An Augmented Kinematic Model for the Cartesian Control of the Hybrid Wheeled-Legged Quadrupedal Robot CENTAURO

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

***Abstract*— This work deals with the kinematic control of *Centauro*, a highly redundant, hybrid wheeled-legged robot designed at *Istituto Italiano di Tecnologia (IIT)*. Given its full wheeled mobility as allowed by its four independently steerable wheels, the choice of some local frame (in addition to the global world) is required in order to express tasks that are naturally defined in a robot-centric fashion. In this work, we show that trivially selecting such a frame as the robot trunk leads to sub-optimal results in terms of motion capabilities; as main contribution, we therefore propose a comparative analysis among three different choices of local frame, and demonstrate that in order to retain all advantages from the whole-body control domain, the kinematic model of the robot must be augmented with an additional virtual frame, which proves to be a useful choice of local frame, enabling e.g. the automatic adaptation of the trunk posture under constraint activation. The resulting Cartesian controller is finally validated by means of an extensive experimental session on the real hardware.**

I. Introduction

At the time of writing, the robotic technology is mature enough for performing useful work in a reliable way inside simple environments. On the other hand, operation inside unstructured scenarios is essentially dominated by overly-simplified machines, such as small tracked vehicles, which trade simplicity of their control (and reliability) for a lack of flexibility in the tasks that they are able to accomplish. The *Centauro* robot [1] (Figure 1), i.e. the main target platform of this work, has been designed with the aim to take a step forward in the direction of versatility, combining powerful manipulation capabilities with a more reliable quadrupedal hybrid wheeled-legged locomotion concept. Because of its peculiar kinematic structure, Centauro provides significant flexibility in terms of motion control: thanks to the actively steerable wheels, in-place manipulation and stepping motions can be combined with modulations of the support polygon, and driving motions of the whole robot. Broadly speaking, a motion controller which completely exploits the aforementioned kinematic capabilities should provide full control of any task of interest (e.g. gaze, pose control of the trunk, end-effectors, wheels, ...) both with respect to a global world frame, and with respect to a *local frame*, depending on the nature of the specific task at hand. Notice that up to this point, the notion of *local frame* is left rather unspecified, and is just to be regarded as some frame which moves rigidly with the robot support polygon; it will however be a key concept in this work.

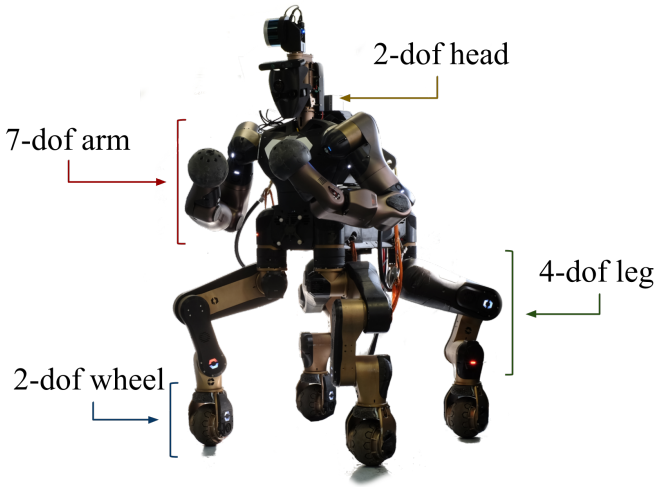

Fig. 1. Kinematics of the Centauro robot. Each of its four legs is actuated by six motors: the first four provide control over the wheel position and pitch orientation, whereas the last two joints permit to rotate the wheel about both its spinning and steering axes. Centauro is also equipped with a humanoid upper body, that is made of two arms of seven degrees-of-freedom each, plus a yaw joint connecting the trunk of the robot to its torso. Finally, a stereo camera is mounted on the neck; two motors actuate the camera about its pan and tilt axes.

Existing works on similar platforms (see e.g. [2]) circumvent this aspect by carefully crafting of all end-effectors desired poses with respect to (w.r.t.) the robot trunk frame such that the robot performs a desired (local or global) task; then, such poses are tracked by a chain-based inverse-kinematics solver. Despite the ability of such a strategy to generate the required motions, it is the authors' belief that significant advantages can be obtained by employing a floating-base formulation for the robot model, and by exploiting techniques from the whole-body control domain. The most prominent advantage lies in the structural enforcement of the required relationships between frames directly through the task matrices (relative *Jacobians*), which makes the controller less reliant on feedback gains, more robust in the presence of constraint, and permits to directly specify reference values for each task in their natural coordinate frame.

When applying such concepts to the motion control problem of the Centauro robot, the key question naturally arises of which frame should play the role of *local* frame. Intuition suggests that such a frame should "travel with the robot", and a natural choice would be to select the trunk frame.

[1] The authors are with the Humanoids & Human Centered Mechatronics lab (HHCM), Istituto Italiano di Tecnologia (IIT), Genova, Italy {name.surname}@iit.it

[2] A. Laurenzi is with Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), Universitá di Genova, Italy

However, it will be shown in this work how this choice falls short of exploiting the versatility of our target platform. The key observation in our work is that, if a standard floating base model is employed, such a frame does not actually exist: in order for the robot to fully exploit its kinematic potential in a whole-body manner, *additional degrees-of-freedom must be added to the robot kinematic model*. The use of additional, fictitious degrees-of-freedom has already been exploited in robotics. Beside being extensively used in order to model floating-base systems in a uniform way w.r.t. fixed based manipulators, such a notion has received attention in specific domains such as robotic grasping [3]–[5], and orbital manipulation [6]. Inspired by such works, we propose to extend this concept in the field of wheeled locomotion, to provide a choice of local frame which permits to completely exploit the robot kinematic capabilities; to the best of the authors' knowledge has not been attempted before.

