Robotics and Autonomous Systems 166 (2023) 104448

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Multi-contact planning and control for humanoid robots: Design and validation of a complete framework

Paolo Ferrari [a,1], Luca Rossini [b,d,*,1], Francesco Ruscelli [b], Arturo Laurenzi [b], Giuseppe Oriolo [a], Nikos G. Tsagarakis [b], Enrico Mingo Hoffman [b,c]

[a] Dipartimento di Ingegneria Informatica, Automatica e Gestionale (DIAG), Sapienza Università di Roma, Via Ariosto 25, Roma, 00185, Italy
[b] Humanoid and Human-Centred Mechatronics (HHCM) lab, Istituto Italiano di Tecnologia (IIT), Via Morego 30, Genova, 16137, Italy
[c] PAL Robotics S.L., Carrer de Pujades 77-79, Barcelona, 08005, Spain
[d] Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), Università di Genova, Via Opera Pia 13, Genova, 16145, Italy

ARTICLE INFO

Article history:
Received 26 October 2022
Received in revised form 11 January 2023
Accepted 5 May 2023
Available online 11 May 2023

Keywords:
Multi-contact framework
Motion planning
Torque-controlled humanoid robots
Loco-manipulation

ABSTRACT

In this paper, we consider the problem of generating appropriate motions for a torque-controlled humanoid robot that is assigned a multi-contact loco-manipulation task, i.e., a task that requires the robot to move within the environment by repeatedly establishing and breaking multiple, non-coplanar contacts. To this end, we present a complete multi-contact planning and control framework for multi-limbed robotic systems, such as humanoids. The planning layer works offline and consists of two sequential modules: first, a stance planner computes a sequence of feasible contact combinations; then, a whole-body planner finds the sequence of collision-free humanoid motions that realize them while respecting the physical limitations of the robot. For the challenging problem posed by the first stage, we propose a novel randomized approach that does not require the specification of pre-designed potential contacts or any kind of pre-computation. The control layer produces online torque commands that enable the humanoid to execute the planned motions while guaranteeing closed-loop balance. It relies on two modules, i.e., the stance switching and reactive balancing module; their combined action allows it to withstand possible execution inaccuracies, external disturbances, and modeling uncertainties. Numerical and experimental results obtained on COMAN+, a torque-controlled humanoid robot designed at Istituto Italiano di Tecnologia, validate our framework for loco-manipulation tasks of different complexity.

1. Introduction

Thanks to their structure, humanoid robots have the potential to accomplish complex tasks that require moving within unstructured and confined environments by repeatedly establishing and breaking multiple, non-coplanar contacts, using feet, hands, and possibly other parts of the body. Examples of *multi-contact loco-manipulation tasks* include crawling under low obstacles, climbing a ladder, and standing up exploiting the environment as support. The latter example is shown in Fig. 1.

To effectively fulfill these tasks, the humanoid must be able to autonomously decide and execute appropriate motions that respect several crucial constraints such as balance, collision avoidance, and kinematic/dynamic limitations. This problem needs to be addressed both at the planning and control level and is therefore known as *Multi-Contact Planning and Control* (MCPC) problem [1]. Although MCPC is a very active research field for almost two decades, humanoids are still far from being able to perform complex multi-contact loco-manipulation tasks in real-world scenarios, as highlighted by the 2015 DARPA Robotics Challenge [2].

1.1. Previous works

At the planning level, the MCPC problem requires computing a discrete sequence of contact combinations, called *stances*, together with a sequence of continuous whole-body motions to realize them. The stance-before-motion paradigm introduced by [3], i.e., choosing first the sequence of stances and subsequently, a sequence of compatible motions is recognized as the most suitable in multi-contact scenarios. The problem of planning the stance sequence is particularly challenging due to its combinatorial nature, especially when traversing complex environments: the sequence of contact combinations is in fact acyclic,

* Corresponding author at: Humanoid and Human-Centred Mechatronics (HHCM) lab, Istituto Italiano di Tecnologia (IIT), Via Morego 30, Genova, 16137, Italy.
E-mail address: luca.rossini@iit.it (L. Rossini).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.robot.2023.104448

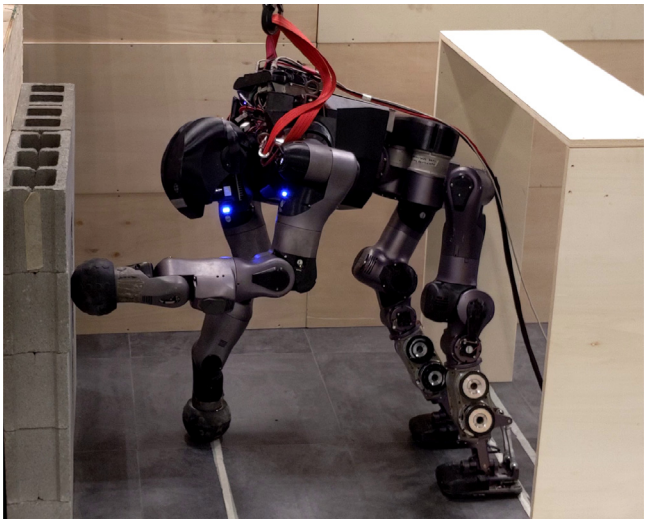


Fig. 1. An example of a multi-contact loco-manipulation task: COMAN+ stands up exploiting the environment, in particular the wall and the ground, as support.

