The EU Framework Programme for Research and Innovation H2020
Research and Innovation Action

CENTAURO

Deliverable D2.6 Legged Locomotion Control Strategies

Dissemination Level: Public

Project acronym:	CENTAURO
Project full title:	Robust Mobility and Dexterous Manipulation in Disaster Response by Fullbody Telepresence in a Centaur-like Robot
Grant agreement no.:	644839
Lead beneficiary:	IIT – Fondazione Istituto Italiano di Tecnologia
Authors:	Arturo Laurenzi, Nikos Tsagarakis
Work package:	WP2 – Robot Platform
Date of preparation:	2018-07-20
Type:	Report
Version number:	1.0

Document History

Version	Date	Author	Description
0.1	2018-07-20	Arturo Laurenzi, Nikos Tsagarakis	Initial version
0.2	2018-08-20	Arturo Laurenzi, Nikos Tsagarakis	Incorporated suggestions from internal reviews
1.0			Submitted version

Executive Summary

This report is associated with D2.6 and presents developments within WP2 and in particular those of Task 2.5 which focus on the legged locomotion control of the CENTAURO platform. The report introduces the principles and approaches adopted to generate and modulate the gait pattern of the quadruped robot, discusses their implementation on the actual platform and presents experimental results from trials on the robot demonstrating the effectiveness of the developed methods. More specifically, simultaneous center-of-mass and footstep optimization is carried out inside a novel QP framework, which avoids bad local minima that are possible under non-linear optimization frameworks.

Contents

1 Introduction

The workpackage WP2 focuses on the development and experimentation of the CENTAURO robot platform. To this end, the main targets of the workpackage are the design and fabrication of the platform, as well as the development of walking and balancing strategies using both the wheels and legs. Finally, the developed strategies will be experimentally validated on the hardware.

In particular within WP2, Task T2.5 addresses the legged stepping motion control of CENTAURO. This is achieved in two parts: first, the development of algorithms that drive each foot of the robot to an appropriate Cartesian pose, which is defined at a higher level control stage; second, the development of legged gait pattern generators, which guide the robot center-of-mass (CoM) and feet along suitable trajectories that define a stable walking of the robot.

This report describes the development of 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)* [6]; 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 [1] 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.

In order to retain the advantages of both a joint optimization strategy *and* a QP formulation, a novel decomposition is proposed 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. It is also worth noticing that, since footholds are *decision variables* inside the proposed QP, they can be given some *target values* from higher level planning stages.

Finally, the proposed approach is validated on the Centauro robot.

2 Stability of Legged Robots

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 [10] 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 [5], 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 [6]:

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

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

As a concluding remark, if condition (4) is satisfied together with its characterizing assumptions, the robot feet will not lose contact with the ground; with the further assumptions that the joint CoM-foot motion does not violate the robot kinematic constraints, this means that the planned walking motion can be executed successfully on the real platform.

3 Walking Pattern Generation

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 [12], [9] and cast the walking gait generation problem into a linear MPC problem, as it is briefly summarized hereafter.

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

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:

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

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):

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

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

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

where t_k is the k-th discretization knot, and M denotes the control horizon length (a fixed discretization step Δt has been used hereafter). 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

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_0 \\ \vdots \\ \boldsymbol{u}_{M-1} \end{bmatrix}; \tag{11}$$

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

$$\left[(\boldsymbol{p}_{j(i),k} - \boldsymbol{p}_{i,k}) \times (\boldsymbol{z}_k - \boldsymbol{p}_{i,k})\right]_z \leq 0 \tag{12}$$

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]. The resulting

[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.

optimization problem takes the form

$$\begin{aligned} \min_{\boldsymbol{U}} \; & \tfrac{1}{2} \sum_{k=1}^{M} \boldsymbol{x}_k^T \, Q_k \, \boldsymbol{x}_k + \boldsymbol{u}_k^T \, R_k \, \boldsymbol{u}_k \\ \text{s.t.} \quad & A_{\text{zmp}}(\boldsymbol{P}) \boldsymbol{U} \leq \boldsymbol{b}_{\text{zmp}}(\boldsymbol{P}, \boldsymbol{x}_0), \end{aligned} \tag{13}$$

where A_{zmp} and $\boldsymbol{b}_{\text{zmp}}$ account for (12) 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 (12) 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. [2]). 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.