In addition to [2], other works have previously addressed the motion control problem of articulated wheeled robots. In [7], which targets the same robotic platform of this work, the authors analyze a wheel-ground contact model under toroidal wheel shape assumptions, and then present a controller for driving motion and support polygon regulation; the work of [8] integrates wheeled contacts into the torque-based controller for the *ANYmal* robot, yielding effective compliant adaptation to terrain roughness. However, both approaches are not concerned with manipulation capabilities. In [9], a trajectory optimization based approach to the design and control of wheeled-legged robots is discussed; even though impressive, highly-dynamic motions are generated, the proposed strategy is too costly for online (e.g. receding horizon) applications. Finally, applications on the stabilization of active suspensions rovers are presented in [10], [11], with an emphasis on terrain adaptation capabilities and tip-over avoidance in rough terrains.

In this work, we propose a Cartesian control framework that permits the full control of the Centauro robot in an *online* fashion as specified in terms of a list of *motion requirements* to be defined in Section III. These requirements specify desired relationships *between frames*, which we manage to enforce by writing suitable relative Cartesian tasks inside a floating-base formulation. To achieve this, a novel kinematic model of the robot must be introduced, allowing to define a local reference frame which is *completely decoupled* from the waist frame. As main advantage, the additional degrees of freedom can be exploited by the whole-body solver with the benefit of enabling automatic adaptation of the trunk pose when a constraint on some chain is activated.

II. Background on Cartesian control

The aim of this section is to introduce the mathematical notation which is used throughout the paper, and to provide background knowledge on the modeling and Cartesian control of highly redundant mechanical systems as the Centauro robot.

A. Kinematic modeling

Being Centauro a legged robot, its configuration space is given not only by its actuated joint positions, but also by the pose of one of its links, which is called the *floating base*. Although such a link can be chosen arbitrarily, it is a natural choice to select the trunk as the floating base. Consequently, let $\boldsymbol{q} \in \mathbb{R}^n$ denote the full configuration vector for the robot. We obtain such a vector by joining the actuated joint configuration vector $\boldsymbol{q}_a \in \mathbb{R}^{n_a}$ with a minimal representation[1] of the floating base pose $\boldsymbol{q}_{\text{fb}} \in \mathbb{R}^6$, as follows:

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_{\text{fb}} \\ \boldsymbol{q}_a \end{bmatrix}. \tag{1}$$

Then, let $\boldsymbol{x}$ denote a Cartesian task of interest (e.g. the pose of one of the robot links), which is dependent on the robot configuration though some non-linear mapping $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$:

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}); \tag{2}$$

by differentiating (2), the task velocity can be computed as

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3}$$

where $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ is the task *Jacobian matrix*. Notice that, with an abuse of notation, we identify the quantity $\dot{\boldsymbol{x}}$ with the task velocity twist

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}, \tag{4}$$

so that $\boldsymbol{J}$ is the *geometric* Jacobian.

B. Hierarchical inverse kinematics

Given some desired task velocity $\dot{\boldsymbol{x}}_{\text{d}}$, the required joint-space velocity that permits to achieve it is given by the inversion of the differential kinematics equation (3). Whenever the robot is redundant w.r.t. the given task, i.e. $n > m$, pseudo-inverse based solutions can be employed (see e.g. [12]), allowing to optimize a low-priority objective in the null-space of the main task. Furthermore, when the desired value $\boldsymbol{x}_{\text{ref}}$ is available directly at the task level, good tracking performance can be achieved with the following proportional law[2]

$$\dot{\boldsymbol{x}}_{\text{d}} = \dot{\boldsymbol{x}}_{\text{ref}} + \lambda \left(\boldsymbol{x}_{\text{ref}} - \boldsymbol{x} \right), \tag{5}$$

where λ is a positive-definite matrix gain.

However, when the robot is *hyper-redundant* w.r.t. the single task (i.e. $n \gg m$) as it is the case for multi-chained platforms like Centauro, many tasks are usually simultaneously defined (e.g. manipulation, balancing, ground contact, gazing, ...). Such simple strategies are unable to properly cope with a multi-task scenario; indeed, avoiding unpredictable conflicts between multiple objectives usually requires the definition of more than two priority levels.

[1] For the sake of simplicity, we use here a minimal representation of a rigid body orientation (e.g. Euler angles). A quaternion-based formulation can be employed whenever a singularity free representation is required.

[2] To keep the notation clean, we use the minus "-" operator to indicate a task-space error; the reader should however be aware that errors on the SO(3) manifold require special care, see e.g. [13].

TABLE I

COMPARISON BETWEEN DIFFERENT PROPOSED STRATEGIES FOR THE MOTION CONTROL OF THE CENTAURO ROBOT (R = ROLL AXIS, P = PITCH AXIS).

Local frame \ Motion	M1	M2	M3	M4
Trunk frame	✓	✗	✓	✓
Horizontal trunk frame	✓	R,P	✓	✓
Virtual frame	✓	✓	✓	✓

Hierarchical inverse kinematics (HIK) schemes have been explored in the literature, as done e.g. in [14], where an approach based on null-space projectors is presented. The present work builds on a quadratic programming (QP) based formulation of the HIK problem (HQP) [15], which has the advantage of also handling inequality constraints.

III. CONTROLLER DESIGN

As mentioned in the introductory section, a complete motion controller for a hybrid wheeled-legged robot such as Centauro must provide the freedom to combine local tasks with global tasks. We now detail this rather generic statement into a list of *motion requirements*, which different controller designs will then be compared against. According to such a list, the Centauro robot should be able to:

- M1: move as a whole w.r.t. a global world frame by appropriately steering and rolling the wheels (*driving* motion);
- M2: while driving, adjust the trunk pose w.r.t. a local frame (e.g. to lift one foot);
- M3: while driving, reshape the support polygon by shifting the wheels position w.r.t. a local frame (e.g. to pass through a narrow passage);
- M4: perform a manipulation task w.r.t. to either a local or global world frame, while shifting the support polygon using the wheels (e.g. to improve stability);

In the course of this section, we incrementally build our proposed approach to the motion controller design, that is presented in Section III-C, starting from the naive strategy of Section III-A. Results of each iteration, as compared to the defined motion requirements, are summarized in Table I, and a thorough discussion on the advantages given by our contribution is presented in Section III-D. Note that our results still hold, even in the case that the target platform is not equipped with independently steerable wheels, provided that a higher-level planning stage properly takes into account the specific system under-actuation when generating reference signals [16]. However, for the sake of this presentation, we will assume that the robot wheels can be steered and rolled in a way such that their slippage (i.e. the relative velocity between the contact point and the ground) is zero, whenever this is physically feasible. An algorithmic way to ensure this behavior will be presented in Section IV.