differently from the case of pure biped locomotion in which the identity of the foot in contact with the ground regularly alternates between the right and left. Early approaches, such as [4], deal with the problem complexity by first creating a stance-adjacency graph based on a set of predesigned possible contacts between robot and environment points, and then searching it to find an appropriate sequence of stances, each one having an associated, kinematically consistent robot configuration.

Other methods (e.g., [5,6]) avoid the specification of predesigned potential contact points, thus allowing contacts anywhere in the environment. These methods first find a guide path, via a standard sampling-based method, for a free-floating robot model that does not collide with obstacles but keeps the limbs sufficiently close to them. Then, the stance sequence is computed through a best-first search in which the generation of associated configurations is driven by the guide path. A more recent work [7], adopting a similar approach, impressively improves the search efficiency by precomputing a feasible set of configurations for each kinematic chain of the robot.

In contrast to the stance-before-motion paradigm, there exist other techniques where motions and contacts are planned simultaneously. These methods are usually based on trajectory optimization (TO) where contacts are specified implicitly or explicitly. For example, in [8], a non-linear TO based on direct collocation is used to discover contacts and motions for Single Rigid Body Dynamics (SRBD) template models, given an explicit contact schedule. Different gaits are discovered by parameterizing and optimizing each step sequence over time. This approach permits non-gaited locomotions to emerge for quadrupeds, point-feet bipeds, and monopods. A similar approach, based on multiple shooting method, has been used in [9] to generate rappelling and climbing maneuvers on a template biped model hanged on a rope. In order to avoid an explicit contact schedule, implicit methods rely on complementarity: if a contact force exists, the contact velocity is nullified and vice-versa [10–13], letting the solver choose the sequence and timing of contacts. Both in the case of implicit or explicit contacts specifications, the TO can be based on simplified models, e.g. Linear Inverted Pendulum or SRBD [14,15], or more recently, on the full robot model [16,17]. Despite TO-based methods result being effective, particularly regarding the generation of highly dynamic and contact-rich motions, the inclusion of complex environments is difficult due to the necessity to have continuous and differentiable environment descriptions, as well as the inclusion of (self)collision avoidance constraints, particularly regarding high Degrees of Freedom (DoFs) systems, such as humanoid robots. Furthermore, the application of such methods for multi-contact planning and control on real legged systems has been limited to quadrupedal robots. Other techniques encompass contact-consistent elastic strips (CES) [18] that plan the stance sequence simultaneously to the associated configurations and the motions between them, respectively. To make the problem tractable, CES involves a preliminary phase in which a sequence of contact regions is computed.

Once the sequences of stances and associated configurations are found, the next step consists in generating the motions between them. In literature, this is done either in an online or offline fashion. With online approaches, when moving from the current stance to the next, the robot configuration is regulated to that associated with the latter using a whole-body controller [19]. On the other hand, offline approaches (e.g., [20]), involve constrained versions of sampling-based planners (see [21] for a recent review) to compute the complete robot motion before execution. Note that, sampling-based methods are also involved in other contexts of humanoid motion planning, such as pure manipulation and biped locomotion planning, which are intrinsically easier than the multi-contact planning problem to be solved. Indeed, in the first case (see, e.g., [22,23]), only tasks that do not require stepping are accomplished; in the second case, randomized approaches are used to search sequences of footsteps [24] or whole-body motions [25] to accomplish tasks requiring cyclic gaits.

At the control level, the MCPC problem consists of choosing online the inputs for the robot actuators in such a way as to track at best the open-loop planned motion while guaranteeing closed-loop balance at any time instant. These inputs correspond to the torque commands when dealing with torque-controlled robots. While motion tracking can be achieved using classical compliance/impedance-based techniques [26], instantaneous balance requires adjusting the distribution of contact forces among the contacting robot links and consequently the joint torques to realize them. In literature, this problem was solved by single- and double-stage methods. The first approach (used for instance in [27,28]) simultaneously optimizes contact forces and joint torques exploiting the full-body inverse dynamics of the robot. On the contrary, double-stage methods either pre- or post-optimize the contact forces. Approaches based on pre-optimization (such as [29–31]) first solve the optimal contact force distribution problem and then maps them to joint torques. Such a strategy requires the (quite tricky) specification of a reference angular momentum. This is avoided with approaches based on post-optimization (such as [32,33]) which first computes the joint torques treating the humanoid as a fully-actuated fixed-base system, and then maps them to contact forces for the actual underactuated system.

