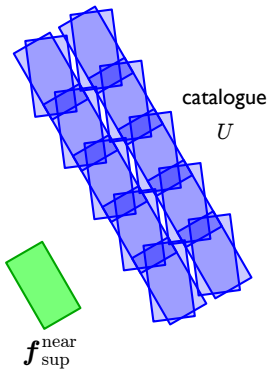

Fig. 3. The catalogue U of primitives specifies the possible planar poses (in blue) of the candidate footstep $\boldsymbol{f}^{\text{cand}}$ with respect to the pose $\boldsymbol{f}_{\text{sup}}^{\text{near}}$ of the current support foot (in green). Here we have considered the case of a swinging right foot; the catalogue for a swinging left foot is specular. Once a primitive is chosen, the z coordinate of the candidate footstep will be retrieved from the elevation map $\mathcal{M}_z$.

a new vertex $v_{\text{new}} = (\boldsymbol{f}^{\text{cand}}, \boldsymbol{f}_{\text{sup}}^{\text{near}})$ is added to the tree, connected to its parent v_{near} by $\boldsymbol{p}_{\text{swg}}^{\text{cand}}$. Note that in v_{new} the roles of the feet have been swapped: in future steps originating from v_{new}, the support foot will stay at $\boldsymbol{f}^{\text{cand}}$ while the swinging foot will move from $\boldsymbol{f}_{\text{sup}}^{\text{near}}$.

Planning terminates when v_{new} completes the walk-to task (i.e., the corresponding midpoint $\boldsymbol{m}$ belongs to $\mathcal{G}$) or a maximum number of iterations has been reached. In the first case, the path joining the root to v_{new} is extracted from the tree, and the footstep sequence $\{\boldsymbol{f}^j\}$ is reconstructed with the associated swing foot trajectories $\{\boldsymbol{p}_{\text{swg}}^j\}$.

IV. Gait Generation via MPC

Once a footstep plan has been computed, it is passed to an on-line gait generation module which must provide a compatible trajectory for the humanoid CoM. This module uses a technique we presented in [13] for generating stable gaits motions on uneven ground. It is based on an MPC scheme which enforces an explicit stability constraint to ensure that the resulting CoM trajectory is bounded w.r.t. the Zero Moment Point (ZMP). In this section, we briefly recall the main points of this technique.

The main feature of the proposed control scheme is to allow those vertical motions of the CoM which still lead to a Linear Time Invariant (LTI) system. In particular, denoting with $\boldsymbol{p}_{\text{c}} = (x_{\text{c}}, y_{\text{c}}, z_{\text{c}})^T$ and $\boldsymbol{p}_{\text{z}} = (x_{\text{z}}, y_{\text{z}}, z_{\text{z}})^T$ the position respectively of the CoM and the ZMP, it has been shown that if the CoM vertical motion satisfies the differential equation

$$\ddot{z}_{\text{c}} = \omega^2(z_{\text{c}} - z_{\text{z}}) - g \tag{1}$$

for some user-defined constant ω, then the x and y CoM motions still verify the typical LIP equations

$$\ddot{x}_{\text{c}} = \omega^2(x_{\text{c}} - x_{\text{z}}) \tag{2}$$

$$\ddot{y}_{\text{c}} = \omega^2(y_{\text{c}} - y_{\text{z}}). \tag{3}$$

Note that ω is a design parameter. Its value determines the equilibrium point of the z_{c} dynamics which is obtained when the difference between the z coordinates of the ZMP

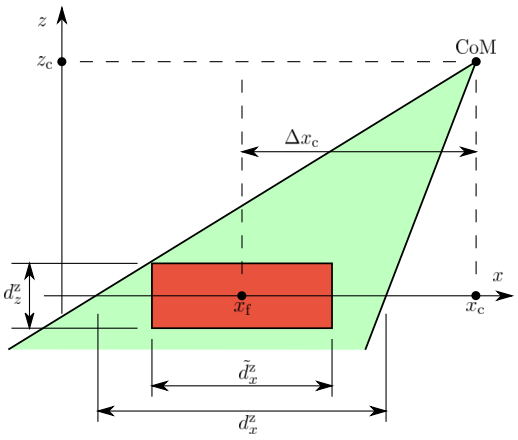

Fig. 4. Side view of the polyhedral cone (green) representing the actual ZMP constraint and the box (red) used as approximate linear constraint.

and CoM is equal to g/ω^2. The considered model neglects angular momentum around the robot CoM.

