Quadrupedal walking motion and footstep placement through Linear Model Predictive Control

Arturo Laurenzi, Enrico Mingo Hoffman, Nikos Tsagarakis

Quadrupedal walking motion and footstep placement through Linear Model Predictive Control

Arturo Laurenzi[1,2], Enrico Mingo Hoffman[1], and Nikos G. Tsagarakis[1]

***Abstract*— The present work addresses the generation of a walking gait with automatic footstep placement for a quadrupedal robot, within a *Linear Model Predictive Control* framework. Existing work has shown how this is only possible within a non-convex programming framework, finding a solution of which is well-known to be very hard. We propose a way to formulate the joint optimization problem as an approximate QP with linear constraints, whose global optimum can be quickly found with off-the-shelf solvers. More specifically, this is done by introducing auxiliary states and control inputs, each of which is subject to linear constraints that are inspired from the literature on bipedal locomotion.**

Finally, we validate our method on the CENTAURO robot, a hybrid wheeled-legged quadruped with a humanoid upper-body.

I. Introduction

Mobile robots are nowadays expected to have an increasingly important role in many domains, from industrial automation and logistics to maintenance and search and rescue applications leaving the flat factory floors to enter less structured and controlled environment. To operate efficiently in these more challenging environments they should be able to safely move around regardless of how cluttered, or inaccurately known is the terrain. Walking robots have the potential to perform locomotion through arbitrarily complex terrains, at the cost of an increased mechanical *and* control complexity related to stability and body coordination issues. In this work we address the problem of generating an *omni-directional walking gait* for a quadrupedal robot, i.e. a coordinated motion of the robot legs satisfying the property that at least three legs must always be on the ground. When tackling such a problem, it is important to notice that legged robots are *floating base* systems, whose global motion can only be obtained by means of *contact forces* exchanged with the environment. In turn, contact forces must fulfill *physical constraints*, and consequently there exist motions that *cannot* be executed by a floating base robot. Simplified models have been proposed in the literature to describe the set of feasible motions in a way that is more suitable for the development of simple and fast planning algorithms, such as the *linear inverted pendulum model (LIPM)* [1]; this simple model forms the basis of many popular walking controllers.

The generation of a walking gait can be decomposed as the series of a footstep planning stage followed by a center-of-mass (CoM) motion planning. This strategy is common among the earliest approaches to legged locomotion, but it was shown [2] that, for the case of bipeds, it is also possible to *jointly* generate both footsteps *and* CoM motion inside a QP framework, gaining improved robustness and disturbance rejection capabilities [3]. On the contrary, in the case of quadrupedal walking, joint optimization over both CoM motion and footsteps gives rise to *non-linear constraints*, which make the optimization problem more difficult and less efficient to solve.

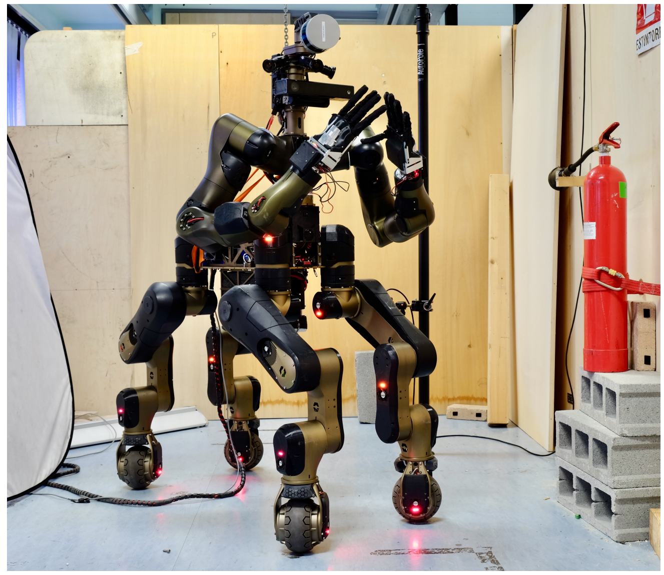

Fig. 1. The CENTAURO robot, a hybrid wheeled-legged quadruped with a humanoid upper-body.

The main contribution of this work is a decomposition of such a joint optimization problem that does not introduce non-linear constraints, by introducing *auxiliary states and control inputs* that are subject to linear constraints. In this way, we formulate an *approximate* optimization problem that can be *exactly* solved. The proposed approach is validated on our new quadrupedal robot *CENTAURO* [4] (see Figure 1), both in simulation and on real hardware.

The remainder of this article is organized as follows:

- in Section II we present a selection of relevant works in the field;
- Section III introduces the mathematical formulation that our algorithm is based on, starting from existing models, and then presenting the decomposition that is the core contribution of the present work;
- Section IV contains the details of our implementation;
- Section V shows real hardware results;

[1]Advanced Robotics Department (ADVR), Istituto Italiano di Tecnologia, Genova, Italy

[2]DIBRIS, Università di Genova, Italy

{arturo.laurenzi, enrico.mingo, nikolaos.tsagarakis}@iit.it

- in Section VI we summarize the outcome of this work and present possible future directions.

