2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

A Benchmarking of DCM Based Architectures for Position and Velocity Controlled Walking of Humanoid Robots

Giulio Romualdi, Stefano Dafarra, Yue Hu, Daniele Pucci[1]

***Abstract*— This paper contributes towards the development and comparison of Divergent-Component-of-Motion (DCM) based control architectures for humanoid robot locomotion. More precisely, we present and compare several DCM based implementations of a three layer control architecture. From top to bottom, these three layers are here called: *trajectory optimization*, *simplified model control*, and *whole-body QP control*. All layers use the DCM concept to generate references for the layer below. For the *simplified model control* layer, we present and compare both instantaneous and Receding Horizon Control controllers. For the *whole-body QP control* layer, we present and compare controllers for position and velocity controlled robots. Experimental results are carried out on the one-meter-tall iCub humanoid robot. We show which implementation of the above control architecture allows the robot to achieve a walking velocity of 0.41 meters per second.**

I. INTRODUCTION

Bipedal locomotion of humanoid robots remains an open problem despite decades of research in the subject. The complexity of the robot dynamics, the unpredictability of its surrounding environment, and the low efficiency of the robot actuation system are only a few problems that complexify the achievement of robust robot locomotion. In the large variety of robot controllers for bipedal locomotion, the Divergent-Component-of-Motion (DCM) is an ubiquitous concept used for generating walking patterns. This paper presents and compares different DCM based control architectures for humanoid robot locomotion.

During the DARPA Robotics Challenge, a common approach for humanoid robot control was that of defining an hierarchical architecture composed of several layers [1]. Each layer generates references for the layer below by processing inputs from the robot, the environment, and the outputs of the layer before. From top to bottom, these layers are here called: *trajectory optimization*, *simplified model control*, and *whole-body quadratic programming (QP) control*.

The *trajectory optimization* layer often generates desired foothold locations by means of optimization techniques. To do so, both kinematic and dynamical robot models can be used [2], [3]. When solving the optimization problem associated with the *trajectory optimization* layer, computational time may be a concern especially when the robot surrounding environment is not structured. There are cases, however, where simplifying assumptions on the robot environment can be made, thus reducing the associated computational time. For instance, flat terrain allows one the view the robot as a simple unicycle [4], [5], which enables quick solutions to the optimization problem for the walking pattern generation [6].

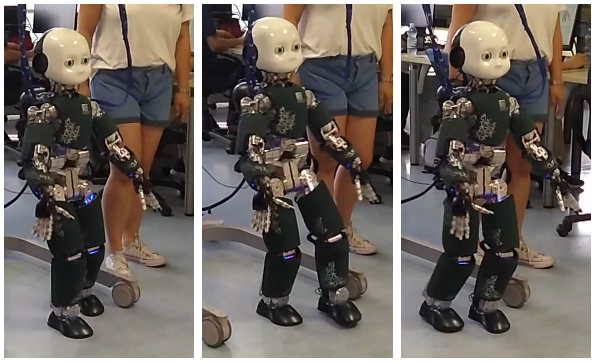

Fig. 1: iCub walks with the presented controller architecture.

The *simplified model control* layer is in charge of finding feasible center-of-mass (CoM) trajectories and is often based on simplified dynamical models, such as the Linear Inverted Pendulum Model (LIPM) [7] and the Capture Point (CP) [8]. These models have become very popular after the introduction of the Zero Moment Point (ZMP) as a stability criterion [9]. To obtain feasible CoM trajectories, the *simplified model control* layer often combines the LIPM with Model Predictive Control (MPC) techniques, also known as the Receding Horizon Control (RHC) [10], [11]. Another model that is often exploited in the *simplified model control* layer is the Divergent Component of Motion (DCM) [12]. The DCM can be viewed as the extension of the capture point (CP) to the three dimensional case, however always under the assumption of a constant Virtual Repellent Point (VRP) to Enhanced Centroidal Moment Pivot point (eCMP) height difference [12]. Attempts at loosening this latter assumption and extending the DCM to more complex models have also been presented [13].

The *whole-body QP control* layer generates robot positions, velocity or torques depending on the available control modes of the underlying robot. These outputs aim at stabilizing the references generated by the layers before. It uses whole-body kinematic or dynamical models, and very often instantaneous optimization techniques, namely, no MPC methods are here employed. Furthermore, the associated optimisation problem is often framed as an hierarchical stack-of-tasks, with strict or weighted hierarchies [14], [15].

This paper presents and compares several DCM based implementations of the above layered control architecture. In particular, the *trajectory optimization* layer is kept fixed with a unicycle based planner that generates desired DCM and foot trajectories. The *simplified model control* layer, instead,

[1] Romualdi, Dafarra, Hu and Pucci are with the Fondazione Istituto Italiano di Tecnologia, 16163 Genova, Italy (e-mail: name.surname@iit.it).

implements two controllers for the tracking of the DCM: an instantaneous and an MPC controller. In the same layer, we also present a controller that ensures the tracking of the CoM and the ZMP, which exploits 6-axes Force Torque sensors (F/T). Finally, the *whole-body QP control* ensures the tracking of the desired CoM and feet trajectories. It achieves this by presenting velocity and inverse kinematic based controllers. The several combinations of the control architecture are tested on the iCub humanoid robot [16] .

The paper is organized as follows. Sec. II introduces notation, the humanoid robot model, and some simplified models used for locomotion. Sec. III describes each layer of the control architecture, namely the trajectory optimization, the simplified model control and and the whole-body QP control layer. Sec. IV presents the experimental validation of the proposed approach, and shows an explanatory table comparing the different control approaches. Finally, Sec. V concludes the paper.

II. BACKGROUND

A. Notation

- I_n and 0_n are used to denote respectively the $n \times n$ identity and zero matrices;
- $\mathcal{I}$ denotes an inertial frame;
- $e_i \in \mathbb{R}^n$ is the canonical vector, consisting of all zeros but the i-th component that is equal to one;
- given two frames, $\mathcal{A}$ and $\mathcal{B}$, ${}^{\mathcal{A}}R_B \in SO(3)$ represents the rotation matrix between the frames, i.e. given two vectors ${}^{\mathcal{A}}p, {}^{\mathcal{B}}p \in \mathbb{R}^3$ respectively expressed in $\mathcal{A}$ and $\mathcal{B}$, the rotation matrix ${}^{\mathcal{A}}R_{\mathcal{B}}$ is such that ${}^{\mathcal{A}}p = {}^{\mathcal{A}}R_{\mathcal{B}}\,{}^{\mathcal{B}}p$;
- given a skew-symmetric matrix $W \in \mathfrak{so}(3)$ the *vee operator* is $.^{\vee} : \mathfrak{so}(3) \to \mathbb{R}^3$;
- ${}^{\mathcal{A}}\omega_{\mathcal{B}} \in \mathbb{R}^3$ denotes the angular velocity between the frame $\mathcal{B}$ and the frame $\mathcal{A}$ expressed in the frame $\mathcal{A}$;
- given a square matrix $A \in \mathbb{R}^{3\times3}$ the *skew* operator is $\text{sk} : \mathbb{R}^{3\times3} \to \mathfrak{so}(3)$, $\text{sk}(A) := (A - A^{\top})/2$;
- the subscripts $\mathcal{T}$, $\mathcal{LF}$, $\mathcal{RF}$ and $\mathcal{C}$ indicates the frames attached to the torso, left foot, right foot and CoM.