Each of the works mentioned above focuses on a specific aspect of the MCPC problem, either related to planning or control. In literature, the few works addressing the complete MCPC problem propose techniques that are specifically designed for a single task such as climbing stairs using a handrail [34], climbing a ladder [35], or pushing a heavy object [36]. To the authors' knowledge, a work encompassing the general MCPC problem, from the design to the validation of a complete framework, with an application on a real humanoid platform, is still missing in the literature.

1.2. Paper contribution and organization

The objective of this paper is to address the complete MCPC problem for torque-controlled multi-limbed robots, in particular humanoids, in the presence of loco-manipulation tasks. Specifically, we consider tasks that, due to their intrinsic complexity

and the limitations imposed by the environment in which they must be performed, are typically achieved via quasi-static motions even by human beings. This is the case, for example, when climbing a ladder or crawling under low obstacles.

To solve the MCPC problem in its entirety, we design a complete framework that includes both a planning and a control layer. The planning layer adopts the above-mentioned stance-before-motion paradigm: first, a stance planner computes a sequence of stances, and then a whole-body planner finds the sequence of humanoid motions to realize them. The control layer allows the execution of the planned motions while guaranteeing closed-loop balance through the action of two specific modules, namely the stance switching and reactive balancing module, which aim at absorbing all possible execution inaccuracies, external disturbances, and modeling uncertainties.

The contributions of the paper can be summarized as follows:

1. We describe our complete MCPC framework, presenting the design of both the planning and control layer, and giving the details of their constituting modules.
2. We propose a novel approach for solving the challenging problem of planning the sequence of stances, which leverages on a randomized strategy. Thanks to such randomized nature, our stance planner does not require the specification of predesigned potential contacts or any kind of precomputation, and thus can be used in different scenarios without significant parameter tuning.
3. We validate our MCPC framework by presenting numerical and experimental results obtained with the COMAN+ humanoid for multi-contact loco-manipulation tasks of different complexity. In particular, we show a real-world application of our framework in which COMAN+ first navigates a narrow passage through a quadrupedal walk, and then stands up exploiting the environment as support. To the authors' knowledge, this is the first time that a multi-contact task of such complexity is demonstrated on real hardware.

This paper is organized as follows. Section 2 recalls notions related to MCPC and the used notation. Section 3 provides the precise formulation of the considered MCPC problem. An overview of the proposed MCPC framework is given in Section 4. The stance and whole-body planners are described in Sections 5 and 6, respectively. The control layer is presented in Sections Section 7. Section 8 gives some details about our ROS-based implementation of the proposed MCPC framework. Section 9 presents numerical and experimental results obtained with COMAN+. The advantages and possible limitations of the proposed MCPC framework are discussed in Section 10. Finally, Section 11 concludes the paper with possible future work directions.

2. Background

Before formally defining the problem of interest, we recall some basic notions that will be used throughout the paper.

2.1. Definitions and notation

The *configuration* of the humanoid robot is described by the vector of generalized coordinates $\boldsymbol{q} = [\boldsymbol{q}_{\text{fb}}^T, \boldsymbol{q}_{\text{jnt}}^T]^T$, where $\boldsymbol{q}_{\text{fb}} = [\boldsymbol{p}_{\text{fb}}^T, \boldsymbol{o}_{\text{fb}}^T]^T \in SE(3)$ is the pose, with $\boldsymbol{p}_{\text{fb}}$ and $\boldsymbol{o}_{\text{fb}}$ the position and orientation[2] coordinates, of the floating-base frame $\mathcal{F}^{\text{fb}}$ w.r.t. the inertial world frame $\mathcal{F}^{\text{w}}$, and $\boldsymbol{q}_{\text{jnt}}$ is the n-vector of joint angles. The dimension of the configuration space $\mathcal{C}$ is $\dim(\mathcal{C}) = 6 + n$.

[2] Throughout the paper it is assumed that a singularity-free representation is used for describing orientations.

Two types of contacts, i.e., *point* and *surface* contacts, may occur between the robot and the environment, and they can be maintained in the *fixed* mode. With a point/surface contact, a point/surface on the exterior of a robot link touches a point/surface of the environment. Moreover, maintaining a point/surface contact in the fixed mode fully constrains the position/pose of the contacting robot point/surface. Contact types other than the point and surface ones, e.g. edge contacts, and contact modes other than the fixed one, e.g. sliding or rolling contacts, are out of the scope of this paper.