II. Related works

Legged locomotion is a rather mature field of research; the solutions that have been proposed can be roughly split into three categories: algorithms that are based on optimal control, *hybrid zero dynamics (HZD)* formulations [5], and algorithms that are based on bio-inspired oscillators [6]. Optimal control methods have demonstrated particularly impressive results, allowing for automatic gait discovery in complex environments [7], [8], albeit with such computational requirements that make them unsuitable for online applications. The rest of this section focuses on relevant works that employ a similar strategy as ours, which is based on optimal control, and with sufficiently low computational burden to allow for online operation.

Such a family of walking gait generators is mainly due to the work of Kajita et al. [1] for bipeds, who tackled the CoM motion generation problem *within feasibility constraints* as a servo tracking problem, where the system dynamics is given as an integrator, and the tracked output is the *Zero Moment Point (ZMP)*, that corresponds to the center of pressure of the ground reaction forces. Inspired by the observation that humans start moving their CoM *before* taking a step, Kajita formulated an LQR problem *with preview* of the future ZMP reference. It was later recognized [9] that to construct a hand-tuned preview on the ZMP trajectory is actually not needed, provided that the ZMP is *constrained* inside future support polygons, which is possible inside a *linear MPC (LMPC)* framework. As final steps in the development of LMPC-based bipedal walking, in [2], [3] the footsteps, CoM motion, and waist orientation [10] were jointly optimized in a single QP problem. Stability analyses of the LMPC *with ZMP constraints* can be found in Wieber [11], [9], showing that, for the limited control horizon case, stability crucially depends on the control horizon time span, with the requirement that some CoM derivative is minimized inside the cost function. The concept of *capture point* was also introduced to identify states that *can be stabilized* without taking a step [12]. Lanari et al. [13], [14] contributed to this topic as well, by relating the capture point idea to a *boundedness constraint*, which describes the initial CoM state and footsteps that permit to generate a bounded CoM motion. Moreover, approaches to bipedal walking have been proposed that control the ZMP according to a feedback law on the capture point [15].

Concerning quadrupedal walking, [16] studied the stability properties of the different *walking gait patterns*, i.e. the order according to which the four legs are lifted, while [17] suggested a simple way of selecting a specific pattern according to the desired speed. In [18][19] heuristic approaches that rely on geometric reasoning to generate *static* walking are proposed. An optimal control approach with preset footholds can be found in [20], while in [21] footsteps are optimized as well inside a *non-linear programming* framework, whose solution is, in general, extremely hard to find.

In this work, we address this specific point by reformulating the joint CoM-footsteps optimization problem as an approximate QP whose solution can be readily found, as explained in the next section.

III. Mathematical formulation

As it was mentioned in Section I, the main difficulty that the walking gait designer must face derives from the fact that legged robots are *underactuated*: global motion cannot be directly achieved by their actuated degrees of freedom; instead, it must be generated by contact forces exchanged with the environment. This intuition is beautifully summarized by the following *centroidal dynamics* equation [1]

$$\begin{aligned} M\ddot{\boldsymbol{p}}_{\text{com}} &= \sum_{i=1}^{N} \boldsymbol{F}_i + M\boldsymbol{g} \\ \dot{\boldsymbol{L}} &= \sum_{i=1}^{N} (\boldsymbol{p}_i - \boldsymbol{p}_{\text{com}}) \times \boldsymbol{F}_i, \end{aligned} \tag{1}$$

where M is the system mass, $\boldsymbol{p}_{\text{com}} \in \mathbb{R}^3$ is the robot CoM position, $\boldsymbol{F}_i \in \mathbb{R}^3$ is the i-th contact force, N is the number of contacts, $\boldsymbol{g} \in \mathbb{R}^3$ is the gravity acceleration, $\boldsymbol{L} \in \mathbb{R}^3$ is the robot angular momentum, and $\boldsymbol{p}_i \in \mathbb{R}^3$ is the i-th contact point. It is remarkably important to notice that these contact forces are *constrained*, and consequently there exist CoM trajectories that cannot be executed by a legged robot. The most important constraint is commonly recognized [11] as the *unilateral constraint*, which takes the following form:

$$\boldsymbol{n}_i^T \boldsymbol{F}_i \geq 0 \quad \forall i \in \{1, \dots, N\} \tag{2}$$

where $\boldsymbol{n}_i \in \mathbb{R}^3$ is the outward normal of the i-th contact surface. Broadly speaking, this means that the robot can only *push* on the ground. Assuming coplanar contacts (and, for simplicity, $\boldsymbol{n} = [0\,0\,1]^T$) and rearranging equations (1) and (2) as in [22], the equivalent centroidal momentum constraint can be obtained as follows:

