A Control Architecture with Online Predictive Planning for Position and Torque Controlled Walking of Humanoid Robots

Stefano Dafarra[1,2], Gabriele Nava[1,2], Marie Charbonneau[1,2], Nuno Guedelha[1,2], Francisco Andrade[1,2], Silvio Traversaro[1], Luca Fiorio[1], Francesco Romano[3], Francesco Nori[3], Giorgio Metta[1], and Daniele Pucci[1]

Abstract— A common approach to the generation of walking patterns for humanoid robots consists in adopting a layered control architecture. This paper proposes an architecture composed of three nested control loops. The outer loop exploits a robot kinematic model to plan the footstep positions. In the mid layer, a predictive controller generates a Center of Mass trajectory according to the well-known table-cart model. Through a whole-body inverse kinematics algorithm, we can define joint references for position controlled walking. The outcomes of these two loops are then interpreted as inputs of a stack-of-task QP-based torque controller, which represents the inner loop of the presented control architecture. This resulting architecture allows the robot to walk also in torque control, guaranteeing higher level of *compliance*. Real world experiments have been carried on the humanoid robot iCub.

I. INTRODUCTION

Despite decades of research in the subject, *robust* bipedal locomotion of humanoid robots is still a challenge for the Robotics community. The unpredictability of the terrain, the nonlinearity of the robot-environment models, and the *low efficiency* of the robot actuators - that are a far cry from the human musculoskeletal system - are only a few complexities that render robot bipedal locomotion an active research domain. In this context, feedback control algorithms for robust bipedal locomotion are of primary importance. This paper contributes towards this direction by presenting an on-line predictive kinematic planner for foot-step positioning and center-of-mass (CoM) trajectories. This planner is also integrated with a stack-of-task torque controller, which ensures a degree of *robot compliance* and further increases the overall system robustness to external perturbations.

A recent approach for bipedal locomotion control that became popular during the DARPA Robotics Challenge [1] consists in defining a hierarchical control architecture. Each layer of this architecture receives inputs from the robot and its surrounding environment, and provides references to the the layer below. The lower the layer, the shorter the time horizon that is used to evaluate the outputs. Also, lower layers usually employ more complex models to evaluate the outputs, but the shorter time horizon often results in faster computations for obtaining these outputs.

From higher to lower layers, *trajectory optimization for foot-step planning*, *receding horizon control (RHC)*, and *whole-body quadratic programming control* represent a common stratification of the control architecture for bipedal locomotion control.

Trajectory optimization for foot-step planning is in charge of finding a sequence of robot footholds, which is also fundamental for maintaining the robot balance. This optimisation can be achieved by considering both kinematic and dynamic robot models [2], [3], [4], and with different optimisation techniques, such as the Mixed Integer Programming [5]. The robust-and-repeatable application of these sophisticated techniques, however, still require time-consuming gain tuning since they usually consider complex models characterizing the hazardous environment surrounding the robot.

For a certain number of applications, the terrain can be considered to be flat. In these cases, it is known that the human upper body is usually kept tangent to the *walking path* [6], [7] all the more so because stepping aside, i.e. perpendicular to the path, is energetically inefficient [8]. All these considerations suggest to use a simple kinematic model to generate the walking trajectories: the unicycle model (see, e.g., [9]). This model can be used to plan footsteps in a corridor with turns and junctions using cameras [10], or to perform *evasive* robot motions [11]. In all these cases, however, the robot velocity is kept to a constant value.

Receding horizon control (RHC), also referred to as Model Predictive Control (MPC) [12], is often in charge of finding feasible trajectories for the robot center of mass along the walking path. The computational burden to find feasibility regions, however, usually calls for using simplified models to characterise the robot dynamics. The Linear Inverted Pendulum [13] and the Capture Point [14] models represent two widespread simplified robot models. These simplified linear models have allowed on-line RHC, also providing references for the footstep locations [15], [16], [17].

Whole-body quadratic-programming (QP) controllers are instantaneous algorithms that usually find (desired) joint torques and contact forces achieving some desired robot accelerations. In this framework, the generated joint-torques and contact forces can satisfy inequality constraints, which allow to meet friction constraints. The desired accelerations, that QP controllers track, shall ensure the stabilisation of reference positions coming from the RHC layer. Although the reference positions may be stabilised by a joint-position control loop, joint-torque based controllers have shown to ensure a degree of compliance, which also allows safe inter-

[1] iCub Facility Department, Istituto Italiano di Tecnologia, 16163 Genova, Italy (e-mail: name.surname@iit.it)

[2] Università degli Studi di Genova, DIBRIS

[3] Romano (fraromano@google.com) and Nori (fnori@google.com) are with Google DeepMind, London, UK.

actions with the environment [18], [19]. From the modeling point of view, full-body *floating-base* models are usually employed in QP controllers. These controllers are often composed of several tasks, organised with strict or weighted hierarchies [20], [21], [22], [23], [24].

This paper presents a *reactive* control architecture for bipedal robot locomotion, namely, the entire architecture uses on-line feedback from the robot and user-desired quantities. This architecture implements the above three layers, and can be launched on both position and torque controlled robots.

The *trajectory optimization for foot-step planning* is achieved by a planning module that uses a simplified kinematic robot model, namely, the unicycle model. Feet positions are updated depending on the robot state and on a desired direction coming from a joypad, which gives a human user teleoperating the robot the possibility of defining desired walking paths. Differently from [10], we do not fix the robot velocity. Compared to [15], [16], [17], we do not assume the robot to be always in single support. As a consequence, the robot avoids to step in place continuously if the desired robot position does not change, or changes slowly. Another drawback of these approaches is that feet rotations are planned separately from linear positions, and this drawback is overcome by our approach.