B. Humanoid robot models

A humanoid robot is an example of floating base multibody systems composed of $n+1$ links connected by n joints with one degree of freedom. Since none of the links of the robot has an a priori constant position and orientation with respect to a global frame $\mathcal{I}$, its configuration can be determined by the position and the orientation of the base frame $\mathcal{B}$ (e.g. the stance foot during the locomotion) and the joints values. Thus the configuration space is defined by $\mathbb{Q} = SO(3) \times \mathbb{R}^3 \times \mathbb{R}^n$. An element of $\mathbb{Q}$ is then a triplet $q = ({}^{\mathcal{I}}p_{\mathcal{B}}, {}^{\mathcal{I}}R_{\mathcal{B}}, s)$, where $({}^{\mathcal{I}}p_{\mathcal{B}}, {}^{\mathcal{I}}R_{\mathcal{B}}) \in (SO(3) \times \mathbb{R}^3)$ is used to represent the position and the orientation of the base frame, $\mathcal{B}$, expressed with respect to the inertial frame $\mathcal{I}$; and $s \in \mathbb{R}^n$ represents the joint angles. Furthermore, according to the group theory, $\mathbb{Q}$ is a Lie group. Indeed given two elements $a = (p_a, R_a, s_a) \in \mathbb{Q}$ and $b = (p_b, R_b, s_b) \in \mathbb{Q}$ the group multiplication $a \cdot b$ is defined as $a \cdot b = (p_a + p_b, R_aR_b, s_a + s_b) \in \mathbb{Q}$, while the inverse of a is given by: $a^{-1} = (-p_a, R_a^{\top}, -s_a) \in \mathbb{Q}$. Since $\mathbb{Q}$ is a Lie group, the velocity of the multibody system is represented by the Lie algebra $\mathbb{V}$. An element of the Lie algebra $\mathbb{V}$ is a triplet $\nu = ({}^{\mathcal{I}}\dot{p}_{\mathcal{B}}, {}^{\mathcal{I}}\omega_{\mathcal{B}}, \dot{s})$ where ${}^{\mathcal{I}}\omega_{\mathcal{B}}$ is the angular velocity of the base with respect the inertial frame whose coordinates are written in the inertial frame, i.e. ${}^{\mathcal{I}}\dot{R}_{\mathcal{B}} = {}^{\mathcal{I}}\hat{\omega}_{\mathcal{B}}\,{}^{\mathcal{I}}R_{\mathcal{B}}$.

Given a link of the floating base system its position and orientation with respect to the inertial frame is uniquely identified by an homogeneous transformation, ${}^{\mathcal{I}}H_{\mathcal{A}} \in SE(3)$, between the inertial frame, $\mathcal{I}$, and the frame attached to that link, $\mathcal{A}$. ${}^{\mathcal{I}}H_{\mathcal{A}}$ is the classical homogeneous transformation containing the rotation matrix, ${}^{\mathcal{I}}R_{\mathcal{A}}$, and the vector connecting the origin of the inertial frame to origin of the frame $\mathcal{A}$ expressed in the inertial frame ${}^{\mathcal{I}}p_{\mathcal{A}}$.

Given a link of the floating base system the velocity of a rigid body can be represented by ${}^{\mathcal{I}}v_{\mathcal{A}} = \begin{bmatrix} {}^{\mathcal{I}}\dot{p}_{\mathcal{A}}^{\top} & {}^{\mathcal{I}}\omega_{\mathcal{A}}^{\top} \end{bmatrix}^{\top}$.

The Jacobian $J_{\mathcal{A}}(q)$ is the map between the robot velocity and the linear and angular velocities of the frame $\mathcal{A}$, i.e. ${}^{\mathcal{I}}v_{\mathcal{A}} = J_{\mathcal{A}}\nu$ where $J_{\mathcal{A}}$ can be split into two parts, one related to the velocity of the base and the other one multiplying the joints velocity, $J_{\mathcal{A}}(q) = \begin{bmatrix} J_{\mathcal{A}}^{b} & J_{\mathcal{A}}^{j} \end{bmatrix}$.

C. Simplified models

For the purpose of this work, the motion of the humanoid robot is approximated by means of the well known *Linear inverted pendulum model* (LIPM) [7]. By using the LIPM, in case of walking on a flat surface the CoM belongs to an horizontal plane with a constant height z_0. The simplified CoM dynamics is given by [7]:

$$\ddot{x} = \omega^2(x - r^{zmp}), \tag{1}$$

where $x \in \mathbb{R}^2$ is the vector containing the projection of the CoM on the walking surface, $r^{zmp} \in \mathbb{R}^2$ is the position of the zero moment point (ZMP) and ω is the inverse of the pendulum time constant, i.e. $\omega = \sqrt{g/z_0}$ where g is the gravity constant.

Analogously, one can define the divergent component of motion (DCM) as $\xi = x + \dot{x}/\omega$ [12]. Clearly, the DCM time derivative is given by:

$$\dot{\xi} = \omega(\xi - r^{zmp}). \tag{2}$$

Using the DCM as state variable, the LIPM dynamics (1) can be decomposed into two parts:

$$\begin{bmatrix} \dot{x} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} -\omega I_2 & \omega I_2 \\ 0_2 & \omega I_2 \end{bmatrix} \begin{bmatrix} x \\ \xi \end{bmatrix} + \begin{bmatrix} 0_2 \\ \omega I_2 \end{bmatrix} r^{zmp}. \tag{3}$$

Performing the state space decomposition we can easily show that the CoM tends the DCM, while the DCM dynamics has a strictly positive real part eigenvalue.