$$\begin{aligned} &\boldsymbol{z} \in \text{ConvHull}\{\boldsymbol{p}_i\}_{i=1}^{N} \\ &\boldsymbol{z} = \left[\boldsymbol{p}_{\text{com}} - \frac{h}{g+\ddot{h}}\ddot{\boldsymbol{p}}_{\text{com}} + \frac{\boldsymbol{n} \times \dot{\boldsymbol{L}}}{Mg + M\ddot{h}}\right]_{x,y}, \end{aligned} \tag{3}$$

where $\boldsymbol{z} \in \mathbb{R}^2$ is commonly referred to as the *Zero Moment Point (ZMP)*. Neglecting variations in the robot CoM height h and angular momentum, (3) gives rise to the popular *cart-table* model [1]:

$$\begin{aligned} &\boldsymbol{z} \in \text{ConvHull}\{\boldsymbol{p}_i\}_{i=1}^{N} \\ &\boldsymbol{z} = \left[\boldsymbol{p}_{\text{com}} - \frac{\ddot{\boldsymbol{p}}_{\text{com}}}{\omega^2}\right]_{x,y}, \end{aligned} \tag{4}$$

with $\omega = \sqrt{\frac{g}{h}}$ representing a parameter that characterizes the influence of the CoM acceleration on the ZMP position. Notice how, according to such a simplified model, the

[1]Notice that we neglect any *torque* exchanged with the ground, which is equivalent to assuming point contacts.

feasibility of a CoM trajectory only depends on whether a *linear combination* of the CoM derivatives belongs to some *convex* set.

A. Classical approach

If we can assume the set of contact points to be given in advance (e.g. by some footstep planning stage), then we can follow [20], [9] and cast the walking gait generation problem into a linear MPC problem, as it is briefly summarized hereafter.

We first specify our process dynamics as a triple integrator of the CoM jerk, as follows:

$$\dot{\boldsymbol{x}} = A\,\boldsymbol{x} + B\,\boldsymbol{u}, \tag{5}$$

where $\boldsymbol{x} \in \mathbb{R}^6$ is the state vector defined by the aggregation of the planar CoM position, velocity and acceleration, and $\boldsymbol{u} \in \mathbb{R}^2$ is the control input (which corresponds to the CoM jerk). Consequently, $A \in \mathbb{R}^{6\times 6}$ and $B \in \mathbb{R}^{6\times 2}$ take the following form:

$$\begin{aligned} A &= \begin{bmatrix} 0_{2\times2} & I_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} & I_{2\times2} \\ 0_{2\times2} & 0_{2\times2} & 0_{2\times2} \end{bmatrix} \\ B &= \begin{bmatrix} 0_{2\times2} \\ 0_{2\times2} \\ I_{2\times2} \end{bmatrix}. \end{aligned} \tag{6}$$

The ZMP can be defined as an output $\boldsymbol{z} \in \mathbb{R}^2$ of (5):

$$\boldsymbol{z} = C_{\text{zmp}}\boldsymbol{x}; \tag{7}$$

the definition of C_{zmp} follows from (4):

$$C_{\text{zmp}} = \begin{bmatrix} I_{2\times2} & 0_{2\times2} & -\frac{1}{\omega}I_{2\times2} \end{bmatrix}. \tag{8}$$

Finally, we assume piece-wise constant control input over some *control horizon*

$$\boldsymbol{u}(t) = \boldsymbol{u}_k \quad \forall t \in [t_k, t_{k+1}], \quad k \in \{0, \dots, M-1\}, \tag{9}$$

where t_k is the k-th discretization knot, and M denotes the control horizon length (in this work, a fixed discretization step Δt has been used). From standard theory of linear systems we know that the ZMP (as well as any other output) at time t_k depends linearly on both the initial state $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$ and the sequence of controls $\boldsymbol{U} \in \mathbb{R}^{2M}$, as specified below:

$$\boldsymbol{z}_k = \tilde{C}^k_{\text{zmp}}\boldsymbol{x}_0 + \tilde{D}^k_{\text{zmp}}\boldsymbol{U} \tag{10}$$

with $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_0^T & \dots & \boldsymbol{u}_{M-1}^T \end{bmatrix}^T$; the matrices $\tilde{C}^k_{\text{zmp}}$ and $\tilde{D}^k_{\text{zmp}}$ are obtained from integration of (5) over the knots (9).

The ZMP can then be constrained to the convex hull of the contact points over the whole control horizon. Indeed, the feasibility constraint (4) can be written as a *linear* inequality of the following form

$$\Big[(\boldsymbol{p}_{j(i),k} - \boldsymbol{p}_{i,k}) \times (\boldsymbol{z}_k - \boldsymbol{p}_{i,k})\Big]_z \le 0 \tag{11}$$

for each time step k over the control horizon, and for each support polygon side $(i, j(i))$, where $j(i)$ denotes the subsequent of the i-th foot, according to a clockwise ordering[2].

[2] As it is customary in the literature, we assign integer labels to the four legs according to a clock-wise ordering and starting from the front-left leg.