Once footsteps are defined, a *receding horizon control (RHC)* module generates kinematically feasible trajectories for the robot center of mass and joint trajectories by using the LIP model [25] and whole-body inverse kinematics. Hence, we separate the generation of footsteps from the computation of feasible CoM trajectories. The implemented RHC module runs at a frequency of 100 Hz.

The CoM and feet trajectories generated by the RHC module are then streamed as desired values to either a joint-position control loop, or to a *whole-body quadratic-programming (QP) controller* running at 100 Hz. This latter controller generates desired joint torques, ensuring a degree of robustness and robot compliance. The desired joint torques are then stabilised by a low-level joint torque controller running at 1 kHz. Experimental validations of the proposed approach are carried out on the iCub humanoid robot [26], with both position and joint torque control experiments.

In light of the above, this paper presents a torque-control, on-line RHC architecture for bipedal robot locomotion.

II. BACKGROUND

A. Notation

In this paper we are going to use the following notation:
- The i_{th} component of a vector x is denoted as x_i.
- $\mathcal{I}$ is a fixed inertial frame with respect to (w.r.t.) which the robot's absolute pose is measured.
- $e_1 := (1,0)^T$ and $e_2 := (0,1)^T$ denote the canonical basis vectors of $\mathbb{R}^2$.
- $R_2(\theta) \in SO(2)$ is the rotation matrix of an angle $\theta \in \mathbb{R}$; $S_2 = R_2(\pi/2)$ is the unitary skew-symmetric matrix.
- given a function of time $f(t)$ the dot notation denotes the time derivative, i.e. $\dot{f} := \frac{\mathrm{d}f}{\mathrm{d}t}$. Higher order derivatives are denoted with a corresponding amount of dots.
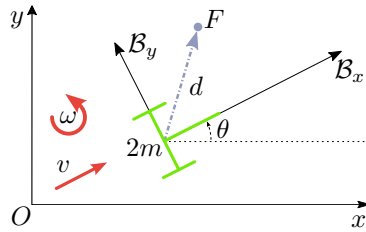

Fig. 1: Notation. The unicycle model is a planar model of a robot having two wheels placed at a distance $2m$, $m \in \mathbb{R}$ with a coinciding rotation axis. Hence, this mobile robot cannot move sideways, i.e. along the wheel axis, but it can turn by moving the wheel at different velocity. $\mathcal{B}$ is a frame attached to the robot whose origin is located in the middle of the wheels axis. Point F is attached to the robot. Its position expressed in $\mathcal{B}$ is given by $d \in \mathbb{R}^2$, a constant vector.

- $\mathbb{1}_n \in \mathbb{R}^{n \times n}$ denotes the identity matrix of dimension n.
- $S(\cdot)$ is the skew-symmetric operation associated with the cross product in $\mathbb{R}^3$.
- The vee operator denoted by $(\cdot)^\vee$ is the inverse of the $S(\cdot)$ operation.
- The Euclidean norm of a vector of coordinates $v \in \mathbb{R}^n$ is denoted by $\|v\|$.
- $^A R_B \in SO(3)$ and $^A T_B \in SE(3)$ denote the rotation and transformation matrices which transform a vector expressed in the B frame into a vector expressed in the A frame.

B. The unicycle model

The unicycle model, represented in Fig. 1. is described by the following model equations [27]:

$$\dot{x} = vR_2(\theta)e_1, \tag{1a}$$
$$\dot{\theta} = \omega, \tag{1b}$$

with $v \in \mathbb{R}$ and $\omega \in \mathbb{R}$ the robot's rolling and rotational velocity, respectively. $x \in \mathbb{R}^2$ is the unicycle position in the inertial frame $\mathcal{I}$, while $\theta \in \mathbb{R}$ represents the angle around the $z-$axis of $\mathcal{I}$ which aligns the inertial reference frame with a unicycle fixed frame. The variables v and ω are considered as kinematic control inputs.

A reasonable control objective for this kind of model is to asymptotically stabilize the point F about a desired point F^* whose position is defined as x_F^*. For this purpose, define the error $\tilde{x}$ as

$$\tilde{x} := x_F - x_F^*. \tag{2}$$

so that the control objective is equivalent to the asymptotic stabilization of $\tilde{x}$ to zero. Since

$$x_F = x + R_2(\theta)d,$$

then by differentiation it yields

$$\dot{x}_F = \dot{x} + \omega R_2(\theta)S_2 d. \tag{3}$$

Eq. (3) describes the output dynamics. Substituting Eq. (1) into Eq. (3), we can rewrite the output dynamics as:

$$\dot{x}_F = B(\theta)u, \tag{4}$$

where $u = [v \ \omega]$ is the vector of control inputs. It is easy to show that $\det B(\theta) = d_1$, which means that when the control point F is not located on the wheels' axis, its stabilization to an arbitrary reference position F^* can be achieved by the use of simple feedback laws. For example, if we define

$$u = B(\theta)^{-1}(\dot{x}_F^* - K\tilde{x}) \tag{5}$$

with K a positive definite matrix, then we have

$$\dot{\tilde{x}} = -K\tilde{x}. \tag{6}$$

Thus, the origin of the error dynamics is an asymptotically stable equilibrium. Notice that this control law is not defined when $d_1 = 0$.