A *stance* is a set $\sigma = \{c_1, \ldots, c_m\}$ of m contacts. We represent its generic *contact* by a triplet $c_i = \langle \mathtt{t}_i, \mathcal{F}_i, \boldsymbol{r}_{c,i} \rangle$, whose fields are described below:

- $\mathtt{t}_i \in \{\mathtt{P}, \mathtt{S}\}$ is the contact type. In particular, $\mathtt{t}_i = \mathtt{P}$ if c_i is a point contact, and $\mathtt{t}_i = \mathtt{S}$ if c_i is a surface contact.
- $\mathcal{F}_i$ is the contact frame, i.e., a reference frame rigidly attached to the contacting robot point or surface. The pose $\boldsymbol{r}_i = [\boldsymbol{p}_i^T, \boldsymbol{o}_i^T]^T$, with $\boldsymbol{p}_i$ and $\boldsymbol{o}_i$ the position and orientation coordinates, of $\mathcal{F}_i$ w.r.t. $\mathcal{F}^{\text{w}}$ is related to the robot configuration $\boldsymbol{q}$ by a forward kinematic map $\boldsymbol{r}_i = \boldsymbol{k}_i(\boldsymbol{q})$. At differential level, it becomes $\dot{\boldsymbol{r}}_i = \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}}$, with $\boldsymbol{J}_i(\cdot)$ the contact Jacobian, i.e., the Jacobian matrix of $\boldsymbol{k}_i(\cdot)$ w.r.t. $\boldsymbol{q}$. In the following, we will denote by $\boldsymbol{k}_{p,i}(\cdot)$ and $\boldsymbol{k}_{o,i}(\cdot)$, respectively, the position and orientation components of $\boldsymbol{k}_i(\cdot)$, and by $\boldsymbol{J}_{p,i}(\cdot)$ and $\boldsymbol{J}_{o,i}(\cdot)$ their corresponding Jacobian matrices.
- $\boldsymbol{r}_{c,i} = [\boldsymbol{p}_{c,i}^T, \boldsymbol{o}_{c,i}^T]^T$ is the pose of $\mathcal{F}_i$ w.r.t. $\mathcal{F}^{\text{w}}$ when c_i is established. While maintained, c_i yields a kinematic constraint of the form:

$$\boldsymbol{k}_{p,i}(\boldsymbol{q}) = \boldsymbol{p}_{c,i}, \quad \text{if } \mathtt{t}_i = \mathtt{P}, \tag{1}$$

$$\boldsymbol{k}_i(\boldsymbol{q}) = \boldsymbol{r}_{c,i}, \quad \text{if } \mathtt{t}_i = \mathtt{S}. \tag{2}$$

In a stance σ, each contact c_i involves a different contact frame. We will retrieve the set $\{\mathcal{F}_1, \ldots, \mathcal{F}_m\}$ of contact frames involved at a stance σ by a function $\Psi(\sigma)$.

A stance σ defines a submanifold $\mathcal{C}_\sigma$ of $\mathcal{C}$, called *stance submanifold*, containing all configurations $\boldsymbol{q}$ that satisfy the kinematic constraints (1)–(2) for all contacts. Let $m_{\mathtt{P}}$ and $m_{\mathtt{S}}$ be the number of, respectively, point and surface contacts in σ, such that $m = m_{\mathtt{P}} + m_{\mathtt{S}}$. Then, the dimension of $\mathcal{C}_\sigma$ is $\dim(\mathcal{C}_\sigma) = \dim(\mathcal{C}) - 3m_{\mathtt{P}} - 6m_{\mathtt{S}}$.

The manifold $\mathcal{C}_\sigma$ contains a subspace $\mathcal{D}_\sigma$, called *feasible subspace*, of feasible configurations. For a configuration $\boldsymbol{q} \in \mathcal{C}_\sigma$ to belong to $\mathcal{D}_\sigma$, it must satisfy the following conditions:

- Joint limits are respected, i.e.,

$$\boldsymbol{q}_{\text{jnt}}^{\min} \leq \boldsymbol{q}_{\text{jnt}} \leq \boldsymbol{q}_{\text{jnt}}^{\max}. \tag{3}$$

- Collisions with the environment, with the exception of the robot points and surfaces involved in the contacts specified by σ, and self-collisions are avoided.
- Static balance is guaranteed (related conditions are discussed in Section 2.2).

Two stances σ and σ' are *adjacent* if both the following conditions are satisfied:

- σ and σ' differ by a single contact, i.e., σ' can be reached by either removing ($\sigma \supset \sigma'$) or adding a contact ($\sigma \subset \sigma'$) from/to σ.
- $\mathcal{D}_\sigma \cap \mathcal{D}_{\sigma'} \neq \emptyset$, i.e., there exists (at least) one configuration $\boldsymbol{q}$, called *transition*, that belongs to both $\mathcal{D}_\sigma$ and $\mathcal{D}_{\sigma'}$. In particular, if $\sigma \subset \sigma'$ ($\sigma \supset \sigma'$), $\boldsymbol{q}$ is a transition if it satisfies the kinematic constraints yielded by the contacts in σ' (σ), and the static balance conditions using the contacts in σ (σ'), in addition to satisfy joint limits and collision avoidance.