III. ARCHITECTURE

This section summarizes the components of the control architecture presented in Fig. 2. In particular, the control architecture is composed of three main layers. The first layer is represented by the *trajectory generator*, whose main

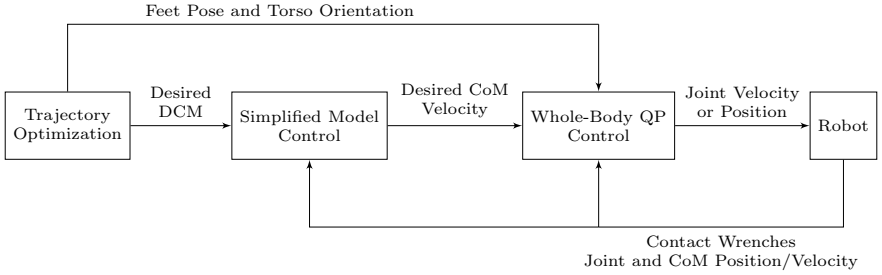

Fig. 2: The control architecture is composed of three layers: the *trajectory optimization*, the *simplified model control*, and the *whole-body QP control*.

purpose is to generate desired footstep positions and orientation and also the desired DCM trajectory. The second layer employs *simplified robot models* to track the desired DCM, CoM and ZMP trajectories. Finally, the third control layer is given by the *whole-body QP* block. It has the main purpose of ensuring the tracking of the desired feet positions and orientations and also the CoM trajectories. This, differently from the previous control layer, exploits whole-body robot kinematic models and feedback.

A. Trajectory optimization layer

The main purpose of this layer is to evaluate the desired footstep positions and the desired feet and DCM trajectories.

To plan the desired footstep positions, the humanoid robot is approximated as an unicycle [6]. The feet are represented by the unicycle wheels, and the footsteps can be obtained through sampling of the unicycle trajectories. In particular, given the finite set of unicycle positions, we can sample particular feet positions. Each position is associated with a time instant k. This time instant is considered as the foot impact time t_{imp}. Thus, the impact time can be used as a decision variable, which allows us to select the feet position and avoid fast/slow step duration and long/short step length. Furthermore, the footsteps are planned to avoid excessive rotation between the two feet positions that could be unfeasible because of joint limits. Once the footsteps are planned, the desired feet trajectory is obtained by cubic spline interpolation.

The DCM is chosen so as to satisfy the following time evolution:

$$\xi = r^{zmp} + e^{\omega t}(\xi_0 - r^{zmp}), \tag{4}$$

where ξ_0 is the initial position of the DCM, r^{zmp} is the position of the ZMP (placed on the center of the stance foot) and t has to belong to the step domain $t \in [0,\ t_i^{step}]$ where t_i^{step} is the duration of the i-th step. Assuming that the final position of the DCM coincides with the ZMP at last step (i.e. $\xi_{N-1}^{eos} = r_N^{zmp}$), (4) can be used to find the desired DCM position at the end of each step [17]

$$\begin{cases} \xi_{N-1}^{eos} = r_N^{zmp} \\ \xi_{i-1}^{eos} = \xi_i^{ios} = r_i^{zmp} + e^{-\omega t_i^{step}}(\xi_i^{eos} - r_i^{zmp}), \end{cases} \tag{5}$$

where ξ_i^{ios} and ξ_i^{eos} are respectively the desired DCM initial and final positions for the i-th step.

By substitution of (5) into (4), one obtains the reference DCM trajectory:

$$\xi_i(t) = r_i^{zmp} + e^{\omega(t-t_i^{step})}(\xi_i^{eos} - r_i^{zmp}). \tag{6}$$

The DCM velocity can be easily evaluated via differentiation of (6): $\dot{\xi}_i(t) = \omega e^{\omega(t-t_i^{step})}(\xi_i^{eos} - r_i^{zmp})$. In light of the above, the DCM trajectory along the walking pattern can be computed recursively. The presented planning method is very powerful and it allows us to generate the desired DCM trajectory in real-time. Nevertheless, it has the main limitation of taking into account single support phases only. Indeed, by considering instantaneous transitions between two consecutive single support phases, the ZMP reference is discontinuous. This leads to the discontinuity of the external forces and also of the desired joint torques. The development of a DCM trajectory generator that handles non-instantaneous transitions between two single support phases becomes pivotal [17].

Since the ZMP is related to the DCM position and velocity, namely $r^{zmp} = \xi - \dot{\xi}/\omega$, a DCM reference trajectory with continuous derivative (C^1 continuity) guarantees a continuous ZMP trajectory. This can be easily ensured using a third order polynomial to smooth the edges of the DCM reference trajectory evaluated using the exponential technique [17]:

$$\xi^{DS} = a_3 t^3 + a_2 t^2 + a_1 t + a_0,$$

where the parameters a_i for $i = 0:3$ have to be chosen in order to satisfy the velocity and position boundary conditions, namely:

$$\begin{cases} \xi_i^{DS_i} = r_{i-1}^{zmp} + e^{\omega(t_{i-1}^{DS_i} - t_{i-1}^{step})}(\xi_{i-1}^{eos} - r_{i-1}^{zmp}) \\ \dot{\xi}_i^{DS_i} = \omega e^{\omega(t_{i-1}^{DS_i} - t_{i-1}^{step})}(\xi_{i-1}^{eos} - r_{i-1}^{zmp}) \\ \xi_i^{DS_e} = r_i^{zmp} + e^{\omega(t_i^{DS_e} - t_i^{step})}(\xi_i^{eos} - r_i^{zmp}) \\ \dot{\xi}_i^{DS_e} = \omega e^{\omega(t_i^{DS_e} - t_i^{step})}(\xi_i^{eos} - r_i^{zmp}). \end{cases}$$

Here ξ^{DS_i} and $\dot{\xi}^{DS_i}$ are the desired DCM position and velocity at the beginning of the double support phase, ξ^{DS_e} and $\dot{\xi}^{DS_e}$ are the desired DCM position and velocity at the end of the double support phase, t^{DS_i} and t^{DS_e} are the initial and final instant of the double support phase.

In Fig. 3 the whole DCM trajectory is shown. During the single support phase (orange segment) the exponential interpolation technique (6) is used; while during the double support phase (light blue curves) the trajectory is obtained using the polynomial technique described above.