C. Zero Moment Point Preview Control

The MPC controller implemented in this work has been derived from the Zero Moment Point (ZMP) preview control described in [25]. This algorithm adopts a simplified model, viz. the table-cart model, based on the Linear Inverted Pendulum (LIP) approximate model. The humanoid model is assimilated to an inverted pendulum that is linearised around the vertical position. The ZMP can be related to the CoM position and acceleration by the following equation:

$$x_{\text{ZMP}} = P_{2D}x_{\text{CoM}} - \frac{x_{\text{CoM},z}}{g}P_{2D}\ddot{x}_{\text{CoM}} \tag{7}$$

where g is the gravitational constant, $x_{\text{ZMP}} \in \mathbb{R}^2$ denotes the Zero Moment Point (ZMP) [28] coordinate, while $x_{\text{CoM}} \in \mathbb{R}^3$ is the center of mass 3D coordinate. $x_{\text{CoM},z}$, i.e. the CoM height, is assumed constant and P_{2D} is the matrix projecting the CoM on the 2D plane, i.e. it considers only the x and y coordinates of the center of mass.

Assuming to have a desired ZMP trajectory $x_{\text{ZMP}}^*(t)$, we want to track this signal at every time instant. One possibility is to consider the ZMP as an output of following the dynamical system:

$$\dot{\chi} = A_{\text{ZMP}}\chi + B_{\text{ZMP}}u, \quad y = C_{\text{ZMP}}\chi \tag{8}$$

where the new state variable χ and control u are defined as

$$\chi := \begin{bmatrix} x_{\text{CoM},x,y} \\ \dot{x}_{\text{CoM},x,y} \\ \ddot{x}_{\text{CoM},x,y} \end{bmatrix} \in \mathbb{R}^6, \quad u := \begin{bmatrix} \dddot{x}_{\text{CoM},x,y} \end{bmatrix} \in \mathbb{R}^2. \tag{9}$$

The system matrices are defined as in the following:

$$A_{\text{ZMP}} = \begin{bmatrix} 0_{2\times2} & \mathbb{1}_2 & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} & \mathbb{1}_2 \\ 0_{2\times2} & 0_{2\times2} & 0_{2\times2} \end{bmatrix}, \quad B_{\text{ZMP}} = \begin{bmatrix} 0_{4\times2} \\ \mathbb{1}_2 \end{bmatrix}$$

$$C_{\text{ZMP}} = \begin{bmatrix} \mathbb{1}_2 & 0_{2\times2} & -\frac{x_{\text{CoM},z}}{g}\mathbb{1}_2 \end{bmatrix}. \tag{10}$$

Defining a cost function $\mathcal{J}$

$$\mathcal{J} = \int_0^{t_f} \|x_{\text{ZMP}}^* - C_{\text{ZMP}}\chi\|_Q^2 + \|u\|_R^2 \, \mathrm{d}\tau. \tag{11}$$

where Q and R are two weight matrices of suitable dimension. $\mathcal{J}$ penalizes the output tracking error plus a regularization term on the effort. In order to minimize $\mathcal{J}$ given Eq. (8) we can implement a Linear Quadratic controller. This simple controller allows to track a desired ZMP trajectory.

D. System Modeling

The full model of a humanoid robot is composed of $n+1$ rigid bodies, called links, connected by n joints with one degree of freedom each. We also assume that none of the links has an *a priori* constant pose with respect to an inertial frame, i.e. the system is *free floating*.

The robot configuration space can then be characterized by the *position* and the *orientation* of a frame attached to a robot's link, called *base frame* $\mathcal{B}$, and the joint configurations. Thus, the robot configuration space is the group $\mathbb{Q} = \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n$. An element $q \in \mathbb{Q}$ can be defined as the following triplet: $q = ({}^{\mathcal{I}}p_{\mathcal{B}}, {}^{\mathcal{I}}R_{\mathcal{B}}, s)$ where ${}^{\mathcal{I}}p_{\mathcal{B}} \in \mathbb{R}^3$ denotes the position of the base frame with respect to the inertial frame, ${}^{\mathcal{I}}R_{\mathcal{B}} \in \mathbb{R}^{3\times3}$ is a rotation matrix representing the orientation of the *base frame*, and $s \in \mathbb{R}^n$ is the joint configuration.

The *velocity* of the multi-body system can be characterized by the *algebra* $\mathbb{V}$ of $\mathbb{Q}$ defined by: $\mathbb{V} = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^n$. An element of $\mathbb{V}$ is then a triplet $\nu = ({}^{\mathcal{I}}\dot{p}_{\mathcal{B}}, {}^{\mathcal{I}}\omega_{\mathcal{B}}, \dot{s}) = (\mathrm{v}_{\mathcal{B}}, \dot{s})$, where ${}^{\mathcal{I}}\omega_{\mathcal{B}}$ is the angular velocity of the base frame expressed w.r.t. the inertial frame, i.e. ${}^{\mathcal{I}}\dot{R}_{\mathcal{B}} = S({}^{\mathcal{I}}\omega_{\mathcal{B}}){}^{\mathcal{I}}R_{\mathcal{B}}$. A more detailed description of the floating base model is provided in [29].