A. Trunk-based control

A first approach to the wheeled motion control of Centauro is to combine *relative* Cartesian tasks between the four wheels and the trunk, with another one controlling the trunk pose w.r.t. a global world frame.

Relative control between frames can be enforced at the Jacobian level as follows. First, let the label *"d"* denote the distal frame, and *"b"* the base frame; then, the relative velocity twist between the two is given by the following expression:

$$\begin{cases} \boldsymbol{v}_{\text{rel}} = \boldsymbol{v}_d - \left(\boldsymbol{v}_b + \boldsymbol{\omega}_b \times \boldsymbol{p}_d^b\right) \\ \boldsymbol{\omega}_{\text{rel}} = \boldsymbol{\omega}_d - \boldsymbol{\omega}_b \end{cases}, \tag{6}$$

where the cross-product term in (6) takes into account the motion of the distal frame as seen from the base frame. In matrix form, equation (6) reads

$$\dot{\boldsymbol{x}}_{\text{rel}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & -\boldsymbol{I}_{3\times3} & \boldsymbol{S}(\boldsymbol{p}_d^b) \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & -\boldsymbol{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}_d \\ \dot{\boldsymbol{x}}_b \end{bmatrix}, \tag{7}$$

where $\boldsymbol{S} \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix such that $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b} \quad \forall \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$. Introducing the distal and base Jacobians w.r.t. the world frame $\boldsymbol{J}_d$ and $\boldsymbol{J}_b$, such that $\dot{\boldsymbol{x}}_d = \boldsymbol{J}_d\,\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{x}}_b = \boldsymbol{J}_b\,\dot{\boldsymbol{q}}$, the relative Jacobian is given by

$$\boldsymbol{J}_{\text{rel}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & -\boldsymbol{I}_{3\times3} & \boldsymbol{S}(\boldsymbol{p}_d^b) \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & -\boldsymbol{I}_{3\times3} \end{bmatrix} \begin{bmatrix} \boldsymbol{J}_d \\ \boldsymbol{J}_b \end{bmatrix}. \tag{8}$$

According to the *math-of-tasks* formalism defined in [17], a possible stack-of-task implementing this solution is the following:

$$\begin{pmatrix} \left(\sum_i {}^{\text{Trunk}}\mathcal{T}_{\text{Wheel}_i}^{[\text{XYZ}]} + {}^{\text{World}}\mathcal{T}_{\text{Trunk}}\right) / \\ \left({}^{[\cdot]}\mathcal{T}_{\text{Hands}} + \sum_i {}^{\text{World}}\mathcal{T}_{\text{Ankle}_i}^{[\text{RPY}]}\right) / \\ \left(\mathcal{T}_{\text{Posture}}\right) \end{pmatrix} << \left(\mathcal{C}_{\substack{\text{Pos.}\\\text{Lims}}} + \mathcal{C}_{\substack{\text{Vel.}\\\text{Lims}}}\right); \tag{9}$$

in the previous equation, ${}^{\text{A}}\mathcal{T}_{\text{B}}$ denotes a Cartesian task of the frame B relative to the frame A. The base frame for end-effectors is left unspecified, as it is dynamically changed from *trunk* to *world* depending on the task. Finally, the plus operator "+" indicates *aggregation* between tasks, whereas the slash " / " sets the left hand side task at a *higher priority* w.r.t. the right hand side task, while the $<<$ symbol is used to specify constraints.

With such a stack, and also assuming that some controller continuously steers and spins each wheel so that it does not slip, the robot base can be conveniently moved around the world frame by just setting its desired Cartesian velocity or pose. Moreover, the robot support polygon can be reshaped by simply setting appropriate desired poses of the wheels w.r.t. the base. However, this scheme cannot handle any roll or pitch motion commanded to the base; indeed, when the base is commanded e.g. to roll as depicted in Figure 2, the imposed relative task causes the wheels to follow such a motion, leading to wheel-ground contact break. Under such circumstance, we obtain a physically unfeasible robot motion.

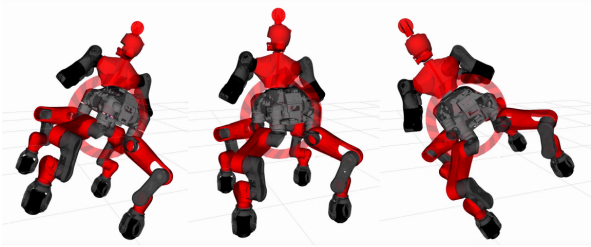

Fig. 2. Kinematic behavior of the Centauro robot while a rolling motion is commanded to it base. The Cartesian pose of the wheels is controlled w.r.t. to the trunk frame (*trunk-based control)*. The obtained motion is not physically feasible.

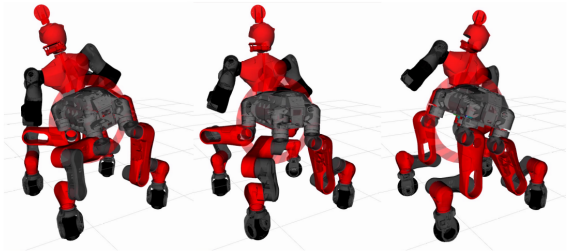

Fig. 3. Kinematic behavior of the Centauro robot while a rolling motion is commanded to it base. The Cartesian pose of the wheels is controlled w.r.t. to a horizontal frame that is attached to the trunk (*horizontal frame-based control)*.