B. Simplified model control layer

As discussed in Sec. II-C, the simplified model (3) shows that the CoM asymptotically converges to a constant DCM, while the DCM has an unstable first order dynamics. Then,

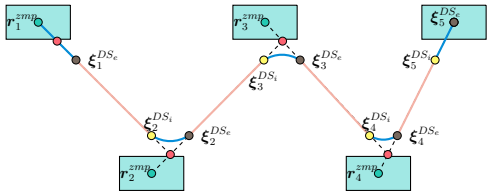

Fig. 3: Planning of 2D-DCM on a flat terrain. The single support trajectories are represented by orange segments. The double support trajectories are represented by light blue curves.

we present in this section control laws that aim at stabilizing the DCM dynamics only assuming the r^{zmp} as control input.

This stabilization problem has been tackled by designing and comparing *instantaneous* and *predictive* (MPC) control laws. Differently from the MPC, the instantaneous control law does not solve any optimization problem and it uses only the current desired and actual position of the DCM to evaluate its output.

1) DCM instantaneous control: Differently from [12], we propose an instantaneous control law that does not perform a cancellation of the unstable system dynamics, which, in practice, is never perfect and may lead to system instabilities. More precisely, we choose

$$r^{zmp}_{ref}=\xi_{ref}-\frac{\dot{\xi}_{ref}}{\omega}+K^{\xi}_{p}(\xi-\xi_{ref})+K^{\xi}_{i}\int(\xi-\xi_{ref})\,\mathrm{d}t, \quad (7)$$

where $K^{\xi}_{p} > I_2$ and $K^{\xi}_{i} > 0_2$

Applying the control input (7) to system (2) leads to the following closed loop dynamics:

$$(\dot{\xi}-\dot{\xi}_{ref})=\omega(I_2-K^{\xi}_{p})(\xi-\xi_{ref})-\omega K^{\xi}_{i}\int(\xi-\xi_{ref})\,\mathrm{d}t. \quad (8)$$

It is easy to show the DCM error and its integral converge asymptotically to zero. The above law (7) is very simple to implement and guarantees the tracking of the desired DCM. The main limitation is that may not ensure the feasibility of the gait since the position of the ZMP may exit the support polygon. Furthermore, the above instantaneous control law does not take into account the future planned desired DCM trajectory.

2) DCM predictive control: In order to overcome these issues, a model predictive controller can be designed [18].

In the MPC framework, the DCM dynamics (2) is used as a prediction model and it is discretized supposing piecewise constant ZMP trajectories:

$$\begin{aligned}\xi_{k+1}&=F\xi_k+Gr^{zmp}_k\\&=\begin{bmatrix}e^{\omega T} & 0\\ 0 & e^{\omega T}\end{bmatrix}\xi_k+\begin{bmatrix}1-e^{\omega T} & 0\\ 0 & 1-e^{\omega T}\end{bmatrix}r^{zmp}_k.\end{aligned} \quad (9)$$

In order to ensure that the stance foot does not rotate around one of its edges, the desired ZMP must not exit the support polygon [19]. This is verified by means of a set of linear inequality constraints:

$$A_{c_k}r^{zmp}_k \le b_{c_k}, \quad (10)$$

where A_{c_k} and b_{c_k} are time variant and their dimension depends on the type of support.

The cost function shall then ensure the tracking of the desired trajectory. In particular, we choose the following cost:

$$J_k = \sum_{j=k}^{N+k-1}(\xi_j-\xi^{ref}_j)^{\top}Q(\xi_j-\xi^{ref}_j)+ \quad (11a)$$

$$(r^{zmp}_j-r^{zmp}_{j-1})^{\top}R(r^{zmp}_j-r^{zmp}_{j-1})+ \quad (11b)$$

$$(\xi_{k+N}-\xi^{ref}_{k+N})^{\top}Q_N(\xi_{k+N}-\xi^{ref}_{k+N}) \quad (11c)$$

where Q, Q_N and R are positive 2×2 symmetric matrices and N is length of the preview window.

The terms (11c) (11a) guarantee the tracking of the given reference trajectory, while (11b) is added for obtaining a smoother ZMP trajectory. Even if the cost function J_k is time-dependent, it is always positive and convex.

The MPC problem can be summarized as follows:

$$\begin{aligned}r^{zmp^*}_k = \underset{\substack{\xi_k,\dots,\xi_{k+N}\\ r^{zmp}_k,\dots,r^{zmp}_{k+N-1}}}{\arg\min}\quad & J_k \\ \text{s.t.}\quad & \xi_{i+1}=F\xi_i+Gr^{zmp}_i\\ & A_{c_k}r^{zmp}_k \le b_{c_k}\\ & \xi_k=\bar{\xi}\\ & r^{zmp}_{k-1}=\bar{r}^{zmp},\end{aligned} \quad (12)$$

where i satisfies $k \le i \le N+k-1$, $\bar{r}^{zmp}$ is the desired ZMP computed at the previous control iteration and $\bar{\xi}$ is the estimated DCM position.

Since the cost function is a quadratic positive function and the constraints are linear, the optimal control problem is quadratic and it can be converted into a strictly convex quadratic programming problem (QP) of the form:

$$\begin{aligned}\min_{w_k}\ & \frac{1}{2}w_k^{\top}H_kw_k+g_k^{\top}w_k\\ \text{s.t.}\ & A^i_{c_k}w_k \le b^i_{c_k}\\ & A^e_{c_k}w_k = b^e_{c_k}.\end{aligned}$$

Here the optimization variables are stacked inside the vector $w_k=\begin{bmatrix}\xi_k^{\top} & \dots & \xi_{k+N}^{\top} & r_k^{zmp^{\top}} & \dots & r_{k+N-1}^{zmp^{\top}}\end{bmatrix}^{\top}$. The Hessian matrix H_k and the gradient vector g_k can be obtained from (11). The inequality constraint matrix $A^i_{c_k}$ and vector $b^i_{c_k}$ embed the ZMP constraint (10). While the equality constraints, $A^e_{c_k}$ and $b^e_{c_k}$ are determined using the prediction model (9).

3) ZMP-CoM Controller: Independently from the chosen DCM controller, namely either the controller in Sec. III-B.1 or in III-B.2, one obtains a desired ZMP and CoM position and velocity to be stabilised. As consequence, another control loop is needed after the DCM controller. In this paper, we choose the proposed control law [20], i.e.:

$$\dot{x}^* = \dot{x}_{ref}-K_{zmp}(r^{zmp}_{ref}-r^{zmp})+K_{com}(x_{ref}-x), \quad (13)$$

where $K_{com} > \omega I_2$ and $0_2 < K_{zmp} < \omega I_2$.