We also assume that the robot is interacting with the environment exchanging n_c distinct wrenches[1]. The application of the Euler-Poincaré formalism [30, Ch. 13.5]:

$$M(q)\dot{\nu} + C(q,\nu)\nu + G(q) = B\tau + \sum_{k=1}^{n_c} J_{\mathcal{C}_k}^{\top} f_k \tag{12}$$

where $M \in \mathbb{R}^{n+6\times n+6}$ is the mass matrix, $C \in \mathbb{R}^{(n+6)\times(n+6)}$ accounts for Coriolis and centrifugal effects, $G \in \mathbb{R}^{n+6}$ is the gravity term, $B = (0_{n\times6}, \mathbb{1}_n)^{\top}$ is a selector matrix, $\tau \in \mathbb{R}^n$ is a vector representing the internal actuation torques, and $f_k \in \mathbb{R}^6$ denotes the k-th external wrench applied by the environment on the robot. The Jacobian $J_{\mathcal{C}_k} = J_{\mathcal{C}_k}(q)$ is the map between the robot's velocity ν and the linear and angular velocity at the k-th contact link.

Lastly, it is assumed that a set of holonomic constraints acts on System (12). These holonomic constraints are of the form $c(q) = 0$, and may represent, for instance, a frame having a constant pose w.r.t. the inertial frame. In the case this frame corresponds to the location at which a rigid contact occurs on a link, we represent the holonomic constraint as $J_{\mathcal{C}_k}(q)\nu = 0$.

Hence, the holonomic constraints associated with all the rigid contacts can be represented as

$$J(q)\nu = \begin{bmatrix} J_{\mathcal{C}_1}(q) \\ \cdots \\ J_{\mathcal{C}_{n_c}}(q) \end{bmatrix} \nu = \begin{bmatrix} J_b & J_j \end{bmatrix} \nu = J_b\mathrm{v}_{\mathcal{B}} + J_j\dot{s} = 0, \tag{13}$$

with $J_b \in \mathbb{R}^{6n_c\times6}, J_j \in \mathbb{R}^{6n_c\times n}$. The base frame velocity is denoted by $\mathrm{v}_{\mathcal{B}} \in \mathbb{R}^6$.

---

[1]As an abuse of notation, we define as *wrench* a quantity that is not the dual of a *twist*, but a 6D force/moment vector.

III. ARCHITECTURE

In this section we summarise the components constituting the presented architecture, namely:

- the footstep planner,
- the RH controller,
- the stack-of-task-balancing controller.

A. The Footstep Planner

Consider the unicycle model presented in Section II-B. Assuming to know the reference trajectory for point F up to time T, we can integrate the closed-loop system described in Section II-B to obtain the trajectory spanned by the unicycle. The next step consists in discretizing the unicycle trajectories at a fixed rate $1/\mathrm{d}t$. This passage allows us to search for the best foot placement option in a smaller space, constituted by the set of discretization points taken from the original unicycle trajectories.

Given a discrete instant $k \in \mathbb{N}$, we can define $x_k = x(k\mathrm{d}t)$ as the position of the unicycle at instant k, while $\theta_k = \theta(k\mathrm{d}t)$ is its orientation along the $z-$axis. We can use the discretized trajectories to reconstruct the desired position of the left and right foot, x_k^l and x_k^r, using the following relations:

$$x_k^l = x_k + R_2(\theta_k)\begin{bmatrix} 0 \\ m \end{bmatrix}, x_k^r = x_k + R_2(\theta_k)\begin{bmatrix} 0 \\ -m \end{bmatrix} \quad (14a)$$

$$\theta_k^l = \theta_k^r = \theta_k. \quad (14b)$$

A step contains two phases: double support and single support. During double support, both robot feet are on ground and depending on the foot, we can distinguish two different states: *switch-in* if the foot is being loaded, *switch-out* otherwise. In single support instead, a foot is in a *swing* state if it is moving, *stance* otherwise. The instant in which a foot lands on the ground is called *impact time*, t_{imp}^f. After this event, the foot will experience the following sequence of states: *switch-in* $\to$ *stance* $\to$ *switch-out* $\to$ *swing*. This sequence is terminated by an another *impact time*. At the beginning of the *switch-out* phase, the *other* foot has landed on the ground with an impact time $t_{\mathrm{imp}}^{\sim f}$. The step duration, Δ_t is then defined as:

$$\Delta_t = t_{\mathrm{imp}}^{\sim f} - t_{\mathrm{imp}}^f.$$

We define additional quantities relating the two feet when in double support (ds):

- the orientation difference when in double support $\Delta_\theta = |\theta_{ds}^l - \theta_{ds}^r|$.
- The feet distance $\Delta_x = \|x_{ds}^l - x_{ds}^r\|$.
- The position of the left foot expressed on the frame rigidly attached to the right foot, ${}^rP_l$.

These quantities will be used to determine the feet trajectories starting from the unicycle ones.

Given the discretized unicycle trajectories, a possible policy consists in fixing the duration of a step, namely $\Delta_t = $ const, or in fixing its length, i.e. setting Δ_x to a constant. Both these two strategies are not desirable. In the former case the robot will always take (eventually) very short steps at maximum speed. In the latter, if the unicycle is

advancing slowly (because of slow moving references), the robot will take steps always at maximum length sacrificing the walking speed. In view of these considerations, we would like the planner to modify step length and speed depending on the reference trajectory. Since we would like to avoid fixing any variable, it is necessary to define a cost function. It is composed of two parts. The first part weights the squared inverse of Δ_t, thus penalizing fast steps, while the second penalize the squared 2-norm of Δ_x, avoiding to take long steps. Thus, the cost function ϕ can be written as:

$$\phi = k_t \frac{1}{\Delta_t^2} + k_x \|\Delta_x\|^2, \quad (15)$$

where k_t and k_x are two positive numbers. Depending on their ratio, the robot will take long-and-slow or short-and-fast steps. Notice that ϕ is not defined when $\Delta_t = 0$, but the robot would not be able to take steps so fast. Thus, we need to bound Δ_t:

$$t_{min} \leq \Delta_t \leq t_{max} \quad (16)$$

where the upper bound avoids a step to be too slow.