2.2. Conditions for static balance

Consider a stance σ and, for each contact $c_i \in \sigma$, denote by $\boldsymbol{w}_{c,i} = [\boldsymbol{f}_{c,i}^T, \boldsymbol{\tau}_{c,i}^T]^T$ the contact wrench (expressed in $\mathcal{F}^{\mathrm{w}}$) exerted by the environment on the contacting robot point (if $\mathtt{t}_i = \mathtt{P}$) or surface (if $\mathtt{t}_i = \mathtt{S}$). Here, $\boldsymbol{f}_{c,i} = [f_{c,i}^x, f_{c,i}^y, f_{c,i}^z]^T$ and $\boldsymbol{\tau}_{c,i} = [\tau_{c,i}^x, \tau_{c,i}^y, \tau_{c,i}^z]^T$ are, respectively, the resultant of the applied contact forces and the moment of these forces around the origin of $\mathcal{F}_i$. Clearly, $\boldsymbol{\tau}_{c,i} = \mathbf{0}$ if $\mathtt{t}_i = \mathtt{P}$.

In the following, $\boldsymbol{w}_{c,i}^i = [\boldsymbol{f}_{c,i}^{i\,T}, \boldsymbol{\tau}_{c,i}^{i\,T}]^T$, with $\boldsymbol{f}_{c,i}^i = [f_{c,i}^{x,i}, f_{c,i}^{y,i}, f_{c,i}^{z,i}]^T$ and $\boldsymbol{\tau}_{c,i}^i = [\tau_{c,i}^{x,i}, \tau_{c,i}^{y,i}, \tau_{c,i}^{z,i}]^T$, denotes the contact wrench $\boldsymbol{w}_{c,i}$ expressed in frame $\mathcal{F}_i$ at a configuration $\boldsymbol{q}$, i.e.,

$$\boldsymbol{w}_{c,i}^i = \begin{bmatrix} \boldsymbol{R}_i^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{R}_i^T \end{bmatrix} \boldsymbol{w}_{c,i}, \tag{4}$$

where $\boldsymbol{R}_i$ is the rotation matrix associated with $\boldsymbol{k}_{o,i}(\boldsymbol{q})$.

Collect in vector $\boldsymbol{W}_c = [\boldsymbol{w}_{c,1}^T, \dots, \boldsymbol{w}_{c,m}^T]^T$ the contact wrenches at all $c_i \in \sigma$. The robot motion is related to $\boldsymbol{W}_c$ and the n-vector $\boldsymbol{\tau}$ of actuated joint torques by the Lagrangian equations

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}_c^T(\boldsymbol{q})\boldsymbol{W}_c, \tag{5}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the robot inertia matrix, $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the Coriolis and centrifugal term, $\boldsymbol{g}(\boldsymbol{q})$ the gravity term, $\boldsymbol{S}$ defined as

$$\boldsymbol{S} = [\mathbf{0}_{n\times6} \quad \boldsymbol{I}_{n\times n}]^T \tag{6}$$

models the system under-actuation, and $\boldsymbol{J}_c(\boldsymbol{q}) = [\boldsymbol{J}_1^T(\boldsymbol{q}), \dots, \boldsymbol{J}_m^T(\boldsymbol{q})]^T$ stacks all the contact Jacobians.

The humanoid at a configuration $\boldsymbol{q} \in \mathcal{C}_\sigma$ is in *static balance* if there exist contact wrenches $\boldsymbol{W}_c$ and actuated torques $\boldsymbol{\tau}$ satisfying the following conditions:

C1 The gravity is compensated. This condition is given by

$$\boldsymbol{g}_u(\boldsymbol{q}) = \boldsymbol{J}_{c,u}^T(\boldsymbol{q})\boldsymbol{W}_c, \tag{7}$$

$$\boldsymbol{g}_a(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{J}_{c,a}^T(\boldsymbol{q})\boldsymbol{W}_c, \tag{8}$$

which are obtained by considering (5) under quasi-static conditions, i.e., $\ddot{\boldsymbol{q}} = \dot{\boldsymbol{q}} = \mathbf{0}$, and explicitly separating the system unactuated (subscript u) and actuated (subscript a) parts associated to, respectively, the first 6 and last n rows of (5). Note that, condition (8) implicitly requires that the actuated torques $\boldsymbol{\tau}$ are within their limits, i.e.,

$$\boldsymbol{\tau}^{\min} \le \boldsymbol{\tau} \le \boldsymbol{\tau}^{\max}. \tag{9}$$