3.1 Proposed Decomposition

As it was mentioned in the previous subsection, our goal is to derive a QP approximation of problem (13) when optimizing for both ZMP and footsteps. More specifically, the approximated feasible set should be a *linear subset* of the complete set (12), 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 (12) 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:

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

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 (14) holds for the same value of α:

$$\boldsymbol{u} = \alpha \, \boldsymbol{u}_A + (1 - \alpha) \, \boldsymbol{u}_B. \tag{15}$$

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:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{x}_B \end{bmatrix}, \qquad \tilde{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_A \\ \boldsymbol{u}_B \end{bmatrix}. \tag{16}$$

Clearly, the auxiliary dynamics

$$\dot{\tilde{\boldsymbol{x}}} = \tilde{A}\,\tilde{\boldsymbol{x}} + \tilde{B}\,\tilde{\boldsymbol{u}} \tag{17}$$

is described by the following matrices:

$$\tilde{A} = \begin{bmatrix} A & 0_{6\times6} \\ 0_{6\times6} & A \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B \\ B \end{bmatrix}. \tag{18}$$

The robot state $\boldsymbol{x}$ can then be recovered as an output for system (17), as it is shown below:

$$\boldsymbol{x} = C_{\text{state}}\,\tilde{\boldsymbol{x}} \tag{19}$$

$$C_{\text{state}} = \begin{bmatrix} \alpha\, I_{6\times6} & (1-\alpha)\, I_{6\times6} \end{bmatrix}. \tag{20}$$

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.

3.2 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:

$$\boldsymbol{z} = \alpha\,\boldsymbol{z}_A + (1-\alpha)\,\boldsymbol{z}_B; \tag{21}$$

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 1.

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

3.3 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 (16) cannot be reconstructed from the measured output, which we assume to be the robot state $\boldsymbol{x}$ defined by (19). 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:

$$\boldsymbol{x}_0 = C_{\text{state}}\,\tilde{\boldsymbol{x}}_0, \tag{22}$$

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 (22) as a constraint.

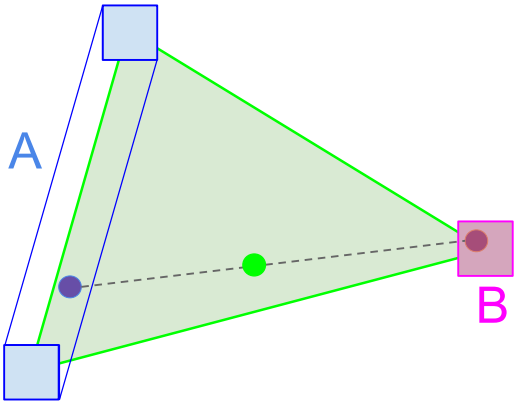

Figure 1: Decomposed feasibility constraint as described in Section 3.2. Auxiliary ZMPs are shown as colored circles for both system A (blue) and B (purple). The resulting global ZMP (green) is inside the support polygon.

3.4 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 favor 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 report we employ a front-back partitioning, while the distribution factor is fixed at $\alpha = \frac{1}{2}$.