B. Horizontal frame-based control

In order to adapt the scheme of Section III-A so that it can handle rolling and pitching of the base without breaking contact with the ground, we need to introduce a special frame of reference, which we call *horizontal trunk frame (HF)*, with the following three properties:

1) the horizontal trunk frame origin coincides with the trunk frame origin;
2) the projection on the horizontal plane of the trunk frame forward axis (x-axis) coincides with the projection of the horizontal trunk frame forward axis;
3) the horizontal trunk frame vertical axis (z-axis) coincides with the world frame vertical axis.

This idea is inspired by [18], in which a similar frame was adopted in the context of quadrupedal trotting. The reason why such a frame is a good candidate to act as the robot *local frame* lies in the fact that its pose matches the trunk pose for what concerns its position and heading, while being unaffected by rolling or pitching of the base. Thanks to this property, using the horizontal trunk frame as base frame for the wheels tasks can solve the contact-braking issue of the trunk-based approach; as it is shown in Figure 3, when the base is commanded to follow the same rolling motion as in Section III-A, the horizontal frame does not move, and the wheel-ground contact is retained.

In order to implement the required Cartesian task w.r.t. the horizontal trunk frame, it is enough to compute its Jacobian, and then apply (8). In order to do so, let us observe that such a frame coincides with the trunk frame, except that it does not move about its roll and pitch axes. Consequently, its jacobian $\boldsymbol{J}_h$ is given by the following expression:

$$\boldsymbol{J}_h = \mathrm{diag}(1,1,0,0,0,1)\,\boldsymbol{J}_{\text{trunk}}. \tag{10}$$

A possible stack of task leveraging the horizontal frame is given in the following equation:

$$\begin{pmatrix} \left(\sum_i {}^{\text{HF}}\mathcal{T}^{[\text{XYZ}]}_{\text{Wheel}_i} + {}^{\text{World}}\mathcal{T}^{[\text{XY,Yaw}]}_{\text{HF}}\right) / \\ \left({}^{\text{HF}}\mathcal{T}^{[\text{Z,RP}]}_{\text{Trunk}} + {}^{[\cdot]}\mathcal{T}_{\text{Hands}} + \sum_i {}^{\text{World}}\mathcal{T}^{[\text{RPY}]}_{\text{Ankle}_i}\right) / \\ \left(\mathcal{T}_{\text{Posture}}\right) \end{pmatrix} << \begin{pmatrix} \mathcal{C}^{\text{Pos.}}_{\text{Lims}} + \mathcal{C}^{\text{Vel.}}_{\text{Lims}} \end{pmatrix}. \tag{11}$$

C. Virtual local frame

As we observed in Section III-B, the Cartesian controller based on the horizontal trunk frame manages to improve the motion capabilities, by introducing a base frame for the wheels that is *partially decoupled* from the trunk link. However, it is unable to achieve local control of the trunk along the coupled directions, namely x, y, and yaw axes. Such commands result in the whole robot rolling in the given direction w.r.t. the world frame, whereas our desired outcome is a local adjustment of the trunk.

In order to achieve the complete decoupling between the local frame and the trunk frame, we propose to inject additional degrees of freedom in the robot model, between the robot trunk and a newly introduced *virtual frame (VF)* which, intuitively speaking, can be interpreted as an additional world frame which *travels with the robot*. The configuration vector for the Centauro robot is thus changed w.r.t. (1), as follows:

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_{\text{fb}} \\ \boldsymbol{q}_{\text{v}} \\ \boldsymbol{q}_{a} \end{bmatrix}, \tag{12}$$

with $\boldsymbol{q}_{\text{v}} \in \mathbb{R}^6$ being a minimal representation of the virtual frame pose w.r.t. the trunk frame. A pictorial representation of the resulting kinematic model is given in Figure 4.

This virtual frame of reference can then be used as base frame for both the wheels and the trunk tasks. Indeed, having introduced six additional degrees-of-freedom between the trunk and the virtual frame (red chain in Figure 4), full local control of the robot waist can be achieved, whereas the global motion of the robot is obtained by means of a Cartesian task for the virtual frame w.r.t. the world, as described by the following stack of tasks:

$$\begin{pmatrix} \left(\sum_i {}^{\text{VF}}\mathcal{T}^{[\text{XYZ}]}_{\text{Wheel}_i} + {}^{\text{World}}\mathcal{T}_{\text{VF}}\right) / \\ \left({}^{\text{VF}}\mathcal{T}_{\text{Trunk}} + {}^{[\cdot]}\mathcal{T}_{\text{Hands}} + \sum_i {}^{\text{World}}\mathcal{T}^{[\text{RPY}]}_{\text{Ankle}_i}\right) / \\ \left(\mathcal{T}_{\text{Posture}}\right) \end{pmatrix} << \begin{pmatrix} \mathcal{C}^{\text{Pos.}}_{\text{Lims}} + \mathcal{C}^{\text{Vel.}}_{\text{Lims}} \end{pmatrix}. \tag{13}$$

D. Discussion

As it has been shown in Section III-C, full local control of the trunk can be achieved by augmenting the kinematic model with an additional virtual chain. A controller based on

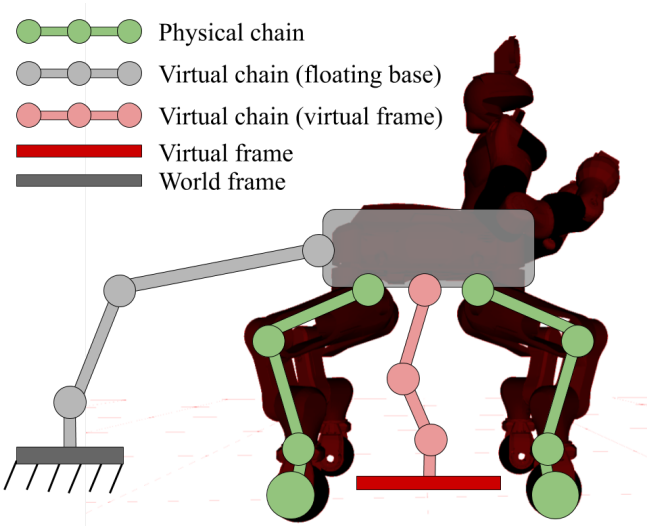

Fig. 4. Augmented kinematic model for a wheeled-legged quadruped. A virtual local frame (dark red) is introduced in order to achieve full local control of the trunk frame. An additional virtual chain (light red) is used to connect such a frame to the robot, so that full decoupling can be obtained.