The resulting optimization problem takes the form

$$\begin{aligned} \min_{\boldsymbol{U}} \; & \tfrac{1}{2} \sum_{k=1}^{M} \boldsymbol{x}_k^T\, Q_k\, \boldsymbol{x}_k + \boldsymbol{u}_k^T\, R_k\, \boldsymbol{u}_k \\ \text{s.t.} \quad & A_{\text{zmp}}(\boldsymbol{P})\boldsymbol{U} \le \boldsymbol{b}_{\text{zmp}}(\boldsymbol{P}, \boldsymbol{x}_0), \end{aligned} \tag{12}$$

where A_{zmp} and $\boldsymbol{b}_{\text{zmp}}$ account for (11) when evaluated over all support polygons sides and over the control horizon as well. Such matrices depend on the current *and* future footsteps, which are collected in the vector $\boldsymbol{P} \in \mathbb{R}^{2\cdot(1+M_P)\cdot 4}$, with M_P representing the number of *predicted footsteps*.

Notice that, if we do not optimize over the footsteps $\boldsymbol{p}_i$, the constraint (11) is linear; on the contrary, if we want to include the footsteps inside the optimization process, non-linearities arise in the form of *quadratic constraints*. Moreover, such a constraint becomes non-convex (see the appendix for a simple proof), resulting in an *NP-hard* problem. Even though several algorithms exist that allow to find a (local) minimizer of such a problem, it is the authors' belief that finding a linearly constrained QP approximation of the full problem would be beneficial for at least two reasons:

- QPs are a standard class of optimization problems that are well-known in the scientific community; global minimizers can be quickly computed by means of off-the-shelf solvers (e.g. [23]). General-purpose NLP solvers, on the other hand, can be expected to be significantly slower.
- NLP solvers can only provide local minima of non-convex problems. It can be argued that the risk of converging to a "bad" local minimum may ruin the planner performance.

The remainder of this section is devoted to the development of such a QP approximation, that is the main contribution of the present work.

B. Proposed decomposition

As it was mentioned in the previous subsection, our goal is to derive a QP approximation of problem (12) when optimizing for both ZMP and footsteps. More specifically, the approximated feasible set should be a *linear subset* of the complete set (11), so that a solution to the approximated QP will also be a feasible point for the original problem.

To this aim, we observe that the nonlinearity in (11) originates from the coupling between stance feet *pairs*. Indeed, also in the case of bipedal walking, the authors of [3] noticed how nonlinearities arise whenever more than one stance foot is considered. With this in mind, we propose to split the set of the feet indices $I = \{1, 2, 3, 4\}$ into two partitions of two indices each, I_A and I_B. Correspondingly, we introduce two auxiliary states $\boldsymbol{x}_A \in \mathbb{R}^6$ and $\boldsymbol{x}_B \in \mathbb{R}^6$, such that the full robot state $\boldsymbol{x}$ is given by a convex combination of the two auxiliary states:

$$\boldsymbol{x} = \alpha\,\boldsymbol{x}_A + (1-\alpha)\,\boldsymbol{x}_B \tag{13}$$

for some parameter $\alpha \in (0\ 1)$, that we call *distribution factor*. In addition, we also define auxiliary control inputs $\boldsymbol{u}_A \in \mathbb{R}^2$

and $\boldsymbol{u}_B \in \mathbb{R}^2$ such that an analogous relation as (13) holds for the same value of α:

$$\boldsymbol{u} = \alpha\, \boldsymbol{u}_A + (1-\alpha)\, \boldsymbol{u}_B. \tag{14}$$

Following these definitions, we can define an auxiliary system whose state $\tilde{\boldsymbol{x}} \in \mathbb{R}^{12}$ and input $\tilde{\boldsymbol{u}} \in \mathbb{R}^4$ are given by the concatenation of the two auxiliary states and inputs:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{x}_B \end{bmatrix}, \qquad \tilde{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_A \\ \boldsymbol{u}_B \end{bmatrix}. \tag{15}$$

Clearly, the auxiliary dynamics

$$\dot{\tilde{\boldsymbol{x}}} = \tilde{A}\,\tilde{\boldsymbol{x}} + \tilde{B}\,\tilde{\boldsymbol{u}} \tag{16}$$

is described by the following matrices:

$$\tilde{A} = \begin{bmatrix} A & 0_{6\times 6} \\ 0_{6\times 6} & A \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B \\ B \end{bmatrix}. \tag{17}$$

The robot state $\boldsymbol{x}$ can then be recovered as an output for system (16), as it is shown below:

$$\boldsymbol{x} = C_{\text{state}}\,\tilde{\boldsymbol{x}} \tag{18}$$

$$C_{\text{state}} = \begin{bmatrix} \alpha\, I_{6\times 6} & (1-\alpha)\, I_{6\times 6} \end{bmatrix}. \tag{19}$$

Likewise, we can define outputs corresponding to the *auxiliary ZMPs* $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$ by considering (4) for the auxiliary states $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$, respectively.