C. Whole-body QP control layer

The main control objective for the whole-body QP control layer is to stabilise some robot kinematic quantities by using the entire robot body. We here choose to track the position of the CoM, the torso orientation, and the left and right feet position and orientation. To do so, we use a stack of tasks formulation. The tracking of the feet poses and of the CoM position is considered as high priority tasks (hard constraint), while the torso orientation is considered as a low priority task (soft constraint). Furthermore, a postural condition is also added as a low-priority task. Then, the following cost function is defined:

$$f(\nu)=\frac{1}{2}[(v^*_{\mathcal{T}}-J_{\mathcal{T}}\nu)^\top K_{\mathcal{T}}(v^*_{\mathcal{T}}-J_{\mathcal{T}}\nu)+(\dot{s}-\dot{s}^*)^\top\Lambda(\dot{s}-\dot{s}^*)]. \quad (14)$$

with $K_{\mathcal{T}} > 0$ and $v^*_{\mathcal{T}} = -K_{\omega_{\mathcal{T}}}\mathrm{sk}({}^{\mathcal{I}}R_{\mathcal{T}}\,{}^{\mathcal{I}}R^{*\top}_{\mathcal{T}})^\vee$, where $K_{\omega_{\mathcal{T}}} > 0$. This latter control law guarantees almost-global stability and convergence of ${}^{\mathcal{I}}R_{\mathcal{T}}$ to ${}^{\mathcal{I}}R^*_{\mathcal{T}}$ [21].

The postural task (14), with $\Lambda > 0$, is achieved by asking for a desired joints velocity that depends on the error between the desired and measured joints position

$$\dot{s}^* = -K_s(s - s^d), \quad (15)$$

where K_s is a positive definite matrix.

Concerning the hard constraints, we have:

$$J_{\mathcal{C}}(\nu)\nu = v^*_{\mathcal{C}}, \quad J_{\mathcal{LF}}(\nu)\nu = v^*_{\mathcal{LF}}, \quad J_{\mathcal{RF}}(\nu)\nu = v^*_{\mathcal{RF}}, \quad (16)$$

where $v^*_{\mathcal{C}}$ is the linear velocity of the CoM, $v^*_{\mathcal{LF}}$ and $v^*_{\mathcal{RF}}$ are respectively the desired left foot and right foot velocities. More specifically $v^*_{\#\mathcal{F}}$, where $\# = [\mathcal{R}, \mathcal{L}]$, is chosen as:

$$v^*_{\#\mathcal{F}} = {}^{\mathcal{I}}\dot{p}^*_{\#\mathcal{F}} - \begin{bmatrix} K^p_{x_{\#f}} e^p_{\#f} + K^i_{x_{\#f}} \int e^p_{\#f}\,\mathrm{d}t \\ K_{\omega_{\#f}}\mathrm{sk}({}^{\mathcal{I}}R_{\#\mathcal{F}}\,{}^{\mathcal{I}}R^{*\top}_{\#\mathcal{F}})^\vee \end{bmatrix}. \quad (17)$$

Here $e^p_{\#f} = {}^{\mathcal{I}}p_{\#\mathcal{F}} - {}^{\mathcal{I}}p^*_{\#\mathcal{F}}$, while the gains $K^p_{x_{\#f}}$, $K^i_{x_{\#f}}$, $K_{\omega_{\#f}}$ are positive definite matrices.

Finally, the desired velocity of the CoM $v^*_{\mathcal{C}}$ is chosen as:

$$v^*_{\mathcal{C}} = \dot{x}^* - K^p_{\mathcal{C}}(x - x^*) - K^i_{\mathcal{C}} \int x - x^*\,\mathrm{d}t,$$

where the gain matrices are positive definite positive $\dot{x}^*$ is the output of the ZMP-CoM (13) controller and x^* is the integrated signal. Finally, we add constraints on the maximum velocity

$$\dot{s}^- \leq \dot{s} \leq \dot{s}^+. \quad (18)$$

The above hierarchical control objectives can be cast into a whole-body optimization problem:

$$\begin{aligned} \nu^* = \arg\min_{\nu} \frac{1}{2}[&(v^*_{\mathcal{T}} - J_{\mathcal{T}}\nu)^\top K_{\mathcal{T}}(v^*_{\mathcal{T}} - J_{\mathcal{T}}\nu) + \\ &(\dot{s} - \dot{s}^*)^\top\Lambda(\dot{s} - \dot{s}^*)] \quad (19) \\ \text{s.t.} \quad &\dot{s} = S\nu \\ &\dot{s}^* = -K_s(s - s^d) \\ &J_{\mathcal{C}}\nu = v^*_{\mathcal{C}} \\ &J_{\mathcal{LF}}\nu = v^*_{\mathcal{LF}} \quad J_{\mathcal{RF}}\nu = v^*_{\mathcal{RF}} \\ &\dot{s}^- \leq \dot{s} \leq \dot{s}^+ \end{aligned}$$

Since the decision variable is the robot velocity ν and the body velocity depends, through the Jacobian matrices, linearly on ν the optimization problem can be converted to the QP problem of the form:

$$\begin{aligned} &\min_{\nu} \frac{1}{2}\nu^\top H\nu + g^\top\nu \\ &\text{s.t. } A_c\nu = b_c \\ &\quad \dot{s}^- \leq \dot{s} \leq \dot{s}^+. \end{aligned}$$

The Hessian matrix H and the gradient vector g are evaluated from (14). The constraint matrix and vector A_c and b_c are obtained from (16). Using this formulation the optimization problem can be solved using a standard numerical QP solver.

1) Position and velocity controlled robot: It is important to notice that the outcome of (19) is (also) the robot joint velocity. When a robot velocity controller is available, one can set these joint velocities to the low level robot controller. In this case, the * quantities in (19) can be evaluated by using robot sensor feedback, and the robot is said to be *velocity controlled*. On the other hand, if the robot velocity control is not available, one may integrate the outcome of (19) to obtain desired joint position to be set to a low level robot position controller. In this case, the * quantities in (19) can be evaluated by using the desired integrated quantities instead of sensor feedback, and (19) behaves as an inverse kinematics module, and the robot is said to be *position controlled*.

IV. EXPERIMENTAL RESULTS

In this section, we present experiments obtained with several implementations of the control architecture shown in Fig. 2. We use the iCub humanoid robot [22] to carry out the experimental activities. Let us recall that iCub is $104\,\mathrm{cm}$ tall, with a foot length and width of $19\,\mathrm{cm}$ and $9\,\mathrm{cm}$, respectively.

The control architecture runs on the iCub head's computer, namely a 4-th generation Intel ® Core i7 @ $1.7\,\mathrm{GHz}$. In any of its implementations, the architecture takes (in average) less than $3\,\mathrm{ms}$ for evaluating its outputs. OSQP [23] library was used to solve the optimization problems.