such formulation can fully exploit the platform flexibility, by adding the ability to shift the trunk w.r.t. the support polygon about all axes. Furthermore, having defined a local frame which is decoupled from the robot trunk enables taking full advantage of our whole-body prioritized control framework. By relaxing the local trunk task ${}^{\text{VF}}\mathcal{T}_{\text{Trunk}}$ (i.e. putting it at low priority, or removing some of its degrees of freedom from the IK problem), the solver can adapt its posture to accommodate for more extreme desired poses of the wheels or of the end-effectors; in other words, local trunk adjustments are managed automatically by the solver. For instance, whenever a wheel (or a hand) is commanded to a local pose which exceeds its own workspace, an adaptation of the trunk pose will be required. Because the first two strategies employ the trunk itself as local frame, they are unable to provide such an adaptation; on the other hand, the virtual-frame based approach has enough degrees of freedom between the local frame and the trunk frame to provide the required trunk adaptation and accomplish the task, as it will be exemplified in Section V (see Figure 9). Having adopted a whole-body floating base formulation, we gained the ability to freely mix local adjustments with global control, by setting the base frame of each task to be the virtual frame or the global world, respectively. Notice how chain based formulations, as e.g. the IK used in [2], need to explicitly reason about desired poses for wheels and end-effectors w.r.t. the trunk frame in order to achieve some task that is natively defined w.r.t. the world frame; for instance, a pitching motion for the base can be obtained by changing the length of front and rear legs. Such poses must be tracked accurately (high λ as in (5)), and activation of a constraint on one chain is not taken into account on other chains, thus hindering automatic adaptations as in Figure 9. On the contrary, our controller allows the user to directly specify the desired task in its native coordinate system, while the solver takes care of the correct relationships between all relevant frames.

IV. Pure Rolling Condition

For the sake of simplicity, previous discussion has neglected the problem of guaranteeing the pure rolling of each wheel on the ground surface. However, such matter is of paramount importance, since the planned motion can be accurately transferred to the hardware only if the relevant *contact conditions* are not violated. In the case of a wheeled robot, we need to ensure a zero-slippage condition, which means that the contact point of each wheel must have zero velocity w.r.t. the ground:

$$\boldsymbol{v}_C = \boldsymbol{J}_C \dot{\boldsymbol{q}} = 0, \tag{14}$$

where $\boldsymbol{J}_C \in \mathbb{R}^{3\times n}$ is the *contact Jacobian*, i.e. the Jacobian of a point $\boldsymbol{p}_C$ which instantaneously moves with the wheel, but is always located at the contact point. If we let $\boldsymbol{p}_w$ denote the center of the wheel, R the wheel radius, and $\boldsymbol{n}_C$ the outward normal to the contact surface, then such a point is given by the following equation:

$$\boldsymbol{p}_C = \boldsymbol{p}_w - R\,\boldsymbol{n}_C. \tag{15}$$

A. Steering control

To gain further insight on how to effectively enforce the pure rolling constraint, we single out the contribution of the wheel joints, as in the equation below:

$$\boldsymbol{v}_C = \boldsymbol{v}_w + R\,\dot{q}_w \boldsymbol{i}_a, \tag{16}$$

where all quantities are conveniently expressed w.r.t. the local frame, such that they are time-invariant for a constant motion in local coordinates. The meaning of the symbols in (16) is as follows: $\dot{q}_w$ is the angular velocity of the wheel about its spinning axis, $\boldsymbol{v}_w$ represents the absolute velocity of the wheel's center, and $\boldsymbol{i}_a$ is the direction of the ankle frame x-axis. Such a frame is defined as depicted in Figure 5: the x-axis points along the wheel forward direction, the z-axis coincides with the steering joint axis, and the y-axis completes the right-handed frame. Furthermore, we will assume $\boldsymbol{v}_w$ to be constant.

With the aim to control the contact velocity to zero, we first compute its variation as follows:

$$\dot{\boldsymbol{v}}_C = R\,\ddot{q}_w \boldsymbol{i}_a + R\,\dot{q}_w\,(\boldsymbol{\omega}_a \times \boldsymbol{i}_a) + \cancelto{0}{\dot{\boldsymbol{v}}_w}; \tag{17}$$

moreover, since the ankle frame angular velocity $\boldsymbol{\omega}_a$ is given by the steering joint rotation, we set $\boldsymbol{\omega}_a = \dot{q}_s \boldsymbol{k}_a$, with $\dot{q}_s$ denoting the steering joint velocity, and $\boldsymbol{k}_a$ being the direction of the steering axis; this yields

$$\dot{\boldsymbol{v}}_C = R\,\ddot{q}_w \boldsymbol{i}_a. + R\,\dot{q}_w \dot{q}_s\,\boldsymbol{j}_a. \tag{18}$$

It can be noticed that the rate of change of contact velocity is given by two terms: a forward acceleration component in the direction $\boldsymbol{i}_a$, which can be controlled to zero by appropriately spinning the wheel joint, and a lateral component in the direction $\boldsymbol{j}_a$ which is also influenced by the steering joint