A. Motion model

In order to achieve a smoother ZMP trajectory, we use as motion model for the MPC algorithm, a third order extension of the LIP model, i.e. we use the derivative of the ZMP $\dot{x}_\mathrm{z}$, $\dot{y}_\mathrm{z}$ and $\dot{z}_\mathrm{z}$ as control inputs. The model along the x-axis is

$$\begin{pmatrix} \dot{x}_\mathrm{c} \\ \ddot{x}_\mathrm{c} \\ \dddot{x}_\mathrm{z} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ \omega^2 & 0 & -\omega^2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_\mathrm{c} \\ \dot{x}_\mathrm{c} \\ x_\mathrm{z} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \dot{x}_\mathrm{z} \qquad (4)$$

The y component is similar while the z component only has the additional additive term in (1).

We use piecewise-constant control inputs over sampling intervals of duration δ, with a prediction horizon $T_\mathrm{h} = N \cdot \delta$. The current time instant is t_k and the successive instants within the prediction horizon are t_{k+i}, $i = 1, \ldots, N$. A similar notation is used for all variables; e.g., the current CoM position is x_c^k and its predicted value at t_{k+i} is x_c^{k+i}.

B. Constraints

On uneven ground, balance is ensured when the ZMP is inside the conic polyhedron having vertex at the CoM and edges passing through the vertices of the support polygon [4]. However, the resulting balance condition would be nonlinear because the polyhedral cone has its vertex at the CoM which changes location. Using the approximation in Fig. 4, the following linear *balance constraint* is obtained

$$-\frac{1}{2}\begin{pmatrix} \tilde{d}_x^\mathrm{z} \\ \tilde{d}_y^\mathrm{z} \\ d_z^\mathrm{z} \end{pmatrix} \leq R_{k+i}^T \begin{pmatrix} x_\mathrm{z}^{k+i} - x_\mathrm{f}^{k+i} \\ y_\mathrm{z}^{k+i} - y_\mathrm{f}^{k+i} \\ z_\mathrm{z}^{k+i} - z_\mathrm{f}^{k+i} \end{pmatrix} \leq \frac{1}{2}\begin{pmatrix} \tilde{d}_x^\mathrm{z} \\ \tilde{d}_y^\mathrm{z} \\ d_z^\mathrm{z} \end{pmatrix}. \qquad (5)$$

Here $\tilde{d}_x^\mathrm{z}$ and $\tilde{d}_y^\mathrm{z}$ are the reduced horizontal sizes of the constraint, chosen to be compatible with the vertical size d_z^z which is a design parameter. R_{k+i}^T is the rotation matrix associated with θ_f^{k+i}. Since x_z^{k+i} can be linearly expressed in terms of the decision variables $\dddot{x}_\mathrm{z}^{k+j}$ for $j = 0, \ldots, i-1$, the constraint is linear in the decision variables. Similarly for the other coordinates y_z^{k+i} and z_z^{k+i}.

To ensure boundedness of the resulting CoM trajectory w.r.t. the ZMP, we enforce the *stability constraint* [12],

$$\boldsymbol{p}_\mathrm{c}^k + \frac{\dot{\boldsymbol{p}}_\mathrm{c}^k}{\omega} + \frac{\boldsymbol{g}}{\omega^2} = \omega \int_{t_k}^{\infty} e^{-\omega(\tau - t_k)} \boldsymbol{p}_\mathrm{z}(\tau) d\tau, \qquad (6)$$

where $\boldsymbol{g} = (0, 0, -g)^T$. For the specific chosen structure of the decision variables, (6) reduces to

$$\frac{1}{\omega}\frac{1 - e^{-\delta\omega}}{1 - e^{-N\delta\omega}} \sum_{i=0}^{N-1} e^{-i\delta\omega} \dddot{x}_\mathrm{z}^{k+i} = x_\mathrm{c}^k + \frac{\dot{x}_\mathrm{c}^k}{\omega} - x_\mathrm{z}^k. \qquad (7)$$

along the x-component (similarly for the y-component). Note that the integral in (6) extends to infinity while we only predict for the horizon T_h. This is handled by *infinitely replicating* the decision variables outside the prediction horizon. For the z-component we assume $\dddot{x}_\mathrm{z}^j = 0$ after the prediction horizon; the stability constraint then becomes

$$\frac{1 - e^{-\delta\omega}}{\omega} \sum_{i=0}^{N-1} e^{-i\delta\omega} \ddot{z}_\mathrm{z}^{k+i} = z_\mathrm{c}^k + \frac{\dot{z}_\mathrm{c}^k}{\omega} - z_\mathrm{z}^k - \frac{g}{\omega^2}. \qquad (8)$$