C. Feasibility constraint

To generate linear constraints, we notice that the two auxiliary states, together with the corresponding footsteps, define two *equivalent bipeds*. Drawing from [3], we can define biped-like feasibility constraints for both auxiliary systems, enforcing the two auxiliary ZMPs to lie inside the corresponding biped supports. Finally, we notice that the full quadruped support is given by the convex hull of the two equivalent bipeds supports, according to the following expression:

$$\boldsymbol{z} = \alpha\, \boldsymbol{z}_A + (1-\alpha)\, \boldsymbol{z}_B; \tag{20}$$

consequently, as the global ZMP is given by a convex combination of the auxiliary ZMPs, it will lie inside the full polygon. An illustration of this is given by Figure 2.

To obtain a numerically stable QP, we set the equivalent bipeds feet size to a small (but not zero) $\boldsymbol{\delta p} \in \mathbb{R}^2$.

D. Auxiliary state initialization

It is worth noticing that, having introduced new auxiliary states in our dynamics, we do not have an *observable* system anymore; broadly speaking, this means that the full state (15) cannot be reconstructed from the measured output, which we assume to be the robot state $\boldsymbol{x}$ defined by (18). As a consequence, it is impossible to compute (or estimate) in a meaningful way the initial value of the auxiliary state $\tilde{\boldsymbol{x}}$, which is needed at each control time by the MPC algorithm. However, since auxiliary sub-states do not carry any physical meaning, we are free to choose the corresponding value arbitrarily, as long as the following equality holds true:

$$\boldsymbol{x}_0 = C_{\text{state}}\,\tilde{\boldsymbol{x}}_0, \tag{21}$$

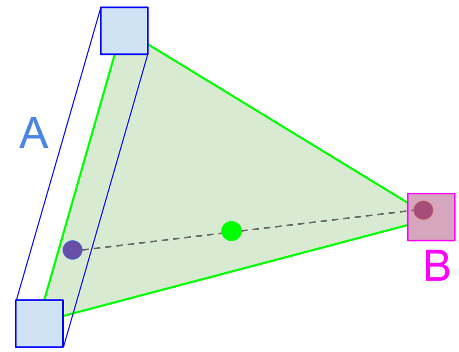

Fig. 2. Decomposed feasibility constraint as described in Section III-C. Auxiliary ZMPs are shown as colored circles for both system A (blue) and B (purple). The resulting global ZMP (green) is inside the support polygon.

i.e. the initial robot state matches the measured one. Finally, we notice how the initial auxiliary state appears *linearly* in both the cost function and the constraints of the LMPC problem; hence, we can let the solver determine an optimal value for $\tilde{\boldsymbol{x}}_0$ by introducing it as decision variable, and enforcing (21) as a constraint.

E. Parameters choice

To implement our decomposition, we first need to choose a partitioning I_A, I_B. To this aim, we notice that the quality of velocity tracking along different directions will differ, depending on the specific choice. More specifically, a front-back partitioning ($I_A = \{1,2\}$, $I_B = \{3,4\}$) will privilege forward walking, while a left-right partitioning ($I_A = \{1,4\}$, $I_B = \{2,3\}$) will favour lateral walking. This is explained as follows: in the first scenario, the supports of the two systems have the possibility to overlap along the forward direction, whereas they are always disjointed along the lateral direction. Consequently, the ZMP trajectory can be continuous along the forward axis, while it is always discontinuous along the vertical axis, causing oscillations that are well known in the literature. For a left-right partitioning, the vice-versa happens instead.

Concerning the role of the distribution factor α, it intuitively controls how much of the robot weight is supported by the auxiliary systems A and B, i.e. their relative load distribution.

Throughout the rest of this work we employ a front-back partitioning, while the distribution factor is fixed at $\alpha = \frac{1}{2}$. A more detailed discussion on the role of the distribution factor is left for future work.

IV. Implementation details

The proposed algorithm was implemented in C++ inside the *OpenSoT* framework [24], that mainly targets hierarchical QP optimization problems with constraints, decoupling the concepts of *front-end*, i.e. the interface that allows to formulate the optimization problem, from the *back-end*, i.e. the tool that is actually used to solve it. More specifically, the front end allows to combine *tasks* and *constraints* in a natural way by overloading suitable operators. The back-end implementation that was used in this work was powered by the *qpOASES* [23] solver. The following tasks and constraints were implemented:

- tracking of a CoM velocity reference $\boldsymbol{v}_{\text{ref}}$:

$$J_{\text{vel}}(\boldsymbol{U}, \tilde{\boldsymbol{x}}_0) = \sum_{k=1}^{M} \left\| \dot{\boldsymbol{p}}_{\text{com},k} - \boldsymbol{v}_{\text{ref}} \right\|^2; \tag{22}$$

- a footstep regularization task, which tries to bias the feet positions to the center of the respective workspaces $\bar{\boldsymbol{p}}_j$, for each foot j belonging to the set of stance feet at time k, denoted by S_k:

$$J_{\text{footstep}}(\boldsymbol{U}, \boldsymbol{P}, \tilde{\boldsymbol{x}}_0) = \sum_{k=1}^{M} \sum_{j \in S_k} \left\| \boldsymbol{p}_{j,k} - \boldsymbol{p}_{\text{com},k} - \bar{\boldsymbol{p}}_j \right\|^2; \tag{23}$$