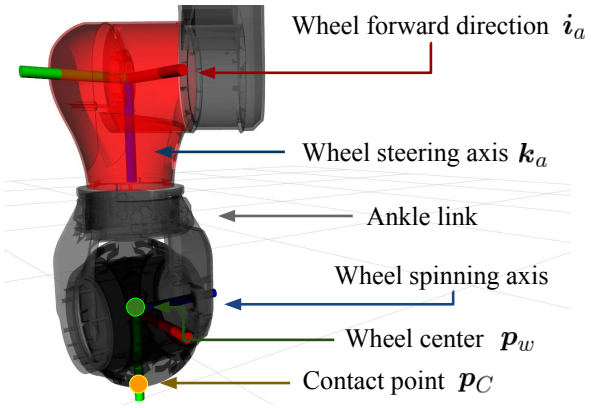

Fig. 5. Structure and reference frames for the Centauro wheel complex.

speed. In principle, the lateral contact velocity can be made to vanish by enforcing the following first order dynamics

$$\boldsymbol{j}_a \cdot [\dot{\boldsymbol{v}}_C + k_C \boldsymbol{v}_C] = 0, \quad k_C > 0, \tag{19}$$

which can be done with a suitable choice of the steering joint velocity. Substituting (18) in (19) and solving for $\dot{q}_s$ gives rise to the steering control law given by the following expression[3]:

$$\dot{q}_s = k_C \frac{\boldsymbol{j}_a \cdot \boldsymbol{v}_C}{R\,\dot{q}_w}, \tag{20}$$

where the scalar gain k_C controls the speed of convergence. Finally, note that the forward component is continuously canceled by the contact task (14).

B. Dealing with joint limits

Under the assumption that the steering joint can spin continuously, (20) provides a feasible solution to the wheel steering problem. However, this is not the case for our Centauro robot: its steering motors are indeed characterized by hard stops, that prevent the cables connecting the wheel to the robot from excessive twisting. Equation (20) is a *local* law, in the sense that it has no knowledge about whether the commanded motion will eventually lead to a constraint violation. For this reason, the presence of joint limits demands a different approach to be followed.

More specifically, we note that in order for $\boldsymbol{v}_C$ to be zero, a necessary condition is that velocity of the wheel center be parallel to the wheel forward direction, so that the two terms in (16) can cancel out. Clearly, such a condition admits two solutions, which can be obtained by adding or subtracting 180 degrees. Steering angle candidates can be computed by the following argument: we first consider the angle ϑ between $\boldsymbol{v}_C$ expressed in the ankle frame and the direction $\boldsymbol{i}_a$; such an angle represents a *steering error* and, as such, can be added to the current steering angle in order to obtain the correct ones:

$$\begin{cases} q_s^{(1)} = q_s + \vartheta \\ q_s^{(2)} = \mathrm{wrap}_{[-\pi,\pi]}\left(q_s^{(1)} + \pi\right) \end{cases}, \tag{21}$$

If the steering joint range spans more than 180 degrees, then at least one between $q_s^{(1)}$ and $q_s^{(2)}$ will not violate the limits. Finally, in order to obtain an as smooth trajectory as possible for the steering joint, whenever both solutions are valid, we select the one that is the closest to the current value.

C. Integration into a stack of tasks

To enforce the pure rolling condition into a stack of tasks, we define two tasks, $\mathcal{T}_{\text{Steering}}$ and $\mathcal{T}_{\text{Rolling}}$, to separately handle steering (21) and slippage control (14). Such tasks must be integrated into the Cartesian control problems (9), (11), and (13) by placing them at the appropriate priority level. To this aim, we observe that the rolling task must have lower priority than the wheel position task ${}^{\text{VF}}\mathcal{T}_{\text{Wheel}}^{[\text{XYZ}]}$, so that the wheel center is free to translate despite the steering angle not being numerically equal to its optimal value; hence, one possibility is to place it at the second priority level. Conversely, steering tasks should be placed at highest priority, so that the steering angle is not affected by the ankle orientation task ${}^{\text{World}}\mathcal{T}_{\text{Ankle}}^{[\text{RPY}]}$. The reader will notice how these considerations are of *heuristic* nature, as they originate from intuition and experience; indeed, selecting an optimal hierarchy requires to assign (i) a priority level and (ii) a weight to each task, which results in a strongly non-linear, combinatorial complexity problem, which is also task-specific. The interested reader may refer to [19] for a possible approach to hierarchy learning.

V. Experiments

The proposed control algorithm was implemented inside the *CartesI/O* framework [20], that is a ROS-based library for online Cartesian control with real-time (RT) support. Under the hood of CartesI/O, the *OpenSoT* library [17] implements the math of tasks in the C++ language through operator overloading; the corresponding stack-of-tasks of the iHQP problem is then set up and solved by OpenSoT in a RT-safe way, leveraging off-the-shelf QP solvers. The Centauro robot is powered by the *XBotCore* middleware [21], which allows for mixed RT (through the development of *real-time plugins*) and non-RT control (via ROS integration). Because the whole framework is parametrized in terms of a standard *URDF*-based description of the robot, implementation of the virtual-frame concept only required to provide an augmented URDF model with appropriate additional joints and links.

To validate the proposed control scheme against our motion requirements (M1) – (M4), and most notably in terms of transferability to the hardware, an extensive set of experiments has been designed and performed. The stack of tasks employed for the experimental sessions is based on (13), adding further tasks for steering and slippage control, as discussed in Section IV-C. Solving the cascaded QPs of the iHQP formulation required on average $\bar{t}_{\text{cpu}} = 3.4$ ms, with a standard deviation of $\sigma_{\text{cpu}} = 0.17$ ms. The reader is

[3]Equation (20) can be regularized around $\dot{q}_w = 0$ by replacing $\frac{1}{\dot{q}_w}$ with a term such as $\frac{\dot{q}_w}{\dot{q}_w^2 + \epsilon}$ for some $\epsilon > 0$.

encouraged to check out the accompanying video, which is also available at `https://youtu.be/uIaAGrhMbuY`.