C. QP and algorithm

The Quadratic Programming (QP) problem is

$$\min_{\dddot{X}_\mathrm{z}^k, \dddot{Y}_\mathrm{z}^k, \dddot{Z}_\mathrm{z}^k} \sum_{i=0}^{N-1} \left((\dddot{x}_\mathrm{z}^{k+i})^2 + (\dddot{y}_\mathrm{z}^{k+i})^2 + (\ddot{z}_\mathrm{z}^{k+i})^2 + \beta (z_\mathrm{z}^{k+i} - z_\mathrm{f}^{k+i})^2 \right)$$

subject to:

ZMP constraints (5)

stability constraints for x, y and z (7) and (8)

Collect the decision variables in vectors

$$\begin{aligned} \dddot{X}_\mathrm{z}^k &= (\dddot{x}_\mathrm{z}^k \ \ldots \ \dddot{x}_\mathrm{z}^{k+N-1})^T \\ \dddot{Y}_\mathrm{z}^k &= (\dddot{y}_\mathrm{z}^k \ \ldots \ \dddot{y}_\mathrm{z}^{k+N-1})^T \\ \dddot{Z}_\mathrm{z}^k &= (\ddot{z}_\mathrm{z}^k \ \ldots \ \ddot{z}_\mathrm{z}^{k+N-1})^T. \end{aligned}$$

The MPC iteration starts at t_k and goes as follows.

1) Compute $\dddot{X}_\mathrm{z}^k, \dddot{Y}_\mathrm{z}^k, \dddot{Z}_\mathrm{z}^k$ that solve the QP problem.
2) Extract the first control samples $\dddot{x}_\mathrm{z}^k$, $\dddot{y}_\mathrm{z}^k$, $\ddot{z}_\mathrm{z}^k$.
3) Set $\dddot{x}_\mathrm{z} = \dddot{x}_\mathrm{z}^k$ in (4) and integrate from $(x_\mathrm{c}^k, \dot{x}_\mathrm{c}^k, x_\mathrm{z}^k)$ to obtain $x_\mathrm{c}(t)$, $\dot{x}_\mathrm{c}(t)$, $\dot{x}_\mathrm{z}(t)$ for $t \in [t_k, t_{k+1}]$. Do the same for the y and z components. This provides the 3D CoM trajectory during the considered time interval.

The MPC algorithm provides the reference CoM trajectory, while a reference trajectory for the swing foot is a direct outcome of the planning phase. Both these trajectories are then tracked on the humanoid robot by kinematic control.

V. Simulations

We performed simulations on the HRP4 humanoid robot in the V-REP environment. We tested our framework in two different scenarios. In both scenarios the robot has to reach a circular goal region of radius 0.5 m. Since the footstep planner is randomized, to evaluate its performance we have performed a set of 10 simulations on each scenario.

All simulations have been performed on an Intel Core i7 running at 2.8 GHz, using the same parameters settings.

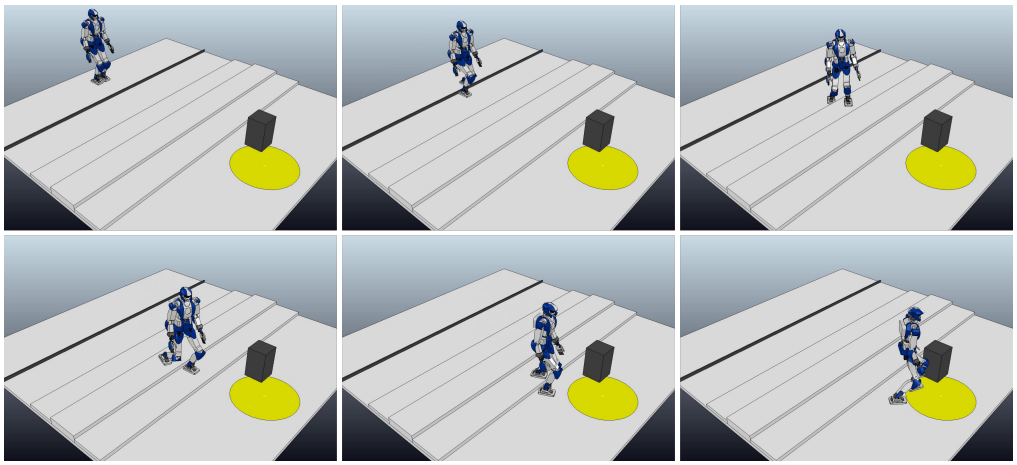

Fig. 5. Simulation 1: the robot reaches the goal going over the black linear obstacle, climbing and descending the staircase, and avoiding the black box.