Regarding Δ_x it needs to be lower than an upper-bound d_{max}, bounding the swinging foot into a circular area drawn around the stance foot with radius d_{max}. Another constraint to be considered is the relative rotation of the two feet. In particular the absolute value of Δ_θ must be lower than θ_{max}.

Finally, in order to avoid the robot to twist its legs, we simply resort to a bound on the $y-$component of ${}^rP_l$ vector. Indeed, it corresponds to the width of the step. By applying a lower-bound w_{min} on it, we avoid the left foot to be on the right of the other foot.

Finally we can write the constraints defined above, together with ϕ, as an optimization problem:

$$\underset{t_{imp}}{\text{minimize}} \quad K_t \frac{1}{\Delta_t^2} + K_x \|\Delta_x\|^2 \quad (17a)$$

$$\text{s.t.} : \quad t_{min} \leq \quad \Delta_t \quad \leq t_{max} \quad (17b)$$

$$\Delta_x \quad \leq d_{max} \quad (17c)$$

$$|\Delta_\theta| \quad \leq \theta_{max} \quad (17d)$$

$$w_{min} \leq {}^rP_{l,y}. \quad (17e)$$

The decision variables are the *impact times*. If we select an instant k to be the *impact time* for a foot, then we can obtain the corresponding foot position and orientation using Eq. (14) and Eq. (15).

The solution of the optimization problem is obtained through a simple iterative algorithm. The initialization can be done by using the measured position of the feet. Starting from this configuration we can integrate the unicycle trajectories assuming to know the reference trajectories. Once we discretize them, we can iterate over k until we find a set of points which satisfy the conditions defined above. Among the remaining points we can evaluate ϕ to find our optimum.

Once we planned footsteps for the full time horizon, we can proceed in interpolating the feet trajectories and other relevant quantities for the walking motion, e.g. the Zero Moment Point [28].

While walking, references may change and it is not desirable to wait the end of the planned trajectories before changing them. Thus the idea is to "merge" two trajectories instead of serializing them. When generating a new trajectory, the robot is supposed to be standing on two feet and the first *switch* needs to last half of its normal time. In view of these considerations, the middle instant of the double support phase is a particularly suitable point to merge two trajectories. The choice of this point eases the merging process since the feet are not supposed to move in that instant, hence the initialization of these trajectories does not need to know their initial desired velocity and/or acceleration. Notice that there may be different *merge points* along the trajectories, depending on the number of (full) double support phases. Two trivial *merge points* are the very first instant of the trajectories themselves and the final point (serialization case).

This method can be assimilated to the *Receding Horizon Principle* [31]. In fact, we plan trajectories for an horizon T but only a portion of them will be actually used, namely the first generated step. This simple strategy allows us to change the unicycle reference trajectory on-line (through the use of a joystick for example) directly affecting the robot motion. In addition, we could correct the new trajectories with the *actual* position of the feet. This is particularly suitable when the foot placement is not perfect, as in torque-controlled walking.

B. The Receding Horizon Controller

The receding horizon controller used in our architecture inherits from the basic formulation described in Sec. II-C. Nevertheless, differently from [25], we have added time-varying constraints to bound the ZMP inside the convex hull given by the supporting foot or feet. In this way we increase the robustness of the controller, avoiding overshoots that may cause the robot to tip around one of the foot edges. These constraints are modeled as linear inequalities acting on the state variables, i.e.

$$Z(t)\chi - z(t) \leq 0. \tag{18}$$

Note that the constraint matrix and vector do depend on the time. In particular, we can exploit the knowledge on the desired feet positions to constraint the ZMP throughout the *whole* horizon, while retaining linearity

To summarize, the full optimal control problem can be represented by the following minimisation problem:

$$\min_{\chi,u} \int_0^{t_f} \|x_{\mathrm{ZMP}}^* - C_{\mathrm{ZMP}}\chi\|_Q^2 + \|u\|_R^2 \, \mathrm{d}\tau.$$
$$\text{s.t. } \dot{\chi} = A\chi + Bu, \quad \forall t \in [t_0, t_f) \tag{19}$$
$$Z(t)\chi - z(t) \leq 0$$
$$\chi(0) = \bar{\chi}.$$

The problem in Eq. (19) is solved at each control sampling time by means of a Direct Multiple Shooting method [32], [33]. This choice has been driven by the necessity of formulating state constraints, Eq. (18), throughout the whole horizon.

Applying the Receding Horizon Principle, only the first output of the MPC controller is used. Since the control input is the CoM jerk, we integrate it in order to obtain a desired CoM velocity and position. The latter is also sent to the inverse kinematics (IK) library [34, InverseKinematics sub-library], together with the desired feet positions to retrieve the desired joints coordinates. Both the MPC an the IK modules are implemented using IPOpt [35].

C. The Stack of Tasks Balancing Controller

The balancing controller has as objective the stabilization of the center of mass position, the left and right feet positions and orientations by defining joint torques. The velocities associated to these tasks are stacked together into Υ:

$$\Upsilon = \begin{bmatrix} \dot{p}_G^\top & \mathrm{v}_L^\top & \mathrm{v}_R^\top \end{bmatrix}^\top \tag{20}$$

where $\dot{p}_G \in \mathbb{R}^3$ is the linear velocity of the center of mass, frame, $\mathrm{v}_L \in \mathbb{R}^6$ and $\mathrm{v}_R \in \mathbb{R}^6$ are vectors of linear and angular velocities of the frames attached to the left and right feet.