A first experiment consists of a single run where the robot performs a driving motion (M1) while simultaneously adjusting the support polygon (see Figure 6) from a starting squared shape, to a narrower configuration, and then to a wider one (M2). Afterwards, local adjustments while driving

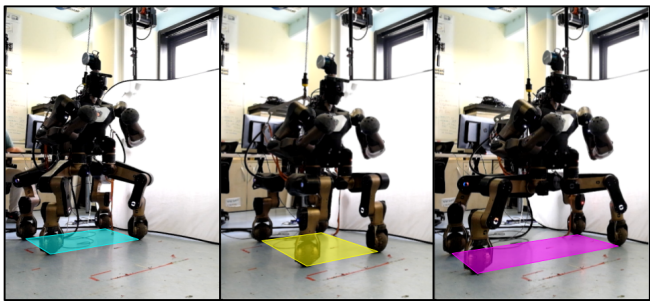

Fig. 6. Support polygon shape modulation while performing a driving motion with forward speed of $v = 0.05$ ms^{-1}.

(Figure 7) are validated with a sequence of lateral, sagittal, and rotation adjustments (M3). Finally, the robot stops to

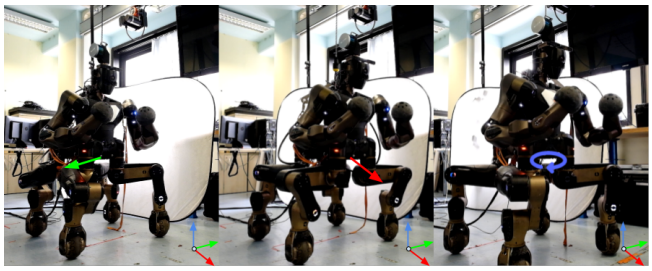

Fig. 7. Trunk motion w.r.t. the virtual frame in right, forward, and clockwise direction.

perform a simulated manipulation task in world coordinates (Figure 8), which consists in reaching a high position; the trunk task ${}^{\text{VF}}\mathcal{T}_{\text{Trunk}}$ is deactivated in order for the trunk to adapt to the end-effectors desired pose. Then, while the end-effectors are kept fixed w.r.t. the global world frame, the support polygon is adjusted by sending suitable references to the virtual frame (M4).

In order to validate the ability of the proposed controller to deal with constraint activation in a whole-body fashion, we command the rear-right wheel to lift from the ground (Figure 9), after having adjusted the trunk position in order to avoid the robot to fall. The commanded position cannot be reached by only moving the rear-right leg, due to the hip pitch joint reaching its hard stop. In such a scenario, where chain-based solvers would be bound to fail, our strategy is able to automatically adapt the trunk position (which is set to have lower priority than the wheel) in order to complete the task. A quantitative assessment is given in Figure 10, where the virtual frame based scheme achieves close to zero wheel position error at the apex of the commanded trajectory (mid point of the red shaded area). Notice how, because end-effectors are being controlled w.r.t. the local frame, the arms compensate for the trunk motion in order to keep the end-effectors in position. This further highlights the benefits of a whole-body approach.

The effectiveness of our proposed steering strategy is assessed in Figure 11, which shows a time history of each wheel slippage velocity, i.e. the relative velocity between each wheel contact point the the ground. It can be noticed how such values are moderately close to zero, resulting in a good fulfillment of the desired contact condition. Short spikes in the slippage profile indicate fast steering maneuvers to align each wheel to the correct direction given by (21). Longer spikes are instead due to the switching between the two solutions in (21), that is necessary due to hard stops in the steering joints. Nevertheless, our simple steering strategy is sufficient for a smooth transition from a kinematic model to the real hardware, as it can be assessed in the accompanying video.

VI. Conclusion and Future Works

The present work has introduced a novel methodology for the Cartesian control of hybrid wheeled-legged robots. A comparison with other approaches has been discussed, leading to two conclusions; first, a local frame of reference for the robot is to be selected with care; indeed, the trivial solution does not permit to achieve the complete control of the platform, as given by the set of motion requirements $(\text{M1}) - (\text{M4})$. Second, only by adding further virtual degrees of freedom to the kinematic model it is possible to achieve a fully decoupled control of the trunk frame w.r.t. to the local frame. Such a choice allows to mix tasks that are naturally defined in local coordinated with tasks that are defined in global coordinates, while taking full advantage of a whole-body floating base formulation. Finally, steering strategies have been proposed and analyzed, and a thorough experimental validation has been carried out.

Future work will address the application of the proposed method to a trunk stabilizer with terrain adaptation capabilities, which will enable the Centauro robot to locomote on non-flat surfaces.

References

[1] N. Kashiri, L. Baccelliere, L. Muratore, A. Laurenzi, Z. Ren, E. M. Hoffman, *et al.*, "Centauro: A hybrid locomotion and high power resilient manipulation platform," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1595–1602, 2019.

[2] M. Schwarz, T. Rodehutskors, M. Schreiber, and S. Behnke, "Hybrid driving-stepping locomotion with the wheeled-legged robot momaro," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[3] K. Tahara, S. Arimoto, and M. Yoshida, "Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[4] Y. Wang, C. Smith, Y. Karayiannidis, and P. Ögren, "Cooperative control of a serial-to-parallel structure using a virtual kinematic chain in a mobile dual-arm manipulation application," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2372–2379, 2015.