- minimum CoM acceleration and jerk tasks, as follows:

$$\begin{aligned} J_{\text{acc}}(\boldsymbol{U}, \tilde{\boldsymbol{x}}_0) &= \sum_{k=1}^{M} \left\| \ddot{\boldsymbol{p}}_{\text{com},k} \right\|^2 \\ J_{\text{jerk}}(\boldsymbol{U}, \tilde{\boldsymbol{x}}_0) &= \sum_{k=1}^{M} \left\| \boldsymbol{u}_k \right\|^2; \end{aligned} \tag{24}$$

- feasibility constraint (for the single auxiliary states), as described in Section III-C;
- footspan constraint, whose aim is to ensure that the relative position of the feet lies between some lower and upper bound:

$$\Delta p_{\min}^{i,j} \leq \boldsymbol{p}_{i,k} - \boldsymbol{p}_{j(i),k} \leq \Delta p_{\max}^{i,j} \quad \forall i \in S_k; \tag{25}$$

  in (25) $j(i) \in S_k$ denotes the index of the leg adjacent to leg i, according to a clockwise ordering.
- Initial state consistency constraint (21).

The final objective function was obtained as a weighted sum of the atomic tasks that were listed above.

V. Experimental results

In order to validate the proposed approach, we test it on our CENTAURO robot, a hybrid wheeled-legged quadruped with a humanoid upper body, which is powered by our control framework *XBotCore* [25]. XBotCore allows us to control the robot under hard real-time (RT) constraints, while offering at the same time a complete interface to non-RT (NRT) external processes. We command a piece-wise constant velocity reference for the CoM, both in the forward and lateral direction. The gait pattern is dynamically computed as a function of the velocity reference according to [17], in order to maximize the static stability margin, using a fixed stride time T and duty cycle β. We tune the parameters as in Table I, trying to balance tracking performance while avoiding excessive stretching of the legs. The resulting optimization problem has $n_V = 108$ decision variables and $n_C = 208$ constraints, which leads to roughly 50 Hz average execution frequency (see Fig. 4), which is more than three times faster when compared to [21].

However, it should be noted that our implementation does not take advantage of the sparsity pattern, as it does not exploit the fact that the hessian of the objective function is actually constant and does not need to be recomputed and re-factorized at each iteration. As a remark, notice that, by choosing $\beta < 0.75$, our algorithm can also generate trotting motions.

To transfer the planned motion to the robot, we adopt a simple inverse kinematics (IK) scheme. Once again, we leverage the OpenSoT framework to write a hierarchical IK problem with the following priorities:

1) CoM task + Feet position task
2) Knee task + Waist orientation task + Postural task,

where the aim of the *knee task* is to avoid the collision of the robot knees. Moreover, joint position and velocity limits are enforced as constraints. We assign a low weight to the waist orientation task, so that natural rotations arise from the minimization of joint velocities given by the postural task. From the software architecture point of view, the IK runs inside the RT loop at 1 kHz frequency, while the motion planning runs on a NRT ROS node.

Figure 5 and 3 show the achievable tracking performance. It can be noticed that, as discussed in Section III-E, the forward velocity is tracked smoothly and precisely; on the contrary, lateral velocity is tracked only on average, and with greater steady-state error. Indeed, this behavior is inherited from the approach of [3], that the present work aims to extend to the quadrupedal case. Finally, it can be visually checked from Figure 6 that the proposed method does indeed

TABLE I

PARAMETERS USED FOR THE EXPERIMENT.

Parameter	Value	Parameter	Value
M	20	w_{acc}	1
M_P	2	$w_{\text{vel},\,x}$	100
Δt	0.05 s	$w_{\text{vel},\,y}$	1000
$\delta p_{x,y}$	0.05 m	$w_{\text{footsteps}}$	1000
$\Delta p_{\min}^{1,2}$, $\Delta p_{\min}^{4,3}$	$[-0.3, 0.3]$ m	$\Delta p_{\min}^{2,3}$, $\Delta p_{\min}^{1,4}$	$[0.6, -0.2]$ m
$\Delta p_{\max}^{1,2}$, $\Delta p_{\max}^{4,3}$	$[0.3, 0.7]$ m	$\Delta p_{\max}^{2,3}$, $\Delta p_{\max}^{1,4}$	$[1.2, 0.2]$ m
T	3.0 s	β	0.8

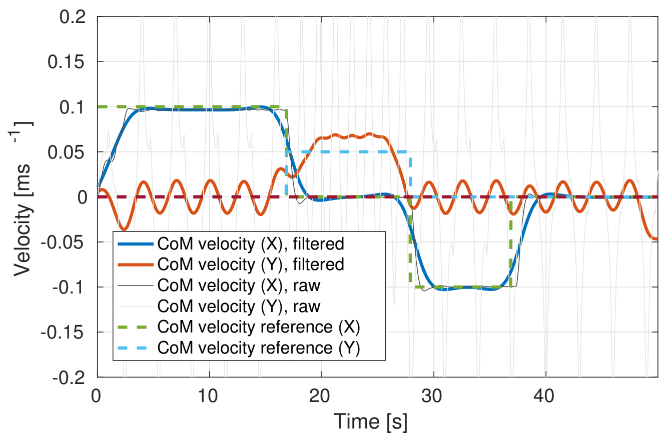

Fig. 3. Planned CoM velocity profile (solid) against reference (dash). Data were processed through zero-phase low-pass filtering with cutoff frequency $f_c = 0.2$ Hz. Raw data are represented in grey.