Tab. I summarizes the maximum velocities achieved using the different implementations of the control architecture. In particular, the labels *instantaneous* and *predictive* mean that the associated layer generates its outputs considering inputs and references either at the single time t or for a time window, respectively. The labels, *velocity* and *position* control, instead, mean that the layer outputs are either desired joint velocities or position, respectively – see Sec. III-C.1.

Let us remark that all the implemented control architectures exploit the controller presented in section III-B.3 to attempt the stabilization of the desired center-of-pressure

TABLE I: Maximum forward straight walking velocities achieved using different implementations of the control architecture.

Simplified Model Control	Whole-Body QP Control	Max Straight Velocity (m/s)
Predictive	Velocity	0.19
Predictive	Position	0.20
Instantaneous	Velocity	0.22
Instantaneous	Position	0.41

and desired center-of-mass position and velocity. The performances of this controller much depend on the gains K_{zmp} and K_{com}. In particular, we observed that the gains achieving good tracking during standing and walking were not the same. For this reason, we implemented a gain-scheduling technique depending on the robot is walking or standing. The transition between the two sets of gains is smoothed with a minimum jerk trajectory [24].

Comparing different control architectures, however, is a far cry from being an easy task. To do so, we decided to perform two main experiments only, which are used as benchmarks for all control architecture implementations. These two experiments represent the maximum robot velocity that has been achieved with all architectures, and the maximum velocity achieved with a specific architecture only – see Tab. I. Namely,

- **Experiment 1** a forward robot speed of $0.19\,\mathrm{m\,s^{-1}}$;
- **Experiment 2** a forward robot speed of $0.41\,\mathrm{m\,s^{-1}}$.

A. Simplified model control: Predictive versus Instantaneous

In this section, we compare the control laws (7) and (12), which both generate a (desired) center-of-pressure that attempt at stabilizing a desired DCM. To simplify the comparison, the controller of the *whole-body QP layer* is kept fixed in this section, and we show and discuss only the results when the robot is position controlled. The time horizon of the predictive control is $2\,\mathrm{s}$.

1) Experiment 1: Figs. 4a and 4d depict the DCM tracking performances obtained with the instantaneous and predictive controllers, respectively. Both controllers seem to show good tracking performances, and the DCM error is kept below $5\,\mathrm{cm}$ in both cases. Note that the instantaneous controller induces, however, faster variations of the measured DCM. This contributes to overall higher vibrations of the robot. One of the reasons for this variation is that the instantaneous controller (7) injects a (desired) center-of-pressure proportional to the measured DCM $\xi = x + \dot{x}/\omega$, which in turns contains the center-of-mass velocity. To mitigate this, we suggest to filter joint velocities appropriately. In our case, however, joint velocities were not filtered to avoid delays in the measured DCM. Our experience showed that adding a filter on joint velocities is not an easy task, and we did not find the right trade off for obtaining overall performance improvements.

Figs. 4b and 4e, depict the CoM tracking performances, which are mainly dependent on the ZMP-CoM controller (13). This controller receives desired DCM values from the *simplified model control* layer, which are obtained with either the instantaneous or predictive controllers. In both cases, CoM tracking performances are good, and the CoM error is kept below $2\,\mathrm{cm}$.

Figs. 4c and 4f depict the ZMP tracking performances, which are still mainly dependent on the ZMP-CoM controller (13). It is important to observe that the desired ZMP is smoother when the *simplified model control* uses the predictive law (12) to generate it. This is a tunable property that depends on the associated weight in the cost function of the MPC problem. Although this smoother behaviour does contribute to less robot vibrations, the overall robot performance became less reactive and, consequently, less robust to robot falls. Although the extensive hand-made tuning, we were not able to increase the robot velocity when the *simplified model control* used the predictive law (12).

2) Experiment 2: At a robot desired walking speed of $0.41\,\mathrm{m\,s^{-1}}$, there is initially no significant difference between the DCM tracking obtained with instantaneous and predictive control laws – see Fig. 5d and 5a for $t < 1.5\,\mathrm{s}$. However, fast robot walking velocities require fast variations of the desired CoM and ZMP. In the case of the predictive controller, this fast variation induces a not-very-good performance of the desired ZMP around $t = 1.5\,\mathrm{s}$ – see Fig. 5f. Clearly, these bad performances in turn induce a bad tracking of the DCM shown in Fig. 5d at $t \approx 2\,\mathrm{s}$, and consequently a robot fall. At this point, one is tempted to increase the gain K_{zmp} of the controller (13), which shall induce a better tracking of the ZMP. This unfortunately gives raise to higher robot oscillations, which in turn degrades the ZMP-CoM tracking, and still a robot fall.

In a nutshell, the *predictive simplified control* is much less robust than the *instantaneous simplified control* with respect to ZMP tracking errors. We suggest to filter the measurements from force sensors to obtain less noisy ZMP measurement, which should allow one to increase the gains for ZMP tracking. In our case, adding filters led to slower system response and, consequently, a robot fall.

B. Whole-Body QP Control: Position versus Velocity

In this section, we compare the performances of the three layer architecture when the robot is either position or velocity controlled – see Sec. III-C.1 for the meaning of these modes. For comparison purposes, the *simplified model control* is achieved via the instantaneous control (7). Also, to emphasize the comparison, we stress the importance of the tracking of the desired feet positions.

1) Experiment 1: Figs. 6a and 6c depict the tracking of desired left foot positions when the robot is either position or velocity controlled, respectively. The position controller ensures better tracking performance than the velocity one. One is then tempted to increase the gains of the * quantities (17), which should increase the tracking performances of the velocity control. However, we observed that the noise due to numerical derivative is harmful for the overall performance, and makes the robot shake and then fall. Again, filtering the taken joint measures may be helpful, but it introduced a delay that did not allow us to increase the walking speed.

2) Experiment 2: The aforementioned foot position tracking problem worsens at higher walking velocity. Fig. 6b shows that the feet tracking error is lower than $5\,\mathrm{cm}$ on the x axis and $1\,\mathrm{cm}$ on the z one for position control. Instead, the velocity control in Fig. 6d keeps the error always lower than $6\,\mathrm{cm}$ on the x component and $3\,\mathrm{cm}$ on the z direction.