C2 For each contact $c_i \in \sigma$, the contact force $\boldsymbol{f}_{c,i}$ lies inside the Coulomb friction cone having apex at $\boldsymbol{p}_{c,i}$ and directed by the unit normal $\boldsymbol{n}_{c,i}$ at $\boldsymbol{p}_{c,i}$ pointing from the environment to the robot, i.e.,

$$\begin{aligned} &\boldsymbol{f}_{c,i} \cdot \boldsymbol{n}_{c,i} > 0, \\ &\|\boldsymbol{f}_{c,i}^t\| \le \mu_i(\boldsymbol{f}_{c,i} \cdot \boldsymbol{n}_{c,i}), \end{aligned} \tag{10}$$

which imposes unilaterality and non-slippage of c_i, with $\boldsymbol{f}_{c,i}^t = \boldsymbol{f}_{c,i} - (\boldsymbol{n}_{c,i} \cdot \boldsymbol{f}_{c,i})\boldsymbol{n}_{c,i}$ the tangential component of $\boldsymbol{f}_{c,i}$, and μ_i the static friction coefficient. By approximating the Coulomb friction cone with an inscribed pyramid (see [1, 37]), condition (10) takes the linear form

$$f_{c,i}^{z,i} > 0, \quad \left|f_{c,i}^{x,i}\right| \le \tilde{\mu}_i f_{c,i}^{z,i}, \quad \left|f_{c,i}^{y,i}\right| \le \tilde{\mu}_i f_{c,i}^{z,i}, \tag{11}$$

where $\tilde{\mu}_i = \mu_i/\sqrt{2}$.

C3 For each surface contact $c_i \in \sigma$ ($\mathtt{t}_i = \mathtt{S}$), the Center of Pressure (CoP) lies inside the contacting robot surface, i.e.,

$$\left|x_{\mathrm{CoP},i}^i\right| \le d_i^x, \quad \left|y_{\mathrm{CoP},i}^i\right| \le d_i^y, \tag{12}$$

where the CoP coordinates in frame $\mathcal{F}_i$ are given by

$$x_{\mathrm{CoP},i}^i = -\frac{\tau_{c,i}^{y,i}}{f_{c,i}^{z,i}}, \quad y_{\mathrm{CoP},i}^i = \frac{\tau_{c,i}^{x,i}}{f_{c,i}^{z,i}},$$

and d_i^x, d_i^y are the half-dimensions of the CoP rectangular,[3] admissible region.

C4 For each surface contact $c_i \in \sigma$ ($\mathtt{t}_i = \mathtt{S}$), the yaw moment $\tau_{c,i}^{z,i}$ is bounded as

$$\tau_{c,i}^{z,\min} \le \tau_{c,i}^{z,i} \le \tau_{c,i}^{z,\max}, \tag{13}$$

with

$$\begin{aligned} \tau_{c,i}^{z,\min} &= -\tilde{\mu}_i(d_i^x + d_i^y)f_{c,i}^{z,i} + \left|d_i^y f_{c,i}^{x,i} - \tilde{\mu}_i\tau_{c,i}^{x,i}\right| \\ &\quad + \left|d_i^x f_{c,i}^{y,i} - \tilde{\mu}_i\tau_{c,i}^{y,i}\right|, \\ \tau_{c,i}^{z,\max} &= \tilde{\mu}_i(d_i^x + d_i^y)f_{c,i}^{z,i} - \left|d_i^y f_{c,i}^{x,i} + \tilde{\mu}_i\tau_{c,i}^{x,i}\right| \\ &\quad - \left|d_i^x f_{c,i}^{y,i} + \tilde{\mu}_i\tau_{c,i}^{y,i}\right|. \end{aligned}$$

C1 is usually referred to as *centroidal statics condition*, while C2–C4 are known as *contact-stability conditions* (for details about their derivation see [38]).

3. Problem formulation

Consider a torque-controlled humanoid robot that is assigned a multi-contact loco-manipulation task, i.e., a task that requires the robot to move within the environment by repeatedly establishing and breaking multiple, non-co-planar contacts. In our formulation, the task is specified as a desired final stance σ^{fin} that the robot must reach from its initial stance σ^{ini}, which is given together with its initial configuration $\boldsymbol{q}^{\mathrm{ini}}$ and contact wrenches $\boldsymbol{W}_c^{\mathrm{ini}}$.

The objective of this paper is to design and validate a complete MCPC framework that enables the humanoid to autonomously plan and execute the quasi-static motions required to fulfill the assigned task. To this end, it is required to tackle three fundamental issues that will be solved by the MCPC framework, adopting the stance-before-motion paradigm, in the following order:

1. to find an appropriate sequence of adjacent stances leading to the desired one, together with their associated transitions;
2. to compute a sequence of feasible whole-body motions compatible with the sequence of stances, passing through the associated transitions (as graphically shown in Fig. 4);
3. to generate torque commands for the humanoid in order to realize the planned motions, while guaranteeing closed-loop balance at any time instant.