[5] N. Dehio, J. Smith, D. L. Wigand, G. Xin, H. Lin, J. J. Steil, and M. Mistry, "Modeling and control of multi-arm and multi-leg robots: Compensating for object dynamics during grasping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

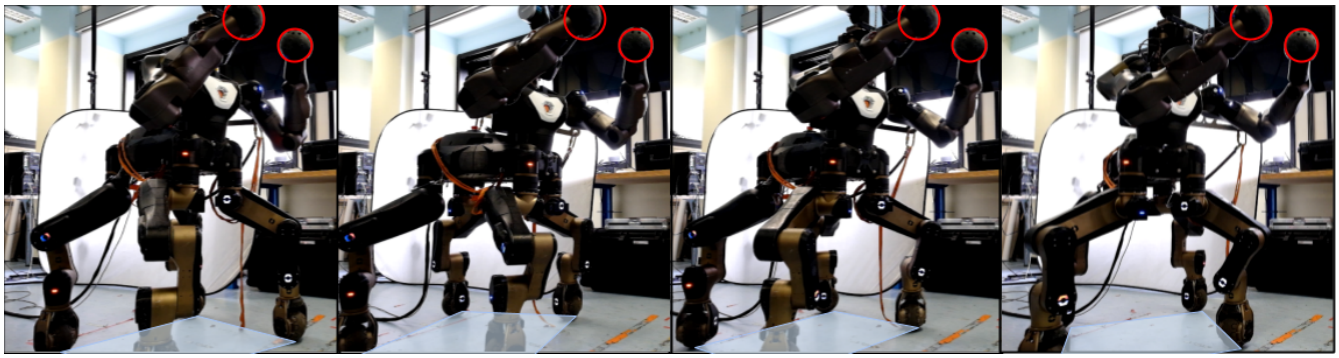

Fig. 8. Manipulation in world frame coordinates with simultaneous support polygon (SP) adjustment. First, the SP is shifted backwards; then, it is rotated clockwise; finally, it is rotated counterclockwise. The trunk posture is automatically adapted by the solver.

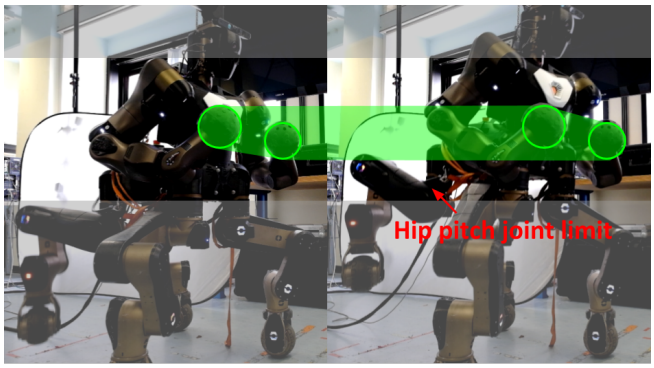

Fig. 9. Trunk adaptation in the presence of constraints. Gray shaded areas highlight the upward motion of the trunk, whereas end-effectors keep their position as emphasized by the green shaded areas.

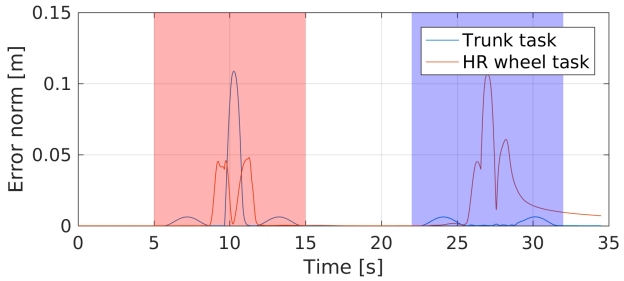

Fig. 10. Wheel lifting under the proposed stack of tasks (red area), compared to the same task when the local frame coincides with the trunk frame (blue area).

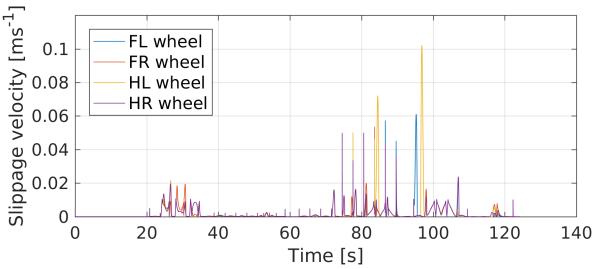

Fig. 11. Time history of the absolute velocity of the wheels contact points w.r.t. the ground, computed along the commanded motion sequence.

[6] S. Dubowsky and E. Papadopoulos, "The kinematics, dynamics, and control of free-flying and free-floating space robotic systems," *IEEE Transactions on Robotics and Automation*, Oct 1993.

[7] M. Kamedula, N. Kashiri, and N. G. Tsagarakis, "On the Kinematics of Wheeled Motion Control of a Hybrid Wheeled-Legged CENTAURO robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2426–2433, 2018.

[8] M. Bjelonic, C. D. Bellicoso, Y. de Viragh, D. Sako, F. D. Tresoldi, F. Jenelten, and M. Hutter, "Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2116–2123, 2019.

[9] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 160, 2018.

[10] C. Grand, F. Benamar, and F. Plumet, "Motion kinematics analysis of wheeled–legged rover over 3D surface with posture adaptation," *Mechanism and Machine Theory*, vol. 45, no. 3, pp. 477 – 495, 2010.

[11] K. Iagnemma, A. Rzepniewski, S. Dubowsky, and P. Schenker, "Control of robotic vehicles with actively articulated suspensions in rough terrain," *Autonomous Robots*, vol. 14, pp. 5–16, Jan 2003.

[12] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of dynamic systems, measurement, and control*, vol. 108, no. 3, pp. 163–171, 1986.

[13] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE Journal on Robotics and Automation*, vol. 4, Aug 1988.

[14] B. Siciliano and J. . E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments*, pp. 1211–1216 vol.2, June 1991.

[15] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, 2014.

[16] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[17] E. Mingo Hoffman, A. Rocchi, A. Laurenzi, and N. G. Tsagarakis, "Robot Control for Dummies: Insights and Examples using OpenSoT," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 736–741, 2017.

[18] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2554–2561, 2013.

[19] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell, "Learning task priorities from demonstrations," *IEEE Transactions on Robotics*, vol. 35, pp. 78–94, Feb 2019.

[20] A. Laurenzi, E. Mingo Hoffman, L. Muratore, and N. G. Tsagarakis, "CartesI/O: A ROS Based Real-Time Capable Cartesian Control Framework," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

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