V. CONCLUSION AND FUTURE WORK

This paper contributes towards the benchmarking of different implementations of state-of-the-art control architectures

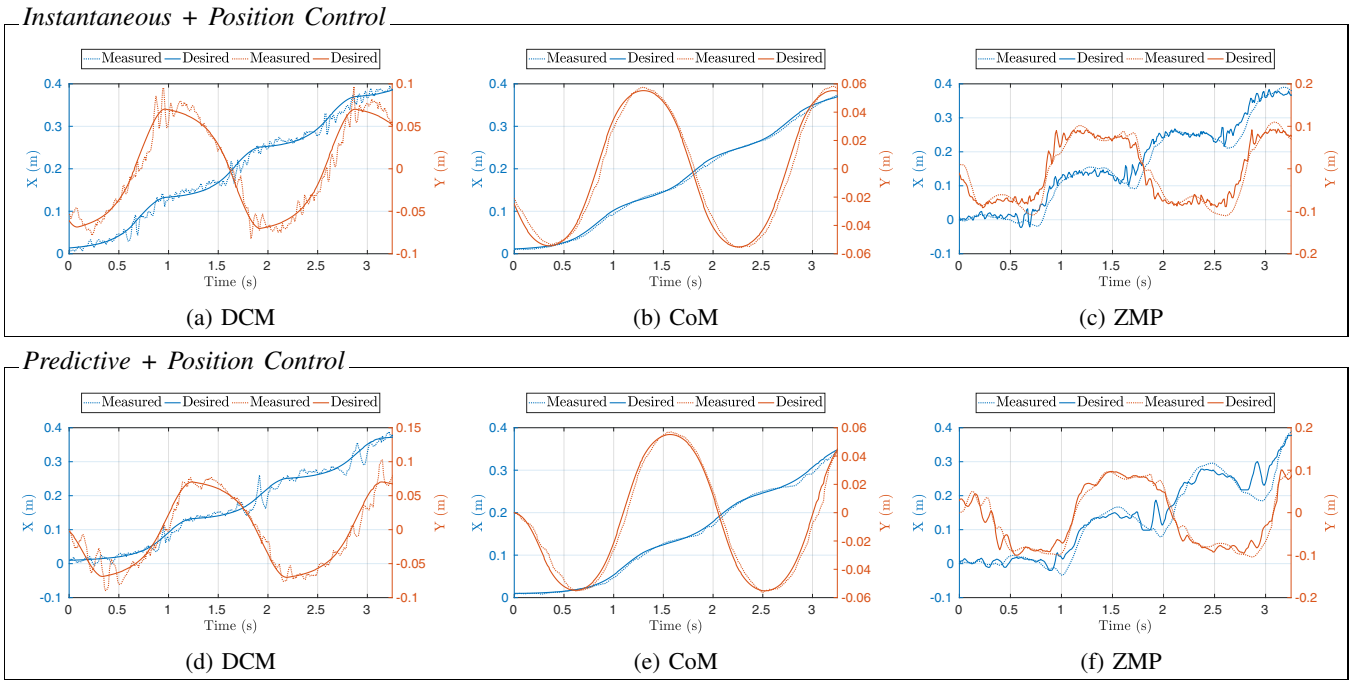

Fig. 4: Tracking of the DCM (a), CoM (b) and ZMP (c) using the instantaneous controller with the whole-body controller as position control. Tracking of the DCM (d), CoM (e) and ZMP (f) using the MPC and the whole-body controller as position control. Walking velocity: $0.19\,\mathrm{m\,s^{-1}}$.

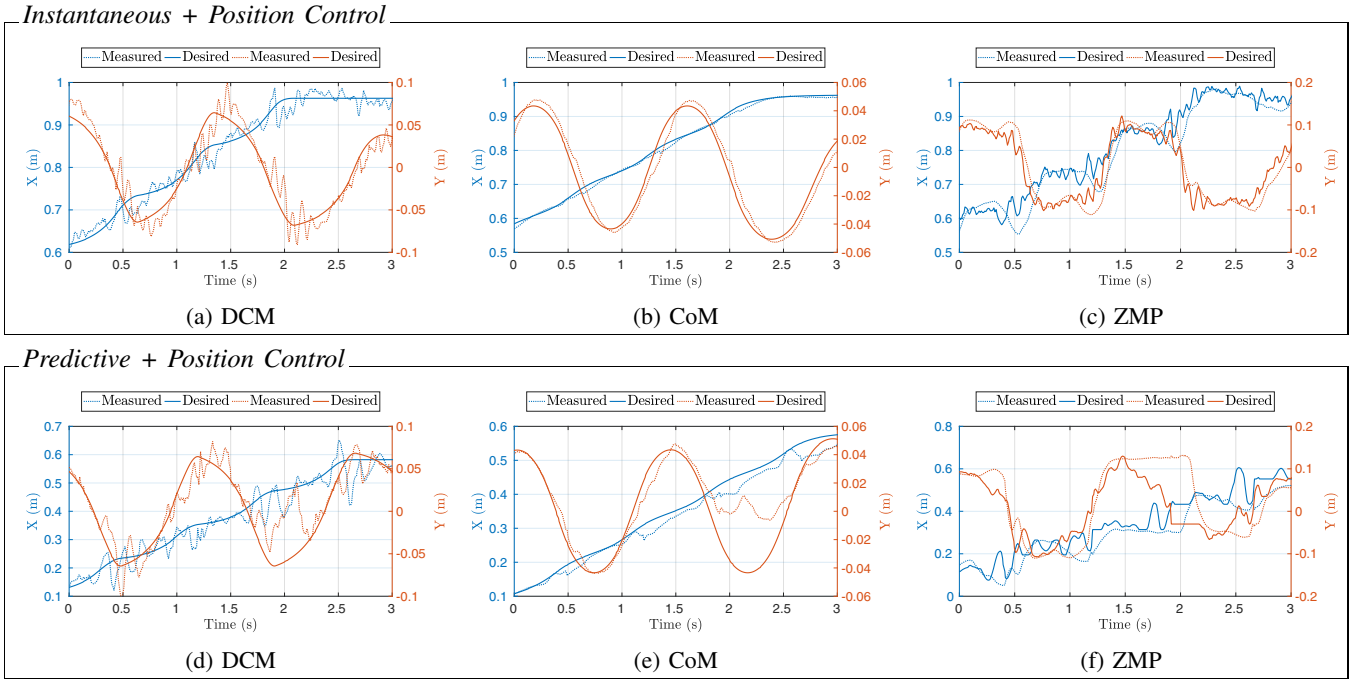

Fig. 5: Tracking of the DCM (a), CoM (b) and ZMP (c) with the instantaneous and whole-body QP control as position. Tracking of the DCM (d), CoM (e) and ZMP (f) with the predictive and whole-body QP control as position control. At $t \approx 2\,\mathrm{s}$, the robot falls down. Walking velocity: $0.41\,\mathrm{m\,s^{-1}}$.

for humanoid robots locomotion. In particular, the control architecture is composed of three layers, which all exploit the concept of the Divergent Component of Motion. The three layers are here called: trajectory optimization, simplified model control, the whole-body QP control. A key feature of this paper is that we compare walking results obtained with predictive and instantaneous controllers for the simplified model layer. Also, we compare position and velocity robot control modes for the whole-body QP control layer. We show that instantaneous controllers combined with robot position control allowed us to achieve a desired walking speed of $0.41\,\mathrm{m\,s^{-1}}$, which is the highest walking velocity ever achieved for the iCub humanoid robot.