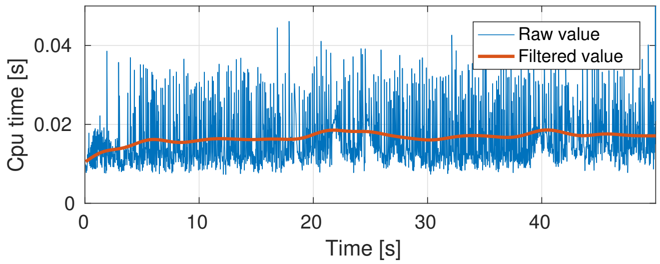

Fig. 4. CPU time needed to fully set up and solve with a naive implementation the MPC QP problem on an Intel i7-6700@3400Hz CPU.

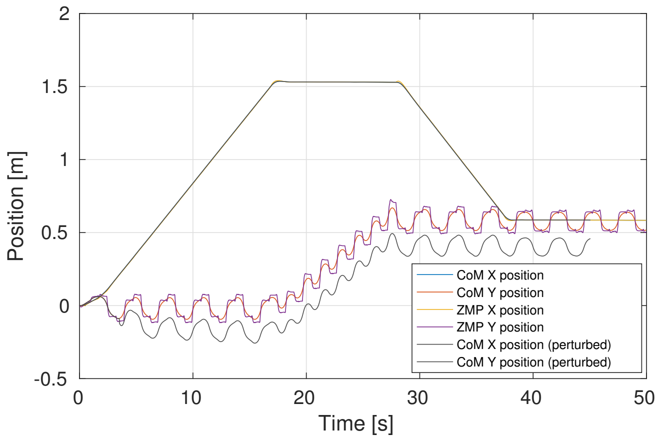

Fig. 5. Planned CoM and ZMP trajectories without any external disturbance (colored lines), and with an external impulsive force $F = 60$ N applied in the negative y direction for 0.3 s (grey lines). For reference, the robot mass is roughly 90 kg.

generate a ZMP which is always *inside* the support polygon, therefore resulting in a feasible motion.

We also tested the disturbance rejection capabilities of our method, by *simulating* an external impulsive force that is applied while the robot is walking. As it can be seen in Figure 5 (grey lines), the CoM plan deviates in the same direction of the force, in order to absorb the impact, while at the same time adapting the footsteps as well.

The outcome of our experiment is summarized in Figure 7, and in the accompanying video as well.

VI. Conclusions

The present work introduced a way to generate a walking motion of a quadrupedal robot, through the joint optimization of both the CoM trajectory *and* the footsteps as well. The proposed method is more robust than fixing the footsteps a-priori since, as discussed in [3], enforcing the ZMP to always lie inside the pre-planned supports could require excessive CoM motions (or be unfeasible altogether). Besides, differently from the work of [21] which allows to find *local* minimizers of a non-convex optimization problem, we propose to find the exact global minimizer of an *approximated* QP problem. Such an approximation leads to the loss of some feasible solutions, and to an increased number of decision variables, which indeed represent drawbacks of our formulation; on the other hand, we eliminate the risk

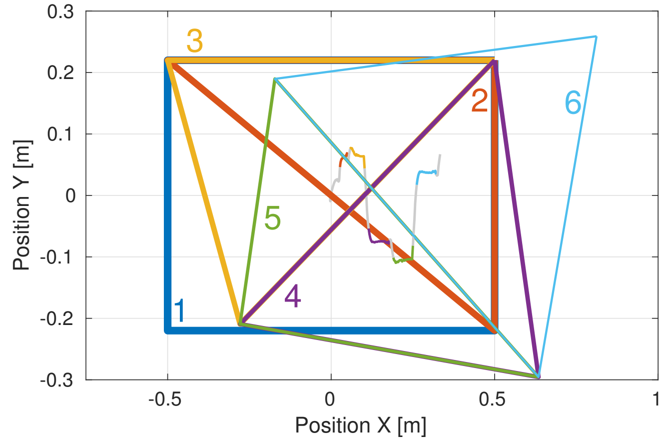

Fig. 6. Sequence of support polygons generated by the proposed algorithm. The ZMP trajectory is plotted as well, with a color that matches the corresponding polygon (grey corresponds to four-stance phases).

Fig. 7. Snapshots taken from an experiment on the actual CENTAURO robot. For the sake of clarity, the final backward phase is not included.

of computing a bad local minimum for a the original nonlinear program. In addition, we achieve a significantly faster computation time when compared to [21] even with a naive, dense implementation.