Letting J_G, J_L and J_R denote respectively the Jacobians of the center of mass position, left and right foot configurations, J_Υ can be defined as a stack of the Jacobians associated to each task:

$$J_\Upsilon = \begin{bmatrix} J_G^\top & J_L^\top & J_R^\top \end{bmatrix}^\top \tag{21}$$

Furthermore, the task velocities Υ can be computed from ν using $\Upsilon = J_\Upsilon \nu$. By deriving this expression, the task acceleration is

$$\dot{\Upsilon} = \dot{J}_\Upsilon \nu + J_\Upsilon \dot{\nu} \tag{22}$$

In view of (12) and (22), the task accelerations $\dot{\Upsilon}$ can be formulated as a function of the control input u composed of joint torques τ and contact wrenches f_k:

$$\dot{\Upsilon}(u) = \dot{J}_\Upsilon \nu + J_\Upsilon M^{-1}(B\tau + \sum_{k=1}^{n_c} J_{\mathcal{C}_k}^\top f_k - h). \tag{23}$$

In our formulation we want $\dot{\Upsilon}(u)$ to track a specified task acceleration $\dot{\Upsilon}^*$, namely:

$$\dot{\Upsilon}(u) = \dot{\Upsilon}^*. \tag{24}$$

The reference accelerations $\dot{\Upsilon}^*$ are computed using a simple PD control strategy for what concerns the linear terms. In parallel, the rotational terms are obtained by adopting a PD controller on $SO(3)$ (see [36, Section 5.11.6, p.173]), thus avoiding to use a parametrization like Euler angles.

While the task defined above can be considered as *high priority* tasks, we may conceive other possible objectives to shape the robot motion. One example regards the torso orientation, which we would like to keep in an upright posture. Similarly to what have been done above, we have $\mathcal{T} = J_\mathcal{T}\nu$ with $J_\mathcal{T}$ the torso Jacobian. By differentiation we would obtain a result similar to Eq. (23). Since this task is considered as *low priority*, we are interested in minimizing

the squared Euclidean distance of $\dot{\mathcal{T}}(u)$ from a desired value $\dot{\mathcal{T}}^*$, i.e

$$\min_{\tau, f_k} \frac{1}{2} \|\dot{\mathcal{T}}(u) - \dot{\mathcal{T}}^*\|^2. \tag{25}$$

The reference acceleration $\dot{\mathcal{T}}^*$ is obtained through a PD controller on $SO(3)$.

Finally, we also added another lower priority *postural* task to track joint configurations. Similarly as before, the joint accelerations $\ddot{s}(u)$, which depends upon the control inputs through the dynamics equations Eq. (12), are kept close to a reference $\ddot{s}^*$:

$$\min_{\tau, f_k} \frac{1}{2} \|\ddot{s}(u) - \ddot{s}^*\|^2. \tag{26}$$

The postural desired accelerations are defined through a *PD+gravity compensation* control law, [23].

Considering the contact wrenches f_k as control inputs, we need to ensure their actual feasibility given the contacts. They are modeled as unilateral, with a limited friction coefficient. An additional condition regards the Center of Pressure, CoP, which needs to lie inside the contact patch. All these conditions can be grouped in a set of inequality constraints applied to the contact wrenches

$$Cf_k \leq b \quad \forall k \leq n_c. \tag{27}$$

Finally, we can group Eq.s (24)−(27) in the following QP formulation:

$$\min \tau, f_k \qquad \Gamma \tag{28a}$$
$$\text{s.t.} : \dot{\Upsilon}(u) = \dot{\Upsilon}^* \tag{28b}$$
$$Cf_k \leq b \quad \forall k \leq n_c \tag{28c}$$

where $\Gamma = \frac{1}{2}\|\ddot{s}(u) - \ddot{s}^*\|^2 + \frac{w_{\mathcal{T}}}{2}\|\dot{\mathcal{T}}(u) - \dot{\mathcal{T}}^*\|^2 + \frac{w_\tau}{2}\|\tau\|^2$. With respect to Eq. (25) and (26), an additional term is inserted, namely the 2-norm of the joint torques, which serves as a regularization. Among several feasible solutions, we are mostly interested in the one which requires the least effort. By changing the weights w we can tune the relative importance of each term.

Let us focus on the feet contact wrenches f_l and f_r. During the *switch* phases we expect one of the two wrenches to vanish in order to smoothly deactivate the corresponding contact. In order to ease this process we added a cost term in Γ, equal to $\frac{w_f}{2}\left(\mathcal{F}_r\|f_l\| + \mathcal{F}_l\|f_r\|\right)$ where $\mathcal{F}_r$ (respectively $\mathcal{F}_l$) is the normalized load that the right (respectively left) foot is supposed to carry. It is 1 when the corresponding foot is supposed to hold the full weight of the robot, 0 when unloaded. This information is provided by the planner described in Sec III-A. The QP problem of Eq. (28) is solved at a rate of 100Hz through qpOASES [37].