As future work, we plan to extend the proposed benchmarking to torque-control algorithms (e.g. [25]), and to propose architecture implementations allowing the robot

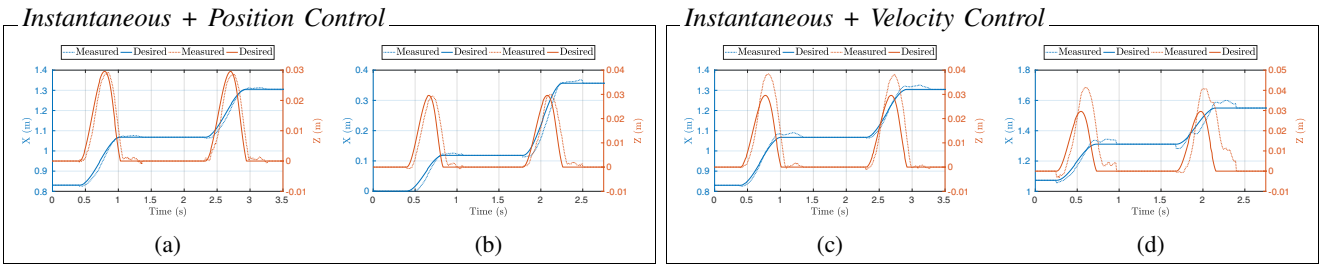

Fig. 6: Tracking of the left foot position using the instantaneous simplified model control. Whole-body QP control as position control (a) and velocity control (c), walking velocity: $0.19\,\mathrm{m\,s^{-1}}$. Whole-body QP control as position control (b) and velocity control (d), walking velocity: $0.41\,\mathrm{m\,s^{-1}}$.

walking on inclined planes. Another interesting future work is the implementation of a dynamic footstep planner. Indeed, currently, the footsteps are generated independently from the state of the robot. A dynamic planner can plan the footprints according to the external disturbance acting on the humanoid, i.e. an unknown contact force acting on the robot.

REFERENCES

[1] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," *Journal of Field Robotics*, vol. 32, no. 2, pp. 293–312, 2015.

[2] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 295–302.

[3] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum control," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 874–880.

[4] P. Morin and C. Samson, *Handbook of Robotics*. Springer, 2008, ch. Motion control of wheeled mobile robots, pp. 799–826.

[5] D. Flavigne, J. Pettrée, K. Mombaur, J.-P. Laumond, *et al.*, "Reactive synthesizing of human locomotion combining nonholonomic and holonomic behaviors," in *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*. IEEE, 2010, pp. 632–637.

[6] S. Dafarra, G. Nava, M. Charbonneau, N. Guedelha, F. Andrade, S. Traversaro, L. Fiorio, F. Romano, F. Nori, G. Metta, and D. Pucci, "A control architecture with online predictive planning for position and torque controlled walking of humanoid robots," in *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on. Preprint arXiv:1807.05395*.

[7] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum model: a simple modeling for biped walking pattern generation," *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. October 2016, pp. 239–246, 2001.

[8] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS*, 2006, pp. 200–207.

[9] M. Vukobratovic and D. Juricic, "Contribution to the Synthesis of Biped Gait," *IEEE Transactions on Biomedical Engineering*, vol. BME-16, no. 1, pp. 1–6, 1969.

[10] 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," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, no. October, pp. 1620–1626, 2003.

[11] 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 *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1121–1126.

[12] 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.

[13] M. A. Hopkins, D. W. Hong, and A. Leonessa, "Humanoid locomotion on uneven terrain using the time-varying divergent component of motion," in *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-Febru. IEEE, nov 2015, pp. 266–272.

[14] B. J. Stephens and C. G. Atkeson, "Dynamic balance force control for compliant humanoid robots," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1248–1255.

[15] G. Nava, F. Romano, F. Nori, and D. Pucci, "Stability analysis and design of momentum-based controllers for humanoid robots," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 680–687.

[16] L. Natale, C. Bartolozzi, D. Pucci, A. Wykowska, and G. Metta, "icub: The not-yet-finished story of building a robot child," *Science Robotics*, vol. 2, no. 13, 2017. [Online]. Available: http://robotics.sciencemag.org/content/2/13/eaaq1026

[17] J. Englsberger, T. Koolen, S. Bertrand, J. Pratt, C. Ott, and A. Albu-Schäffer, "Trajectory generation for continuous leg forces during double support and heel-to-toe shift based on divergent component of motion," in *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 4022–4029.

[18] M. Krause, J. Englsberger, P. B. Wieber, and C. Ott, "Stabilization of the Capture Point dynamics for bipedal walking based on model predictive control," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2012, pp. 165–171.

[19] M. Vukobratov and B. Borovac, "Zero - Moment Point Thirty Five Years of Its Life," *International Journal of Humanoid Robotics*, vol. 1, no. 1, pp. 157–173, 2004.

[20] Y. Choi, D. Kim, Y. Oh, and B.-j. J. You, "On the Walking Control for Humanoid Robot Based on Kinematic Resolution of CoM Jacobian With Embedd ed Motion," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, vol. 23, no. 6, pp. 1285–1293, 2007.

[21] R. Olfati-Saber, "Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles," Cambridge, MA, USA, 2001.

[22] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, "The iCub humanoid robot: An open-systems platform for research in cognitive development," *Neural Networks*, 2010.

[23] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *ArXiv e-prints*, Nov. 2017.

[24] U. Pattacini, F. Nori, L. Natale, G. Metta, and G. Sandini, "An experimental evaluation of a novel minimum-jerk Cartesian controller for humanoid robots," in *IEEE IROS 2010 - Conference Proceedings*, 2010.

[25] D. Pucci, F. Romano, S. Traversaro, and F. Nori, "Highly dynamic balancing via force control," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 141–141.