4 Implementation Details

The proposed algorithm was implemented in C++ inside the *OpenSoT* framework [4], 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 our work was powered by the *qpOASES* [2] solver. The following tasks and constraints were implemented:

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

$$J_{\text{vel}}(\boldsymbol{U}, \tilde{\boldsymbol{x}}_0) = \sum_{k=1}^{M} \left\| \dot{\boldsymbol{p}}_{\text{com},k} - \boldsymbol{v}_{\text{ref}} \right\|^2 ; \tag{23}$$

- 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:

$$J_{\text{footstep}}(\boldsymbol{U}, \boldsymbol{P}, \tilde{\boldsymbol{x}}_0) = \sum_{k=1}^{M} \sum_{j \in S_k} \left\| \boldsymbol{p}_{j,k} - \boldsymbol{p}_{\text{com},k} - \bar{\boldsymbol{p}}_j \right\|^2; \tag{24}$$

- minimum CoM acceleration and jerk tasks, as follows:

$$\begin{aligned} J_{\text{acc}}(\boldsymbol{U}, \tilde{\boldsymbol{x}}_0) &= \sum_{k=1}^{M} \left\| \ddot{\boldsymbol{p}}_{\text{com},k} \right\|^2 \\ J_{\text{jerk}}(\boldsymbol{U}, \tilde{\boldsymbol{x}}_0) &= \sum_{k=1}^{M} \left\| \boldsymbol{u}_k \right\|^2; \end{aligned} \tag{25}$$

- feasibility constraint (for the single auxiliary states), as described in Section 3.2;

- footspan constraint, whose aim is to ensure that the relative position of the feet lies between some lower and upper bound:

$$\Delta p_{\text{min}}^{i,j} \leq \boldsymbol{p}_{i,k} - \boldsymbol{p}_{j(i),k} \leq \Delta p_{\text{max}}^{i,j} \quad \forall i \in S_k; \tag{26}$$

  in (26) $j(i) \in S_k$ denotes the index of the leg adjacent to leg i, according to a clockwise ordering.

- Initial state consistency constraint (22).

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

5 Experimental Results

We test our walking pattern generator on the Centauro robot, which is powered by our control framework *XBotCore* [8]. 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 [7], 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 1, 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. 3), which is more than three times faster when compared to [11].

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.

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:

Table 1: 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
$\Delta p^{1,2}_{\text{min}}$, $\Delta p^{4,3}_{\text{min}}$	$[-0.3,\ 0.3]$ m	$\Delta p^{2,3}_{\text{min}}$, $\Delta p^{1,4}_{\text{min}}$	$[0.6,\ -0.2]$ m
$\Delta p^{1,2}_{\text{max}}$, $\Delta p^{4,3}_{\text{max}}$	$[0.3,\ 0.7]$ m	$\Delta p^{2,3}_{\text{max}}$, $\Delta p^{1,4}_{\text{max}}$	$[1.2,\ 0.2]$ m
T	3.0 s	β	0.8