Our first trials on the quadruped robot CENTAURO have shown promising results. A mixed forward-lateral-backward gait was transferred to the actual hardware with little parameter tuning.

Future work will address the integration of in place rotations, that are possible under this framework provided that the orientation trajectory is fixed beforehand, as in [3]. Moreover, the role of the *load distribution* α, introduced in Section III, must be further investigated. Such a parameter may also be considered to be time-varying, in order to produce more variegate CoM motions, and recover part of the lost solutions.

Lastly, more optimized solvers can be implemented by avoiding to uselessly recompute and re-factorize the hessian matrix, and also by carrying out an analysis of consecutive MPCs *active sets*.

Appendix

A. Non-convexity of the feasibility constraint

We can equivalently formulate the feasibility constraint (11) as follows:

$$\left[\Delta\boldsymbol{p}_i \times \Delta\boldsymbol{p}_j\right]_z \leq 0, \tag{26}$$

where $\Delta\boldsymbol{p}_i = \boldsymbol{p}_i - \boldsymbol{z}$ and $\Delta\boldsymbol{p}_j = \boldsymbol{p}_j - \boldsymbol{z}$, and both such quantities depend linearly on the optimization variables. Noticing that such a constraint is *quadratic*, we may write it in the following form:

$$\begin{bmatrix}\Delta\boldsymbol{p}_i^T & \Delta\boldsymbol{p}_j^T\end{bmatrix} Q \begin{bmatrix}\Delta\boldsymbol{p}_i \\ \Delta\boldsymbol{p}_j\end{bmatrix} \leq 0, \tag{27}$$

where the matrix $Q \in \mathbb{R}^{4\times 4}$ is given by

$$Q = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}. \tag{28}$$

Finally, since a constraint of the form $h(x) \leq 0$ is convex if and only if h is a convex function, we just need to check whether $Q = Q^T$ is *positive semi-definite*. Indeed, this is not the case, since some of its eigenvalues are negative:

$$\det(\lambda I - Q) = \left(\lambda^2 - 1\right)^2 \rightarrow \mathrm{sp}(Q) = \{-1, -1, +1, +1\}. \tag{29}$$

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No. 644839 (CENTAURO) and No. 644727 (CogIMon). The authors want to thank Luca Muratore for the support in the experimental sessions.

References

[1] 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 *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1620–1626, IEEE, 2003.

[2] H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1121–1126, IEEE, 2008.

[3] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[4] L. Baccelliere *et al.*, "Development of a human size and strength compliant bi-manual platform for realistic heavy manipulation tasks," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 5594–5601, IEEE, 2017.

[5] K. Sreenath, H.-W. Park, I. Poulakakis, and J. W. Grizzle, "A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1170–1193, 2011.

[6] Q. Wu, C. Liu, J. Zhang, and Q. Chen, "Survey of locomotion control of legged robots inspired by biological concept," *Science in China Series F: Information Sciences*, vol. 52, no. 10, pp. 1715–1729, 2009.

[7] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, 2018.

[8] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, 2017.

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

[10] A. Herdt, N. Perrin, and P.-B. Wieber, "Walking without thinking about it," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 190–195, IEEE, 2010.

[11] P.-B. Wieber, "Viability and predictive control for safe locomotion," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1103–1108, IEEE, 2008.

[12] T. Koolen, T. de Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[13] L. Lanari, S. Hutchinson, and L. Marchionni, "Boundedness issues in planning of locomotion trajectories for biped robots," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pp. 951–958, IEEE, 2014.

[14] N. Scianca, M. Cognetti, D. De Simone, L. Lanari, and G. Oriolo, "Intrinsically stable mpc for humanoid gait generation," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pp. 601–606, IEEE, 2016.

[15] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4420–4427, Sept 2011.

[16] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, pp. 331–351, 1968.

[17] S. Ma, T. Tomiyama, and H. Wada, "Omnidirectional static walking of a quadruped robot," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 152–161, 2005.

[18] D. Pongas, M. Mistry, and S. Schaal, "A robust quadruped walking gait for traversing rough terrain," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1474–1479, IEEE, 2007.

[19] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 814–820, IEEE, 2009.

[20] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5148–5154, 2015.

[21] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and foothold optimization for quadruped locomotion," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 5308–5313, IEEE, 2017.

[22] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to humanoid robotics*, vol. 101. Springer, 2014.

[23] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "Qpoases: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, pp. 327–363, Dec 2014.

[24] E. M. Hoffman, A. Rocchi, A. Laurenzi, and N. G. Tsagarakis, "Robot control for dummies: Insights and examples using opensot," in *17th IEEE-RAS International Conference on Humanoid Robotics, 2017*, pp. 736–741, 2017.

[25] L. Muratore, A. Laurenzi, E. M. Hoffman, A. Rocchi, D. G. Caldwell, and N. G. Tsagarakis, "Xbotcore: A real-time cross-robot software platform," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, pp. 77–80, April 2017.