We address the described problem under the following assumptions.

A1 The environment is static and known. Its geometry is represented by a point cloud $\mathcal{P}$, and the unit normal $\boldsymbol{n}$ pointing from the environment to the robot can be readily computed at any point $\boldsymbol{p} \in \mathcal{P}$ whenever needed.

A2 Contacts can be established anywhere in the environment by using a predefined set $U = \{\mathcal{F}^{\mathrm{a}}, \mathcal{F}^{\mathrm{b}}, \mathcal{F}^{\mathrm{c}}, \dots\}$ of potential contact frames, henceforth referred to as *end-effectors*. Each $\mathcal{F}_h \in U$ is rigidly attached to a point/surface on the exterior

[3] Humanoid surfaces that are allowed to establish contacts are typically rectangular (e.g., a foot sole). In general, a rectangular region can also be involved as an inner approximation of a contacting surface having a more complex shape.

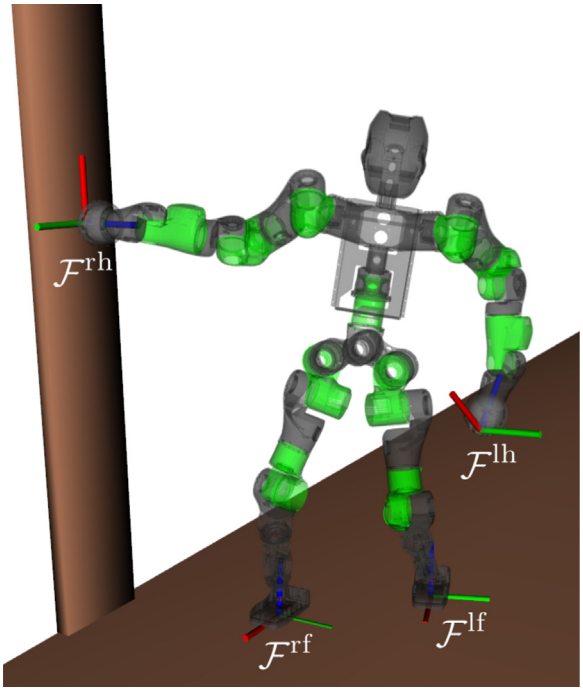


Fig. 2. The predefined set U of potential contact frames. In this example, $U = \{\mathcal{F}^{\text{lf}}, \mathcal{F}^{\text{rf}}, \mathcal{F}^{\text{lh}}, \mathcal{F}^{\text{rh}}\}$ contains the frames rigidly attached on the left/right foot/hand, whose x, y, and z axes are depicted in red, green and blue, respectively. Feet and hands can, respectively, establish surface and point contacts, i.e., $\varphi(\mathcal{F}^{\text{lf}}) = \varphi(\mathcal{F}^{\text{rf}}) = \text{S}$ and $\varphi(\mathcal{F}^{\text{lh}}) = \varphi(\mathcal{F}^{\text{rh}}) = \text{P}$. The robot is at a stance σ such that $\Psi(\sigma) = \{\mathcal{F}_1 = \mathcal{F}^{\text{lf}}, \mathcal{F}_2 = \mathcal{F}^{\text{rf}}, \mathcal{F}_3 = \mathcal{F}^{\text{rh}}\}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of a robot link that is allowed to establish point/surface contacts and is oriented so that the xy-plane is tangent/parallel to it and the z-axis points inward (see Fig. 2). We will retrieve the type of contact that an end-effector $\mathcal{F}_h \in U$ can establish by a function $\varphi(\mathcal{F}_h)$, with $\varphi(\mathcal{F}_h) = \text{P}$ or $\varphi(\mathcal{F}_h) = \text{S}$ depending on whether $\mathcal{F}_h$ can establish point or surface contacts, respectively.

4. Proposed framework

To solve the described problem, we propose the MCPC framework whose underlying architecture is shown in Fig. 3. It is composed of two layers dedicated to, respectively, multi-contact motion planning and control.

The planning layer works offline and consists of two sequential sub-planners: the *stance planner* and the *whole-body planner*. Fig. 4 illustrates the role of these two planning modules and the necessary existence of transitions to move through consecutive feasible subspaces.