IV. EXPERIMENTS

The presented architecture is composed by three different parts. In this section, we are going to show three different experiments whose aim is to test each part in an incremental way. All the experiments have been performed on the iCub robot, controlling 23 Dofs either in position or in torque control. The complete experiments videos are available as multimedia attachements to this paper
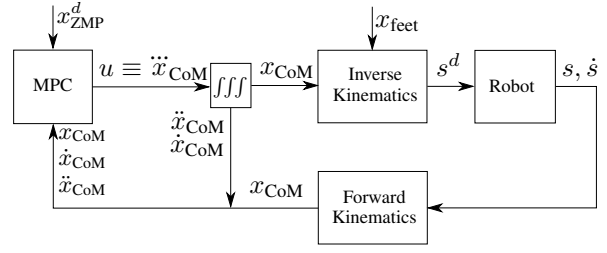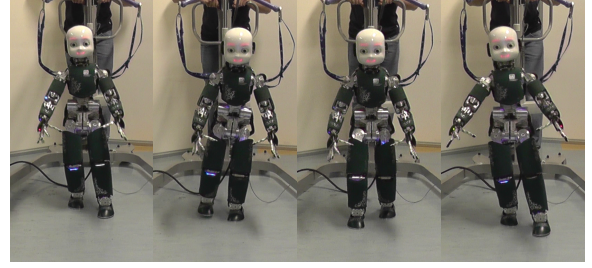

Fig. 2: A first skeleton of the architecture, composed only by the MPC (RH) controller connected to the inverse kinematics (IK). Their output is directly sent to the robot.

(a) $t = t_0$ (b) $t = t_0 + 1s$ (c) $t = t_0 + 2s$ (d) $t = t_0 + 3s$

Fig. 3: Snapshots extracted from the accompanying video. The robot is walking straight in position control.

A. *Test of the RH controller in position control*

In the first experiment we used the RH module to control the robot joints in *position control*. As depicted in the architecture of Figure 2 the control loop is closed on the CoM position only, avoiding to stream in the controller noisy measurements like joints velocities. The desired feet trajectories x_{feet} and the desired ZMP profile x_{ZMP}^d are obtained by an offline planner described in [38].

Both the RHC and the IK are running on the iCub head, which is equipped with a 4th generation Intel® Core i7@1.7GHz and 8GB of RAM. The whole architecture takes in average less than 8ms for a control loop.

The robot is taking steps of 14cm long in $1.25s$ (of which $0.65s$ in double support). Fig. 3 presents some snapshots of the robot while walking.

B. *Adding the unicycle planner*

The controller has been improved by connecting the unicycle planner, as shown in Fig. 4. This allow us to adapt the robot walking direction in an *online* fashion. As an example, by using a joystick we can drive a reference position x_F^* for the point F attached to the unicycle. Depending on how much we press the joypad, this point moves further away from the robot.

Using this planner, the step timings and dimensions are not fixed a priori. In this particular experiment, a single step could last between $1.3s$ and $5.0s$, while the swing foot can land in a radius of $0.175m$ from the stance foot. Fig. 5 presents some snapshots of the robot while negotiating a right turn.
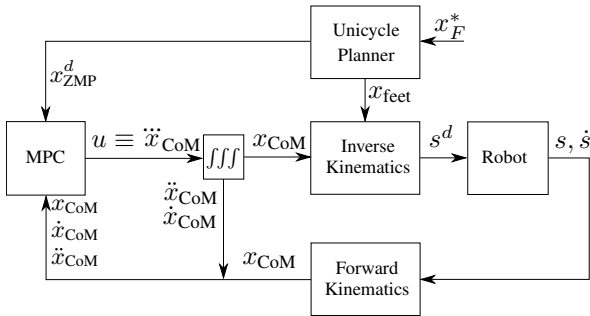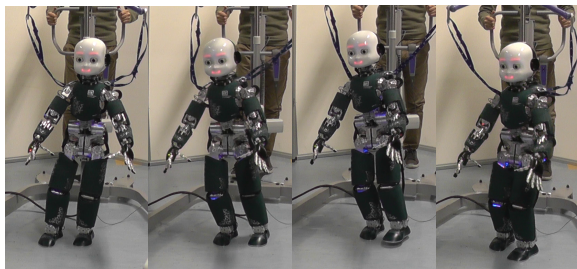
Fig. 4: The scheme of Figure 2 has been upgrade with the unicycle planner which is responsible of providing online references.

(a) $t = t_0$ (b) $t = t_0 + 1s$ (c) $t = t_0 + 2s$ (d) $t = t_0 + 3s$

Fig. 5: Thanks to the unicycle planner, the robot walks while turning right.

C. Complete Architecture

Finally, we plugged also the task based balancing controller presented in Sec. III-C. The overall architecture is depicted in Figure 6, highlighting that the stack of task balancing controller is now in charge of sending commands the robot. In order to draw comparisons with the first experiments, we followed again a straight trajectory. Even if the task is similar, it is necessary to use the Unicycle Planner now in order to cope with feet positioning errors (see Fig. 8). In fact, trajectories can be updated in order to take into account possible feet misplacements. Fig. 7 shows the CoM tracking capabilities of the presented balancing controller. During this experiment, the minimum stepping time had to be increased to $3s$ (the maximum is still $5s$), while maintaining the same maximum step length of the previous experiment. Even if we had to slow down a bit the walking speed, the results are still promising. While walking, the robot is much smoother in its motion, reducing the probabilities of falling in case of an unforeseen disturbances.