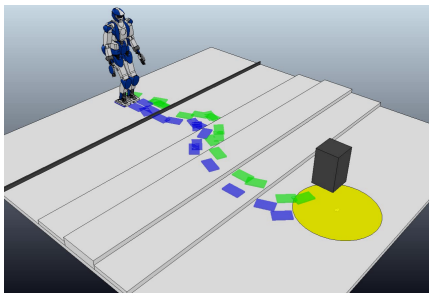

Fig. 6. The footstep plan for Simulation 1.

In particular, in the footstep planning module we have set $\alpha = 1$, $\Delta z_{\max} = 0.08$ m, $h_{\min} = 0.02$, $h_{\max} = 0.12$ and $\Delta h = 0.02$. For the gait generation module, we used $\omega = 3.6$ s$^{-}1$, the step duration is $T_{\mathrm{s}} = 0.8$ s of which 0.5 s of single support and 0.3 s of double support, the size of the constraints is $\tilde{d}_x^{\mathrm{z}} = \tilde{d}_y^{\mathrm{z}} = 0.08$ m and $d_z^{\mathrm{z}} = 0.03$ m. Finally, the elevation map $\mathcal{M}_z$ has a resolution of 0.02 m.

In the first scenario, the robot and the goal are placed at opposite sides of the environment. The goal can be reached by walking a mostly straight path, but footsteps need to be placed appropriately to satisfy all the requirements. Figure 5 shows one of the solutions found by our method, with the underlying footstep sequence in Fig. 6. The robot starts walking forward and reaches the proximity of the bar obstacle, which does not provide a large enough surface to step on. Then, the robot goes over it by taking a longer step, with a swing foot trajectory sufficiently high to avoid collisions. After that, the staircase is ascended and descended. Finally, the robot avoids the black box and reaches the goal region.

performance data	scenario 1	scenario 2
tree size (# vertices)	829.8	2442.8
solution size (# footsteps)	34.2	68.3
planning time (s)	3.69	13.40

TABLE I

SIMULATION PERFORMANCE DATA

The second scenario is traversed by a ditch which can only be entered from the left and exited from the right, because the platform in the middle of it is too low to be accessed directly. One solution produced by our method is shown in Fig. 7, with the underlying footstep sequence in Fig. 8. The robot moves appropriately among the different levels, first going down in the ditch and then up towards the goal.

Video clips of the above simulations are available at the link https://youtu.be/mKVBwlxsXLM.

Table I reports some data obtained by averaging the results of the 10 simulations on each scenario. The table indicates the number of vertexes in the tree, the number of footsteps in the solution and the time needed to compute it.

It is clear that the first scenario is significantly easier than the second to solve. The reason is that in the first scenario there is a straight path to the goal, while in the second the robot must take an S-shaped path which goes through some narrow passages. The descending steps in the second scenario also require the robot to perform the difficult maneuver of rotating in place in a small space. Although, the solution is found in a reasonably short time even in the second scenario.

We do not show any explicit comparison with other existing footstep planners, since the aim of this paper is to introduce an integrated motion planner/controller where an intrinsically stable MPC framework is successfully combined with a simple footstep planning strategy, while optimal and efficient footstep sequence generation will be the focus of further works. Just as example, we report that an A*-based planner, in the spirit of [16], finds shorter footstep sequences but requires more than ten times the average planning time of our approach (51 sec) to solve the first scenario.

An analysis of the video clips reveals that the generated motions are suboptimal in terms of efficiency, particularly in the second simulation. Although there is no optimization stage in our footstep planner, a simple way to improve the quality of the solutions would be to compute several footstep plans rather than a single one, and then choose the best. Another possibility could be to post-process the footstep sequence in order to eliminate unnecessary motions.