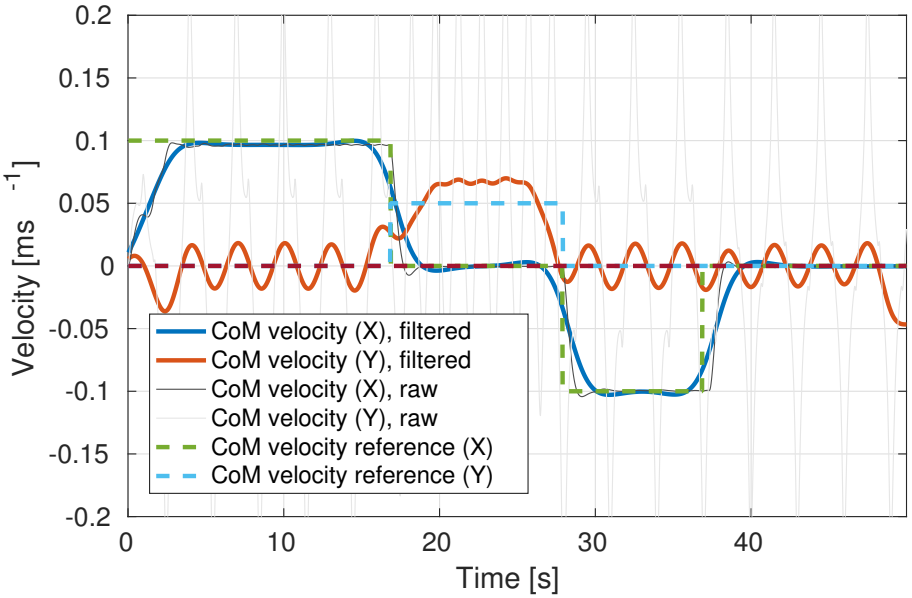

Figure 2: Planned CoM velocity profile (solid) against reference (dash). Data were processed through zero-phase low-pass filtering with cutoff frequency $f_c = 0.2$ Hz. Raw data are represented in grey.

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 4 and 2 show the achievable tracking performance. It can be noticed that, as discussed in Section 3.4, 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 5 that the proposed method does indeed generate a ZMP which is always *inside* the support polygon, therefore resulting in a feasible

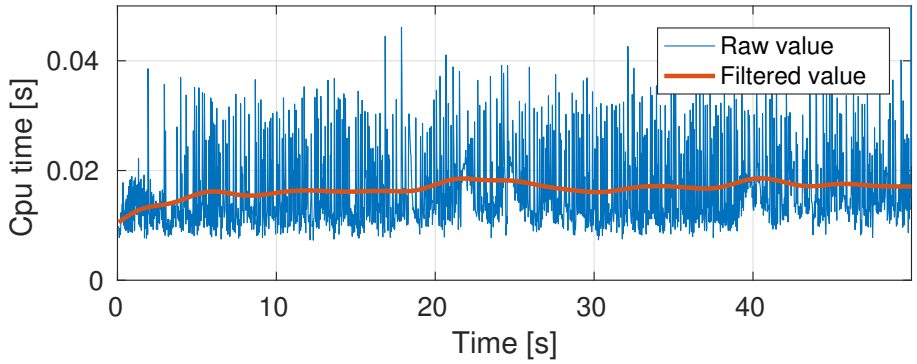

Figure 3: CPU time needed to fully set up and solve with a naive implementation the MPC QP problem on an Intel i7-6700@3400Hz CPU.

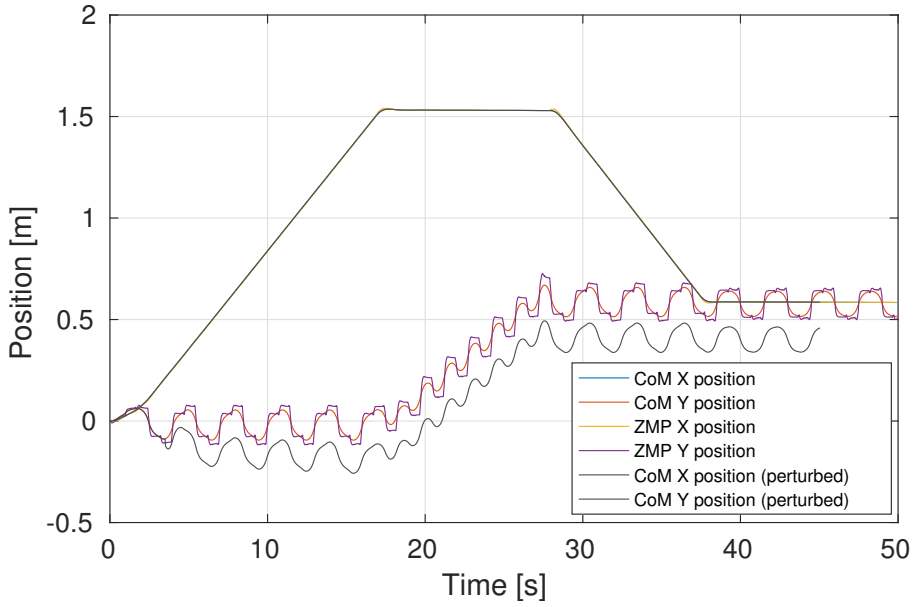

Figure 4: Planned CoM and ZMP trajectories without any external disturbance (colored lines), and with an external impulsive force $F = 60$ N applied in the negative y direction for 0.3 s (grey lines). For reference, the robot mass is roughly 90 kg.