V. CONCLUSIONS

This architecture is able to cope with variable walking speed and advance while turning while keeping certain degree of compliance to better adapt to the *online* changes of desired reference trajectories. Focusing on torque control, experiments have shown some bottlenecks that, for the time being, prevented to achieve higher performances. These issue are mainly related to the estimation of joint torques
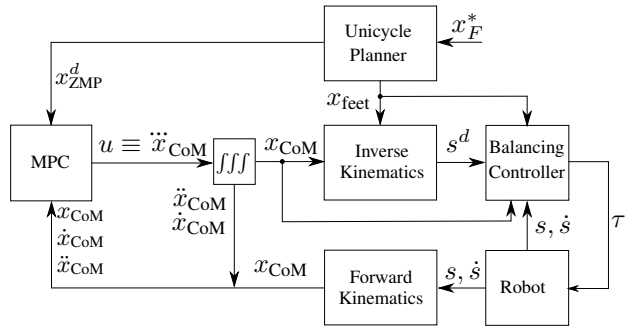

Fig. 6: The complete architecture. The output of the IK is no longer sent to the robot, but to the stack-of-task balancing controller, together with the desired CoM and feet position. Joint positions and velocities are taken as feedback from the robot.
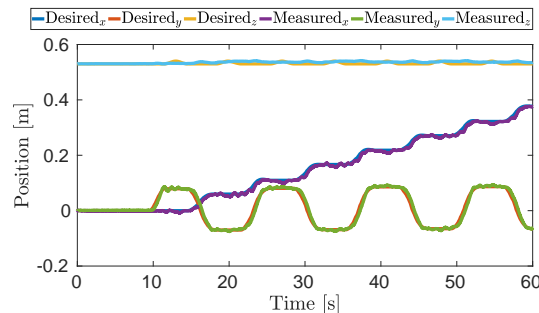

Fig. 7: Center of Mass (CoM) tracking when taking some steps in torque control. The quantities are expressed in an inertial frame $\mathcal{I}$

(limiting the performances of the joint torque controller) and to the floating base estimation. As a future work, we plan to increase the robustness of the architecture against these uncertainties.

Even if the results presented on this paper have to be considered as preliminary, they enlighten the flexibility properties of the proposed architecture. The result on torque control walking is promising, since its inherent compliance can increase the robustness of the walking motion, against, for example, un-modeled ground slope variations.

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] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 2016, pp. 3555–3561.

[5] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on.*
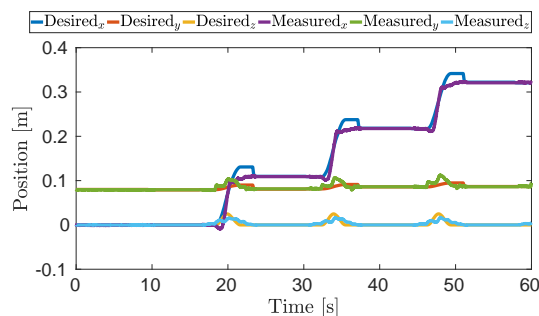
Fig. 8: Tracking of the left foot positions. Notice that after the step is taken, the *desired* values are updated according to the measured position.

[6] 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.

[7] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotionan inverse optimal control approach," *Autonomous robots*, vol. 28, no. 3, pp. 369–383, 2010.

[8] M. L. Handford and M. Srinivasan, "Sideways walking: preferred is slow, slow is optimal, and optimal is expensive," *Biology letters*, vol. 10, no. 1, p. 20131006, 2014.

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

[10] A. Faragasso, G. Oriolo, A. Paolillo, and M. Vendittelli, "Vision-based corridor navigation for humanoid robots," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 3190–3195.

[11] M. Cognetti, D. De Simone, L. Lanari, and G. Oriolo, "Real-time planning and execution of evasive motions for a humanoid robot," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4200–4206.

[12] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789 – 814, 2000.

[13] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation," in *Intelligent Robots and Systems, 2001. Proceedings. IEEE/RSJ International Conference on*, pp. 239–246.

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

[15] 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.

[16] M. Missura and S. Behnke, "Balanced walking with capture steps," in *Robot Soccer World Cup*. Springer, 2014, pp. 3–15.

[17] M. Bombile and A. Billard, "Capture-point based balance and reactive omnidirectional walking controller," in *IEEE RAS International Conference on Humanoid Robots*, no. EPFL-CONF-231920, 2017.

[18] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Souères, and J. Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, April 2013.

[19] C. Ott, M. A. Roa, and G. Hirzinger, "Posture and balance control for biped robots based on contact force optimization," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pp. 26–33.

[20] 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.

[21] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 981–988.

[22] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, vol. 33, no. 4, pp. 399–414, Nov 2012.

[23] 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.

[24] 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)*, pp. 141–141.

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

[26] 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.

[27] D. Pucci, L. Marchetti, and P. Morin, "Nonlinear control of unicycle-like robots for person following," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 3406–3411.

[28] M. Vukobratović and B. Borovac, "Zero-moment point: thirty five years of its life," *International journal of humanoid robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[29] S. Traversaro, D. Pucci, and F. Nori, "A unified view of the equations of motion used for control design of humanoid robots," *On line*, 2017. [Online]. Available: https://traversaro.github.io/preprints/changebase.pdf

[30] J. E. Marsden and T. S. Ratiu, *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Springer Publishing Company, Incorporated, 2010.

[31] H. Michalska and D. Q. Mayne, "Receding horizon control of nonlinear systems," in *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*. IEEE, 1989, pp. 107–108.

[32] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603 – 1608, 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.

[33] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast motions in biomechanics and robotics*. Springer Berlin Heidelberg, 2006, pp. 65–93.

[34] Dynamic Interaction Control Lab - Istituto Italiano di Tecnologia, "iDynTree library," https://github.com/robotology/idyntree, 2016.

[35] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57.

[36] R. Olfati-Saber, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[37] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[38] Y. Hu, J. Eljaik, K. Stein, F. Nori, and K. Mombaur, "Walking of the icub humanoid robot in different scenarios: implementation and performance analysis," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pp. 690–696.