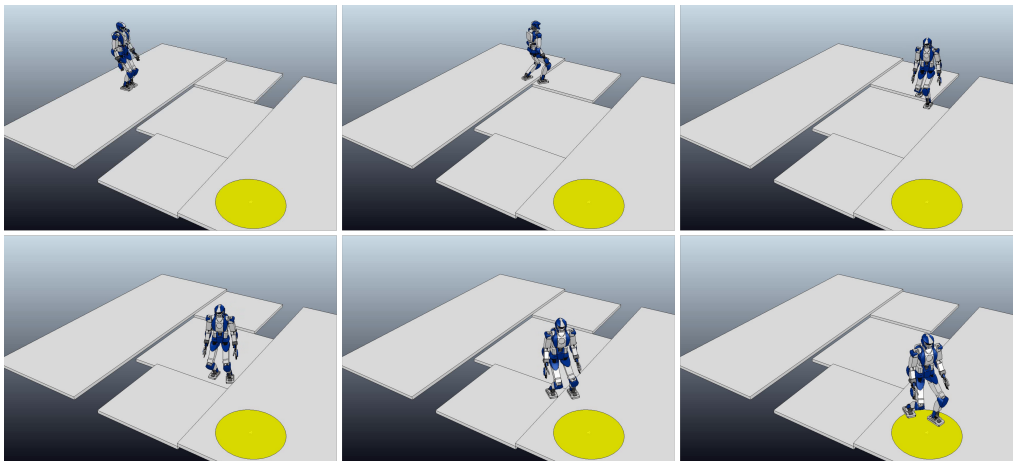

Fig. 7. Simulation 2: the robot reaches the goal going through the ditch, which can only be accessed from the left and exited from the right.

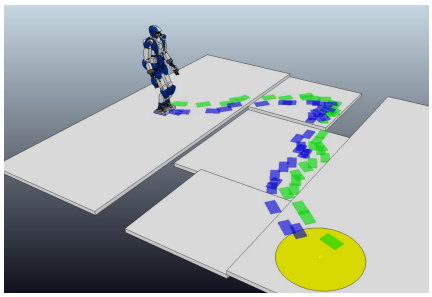

Fig. 8. The footstep plan for Simulation 2.

VI. Conclusions

We presented an architecture integrating planning and control for generating suitable walking motions for a humanoid that has to fulfill a walk-to task on uneven ground. The offline RRT-based footstep planning module has proven to be capable of generating suitable footstep sequences in a short amount of time, which the MPC module utilizes to generate a 3D gait in real-time. The overall architecture, tested by simulations in V-REP with the HRP4 humanoid, revealed to be flexible, as it can embed constraints and can be applied to different environments without needing any pre-processing.

The results are promising and open up to multiple further improvements. These will be addressed in future works, which will include: (i) the introduction of optimality criteria in the planning stage; (ii) the optimization of computation to be able to replan mid-walk if needed; (iii) the implementation of the proposed framework on the real humanoid.

References

[1] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *1991 IEEE Int. Conf. on Robotics and Automation*, 1991, pp. 1405–1411.

[2] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 1620–1626.

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

[4] S. Caron and A. Kheddar, "Dynamic walking over rough terrains by nonlinear predictive control of the floating-base inverted pendulum," in *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017, pp. 5017–5024.

[5] S. Caron, A. Escande, L. Lanari, and B. Mallein, "Capturability-based pattern generation for walking with variable height," *arXiv:1801.07022*, 2018.

[6] A. Herdt, "Model predictive control of a humanoid robot," Ph.D. dissertation, Ecole Nationale Supérieure des Mines de Paris, 2012.

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

[8] T. Kamioka, T. Watabe, M. Kanazawa, H. Kaneko, and T. Yoshike, "Dynamic gait transition between bipedal and quadrupedal locomotion," in *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 2195–2201.

[9] K. Terada and Y. Kuniyoshi, "Online gait planning with dynamical 3d-symmetrization method," in *7th IEEE-RAS Int. Conf. on Humanoid Robots*, 2007, pp. 222–227.

[10] S. C. Luo, P. H. Chang, J. Sheng, S. C. Gu, and C. H. Chen, "Arbitrary biped robot foot gaiting based on variate CoM height," in *13th IEEE-RAS Int. Conf. on Humanoid Robots*, 2013, pp. 534–539.

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

[12] N. Scianca, M. Cognetti, D. De Simone, L. Lanari, and G. Oriolo, "Intrinsically stable MPC for humanoid gait generation," in *16th IEEE-RAS Int. Conf. on Humanoid Robots*, 2016, pp. 601–606.

[13] A. Zamparelli, N. Scianca, L. Lanari, and G. Oriolo, "Humanoid gait generation on uneven ground using intrinsically stable MPC," *IFAC-PapersOnLine*, vol. 51, pp. 393–398, 2018.

[14] W. Burgard, M. Hebert, and M. Bennewitz, "World modeling," in *Springer handbook of robotics*. Springer, 2016, pp. 1135–1152.

[15] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 279–286.

[16] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade, "Footstep planning for the Honda ASIMO humanoid," in *2005 IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 629–634.

[17] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *2012 IEEE-RAS Int. Conf. on Humanoid Robots*, 2012, pp. 674–679.

[18] Z. Xia, G. Chen, J. Xiong, Q. Zhao, and K. Chen, "A random sampling-based approach to goal-directed footstep planning for humanoid robots," in *2009 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, 2009, pp. 168–173.

[19] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 427–439, 2012.