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 4 (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 Fig. 6, and in the accompanying video as well.

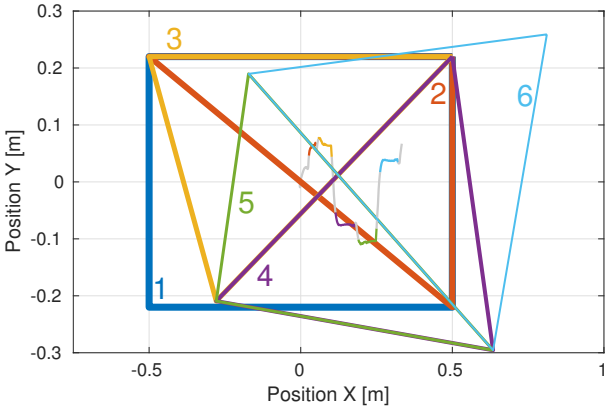

Figure 5: Sequence of support polygons generated by the proposed algorithm. The ZMP trajectory is plotted as well, with a color that matches the corresponding polygon (grey corresponds to four-stance phases).

Figure 6: Snapshots taken from an experiment on the actual Centauro robot. For the sake of clarity, the final backward phase is not included.

6 Conclusions

This deliverable presented the first walking pattern generator fully integrated with the Centauro robot. Joint optimization of both the CoM trajectory *and* the footsteps is carried out, in order to gain robustness over fixed footsteps approaches, 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 [11] 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 of computing a bad local minimum for a the original non-linear program. In addition, we achieve a significantly faster computation time when compared to [11] even with a naive, dense implementation.

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

References

[1] Holger Diedam, Dimitar Dimitrov, Pierre-Brice Wieber, Katja Mombaur, and Moritz 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*, pages 1121–1126. IEEE, 2008.

[2] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. Qpoases: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, Dec 2014.

[3] Andrei Herdt, Holger Diedam, Pierre-Brice Wieber, Dimitar Dimitrov, Katja Mombaur, and Moritz Diehl. Online walking motion generation with automatic footstep placement. *Advanced Robotics*, 24(5-6):719–737, 2010.

[4] Enrico Mingo Hoffman, Alessio Rocchi, Arturo Laurenzi, and Nikos G. Tsagarakis. Robot control for dummies: Insights and examples using opensot. In *17th IEEE-RAS International Conference on Humanoid Robotics, 2017*, pages 736–741, 2017.

[5] Shuuji Kajita, Hirohisa Hirukawa, Kensuke Harada, and Kazuhito Yokoi. *Introduction to humanoid robotics*, volume 101. Springer, 2014.

[6] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1620–1626. IEEE, 2003.

[7] Shugen Ma, Takashi Tomiyama, and Hideyuki Wada. Omnidirectional static walking of a quadruped robot. *IEEE Transactions on Robotics*, 21(2):152–161, 2005.

[8] 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)*, pages 77–80, April 2017.

[9] Pierre-Brice 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*, pages 137–142. IEEE, 2006.

[10] Pierre-Brice Wieber. Viability and predictive control for safe locomotion. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1103–1108. IEEE, 2008.

[11] Alexander W Winkler, Farbod Farshidian, Michael Neunert, Diego Pardo, and Jonas Buchli. Online walking motion and foothold optimization for quadruped locomotion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5308–5313. IEEE, 2017.

[12] Alexander W. Winkler, Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G. Caldwell, and Claudio 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)*, pages 5148–5154, 2015.