The stance planner is in charge of finding three sequences

$$\mathcal{S}_\sigma = \{\sigma_0, \dots, \sigma_N\},$$
$$\mathcal{S}_q = \{\boldsymbol{q}_0, \dots, \boldsymbol{q}_N\},$$
$$\mathcal{S}_W = \{\boldsymbol{W}_{c,0}, \dots, \boldsymbol{W}_{c,N}\},$$

where $\mathcal{S}_\sigma$ is the sequence of $N+1$ stances leading to the desired final stance σ^{fin}, i.e., $\sigma_N = \sigma^{\text{fin}}$, while $\mathcal{S}_q$ and $\mathcal{S}_W$ are sequences of transitions and contact wrenches, respectively. More precisely, for each $j = 1, \dots, N$, stance $\sigma_j \in \mathcal{S}_\sigma$ is adjacent to σ_{j-1}; configuration $\boldsymbol{q}_j \in \mathcal{S}_q$ is a transition associated to stance σ_j, i.e., $\boldsymbol{q}_j \in \mathcal{D}_{\sigma,j-1} \cap \mathcal{D}_{\sigma,j}$, with $\mathcal{D}_{\sigma,j-1}$ and $\mathcal{D}_{\sigma,j}$ the feasible subspace at σ_{j-1} and σ_j, respectively; vector $\boldsymbol{W}_{c,j} \in \mathcal{S}_W$ collects the contact wrenches that guarantee static balance at $\boldsymbol{q}_j$ using the contacts specified by σ_j. Clearly, $\sigma_0 = \sigma^{\text{ini}}$, $\boldsymbol{q}_0 = \boldsymbol{q}^{\text{ini}}$, and $\boldsymbol{W}_{c,0} = \boldsymbol{W}_c^{\text{ini}}$. Note also that, in our formulation, N and $\boldsymbol{q}_N$ are not preassigned and will be autonomously determined by the stance planner.

The whole-body planner, for each pair of consecutive configurations $\boldsymbol{q}_j$ and $\boldsymbol{q}_{j+1}$ ($j = 0, \dots, N-1$) belonging to $\mathcal{S}_q$, computes a feasible whole-body motion $\boldsymbol{s}_j$, i.e., a configuration space trajectory in $\mathcal{D}_{\sigma,j}$ that connects the two and has duration δ_j, which is determined by the planner itself. Then, the result of this planner consists of a sequence of motions

$$\mathcal{S}_s = \{\boldsymbol{s}_0, \dots, \boldsymbol{s}_{N-1}\}.$$

The control layer is responsible for generating online the torque commands $\bar{\boldsymbol{\tau}}$ for the humanoid actuators to execute the planned sequence $\mathcal{S}_s$ of whole-body motions throughout the planned sequence $\mathcal{S}_\sigma$ of stances. To this end, it uses two cooperating modules: the *stance switching* and *reactive balancing* module. During execution, the control layer maintains information about the index j of the last achieved stance in the sequence $\mathcal{S}_\sigma$. Then, according to j, stances σ_j, σ_{j+1}, contact wrenches $\boldsymbol{W}_c^j$ and motion $\boldsymbol{s}_j$ are sequentially selected from the corresponding planned sequences.

To successfully accomplish the assigned task it is essential that contacts are actually established/broken in accordance with the planned stance sequence. To this end, the stance switching module computes a feedback action $\boldsymbol{\tau}_{\text{jnt}}$ aimed at realizing the motion $\boldsymbol{s}_j$ while ensuring that stance σ_{j+1} is eventually achieved despite any execution inaccuracy. In particular, to continuously monitor the achievement of σ_{j+1}, this module makes use of the measured/estimated contact wrenches $\widehat{\boldsymbol{W}}_U$ at all the end-effectors in U. Once σ_{j+1} is eventually achieved, index j is incremented.

The reactive balancing module, which works synchronously with the stance switching module, computes a feedforward action $\boldsymbol{\tau}_{\text{bal}}$ aimed at ensuring static balance by online adjusting the planned contact wrenches $\boldsymbol{W}_{c,j}$ to absorb possible external disturbances or modeling uncertainties.

Proprioceptive sensing is used for both control modules. In particular, measured joint positions $\hat{\boldsymbol{q}}_{\text{jnt}}$, velocities $\hat{\dot{\boldsymbol{q}}}_{\text{jnt}}$, and floating-base orientation $\hat{\boldsymbol{o}}_{\text{fb}}$, which is obtained through an Inertial Measurement Unit (IMU) mounted in the humanoid torso, are provided to both modules. Measured joint torques $\hat{\boldsymbol{\tau}}$ are instead provided, together with all the measurements mentioned above, to a *contact wrenches estimation* module which determines $\widehat{\boldsymbol{W}}_U$ assuming again quasi-static conditions

$$\widehat{\boldsymbol{W}}_U = (\boldsymbol{J}_{U,a}^T(\hat{\boldsymbol{q}}))^\dagger(\boldsymbol{g}_a(\hat{\boldsymbol{q}}) - \hat{\boldsymbol{\tau}}), \tag{14}$$

where $\boldsymbol{J}_{U,a}(\cdot)$ consists of the last n columns of $\boldsymbol{J}_U(\cdot)$ which stacks the Jacobian matrices associated to all end-effectors in U; both $\boldsymbol{J}_{U,a}(\cdot)$ and $\boldsymbol{g}_a(\cdot)$ are evaluated at the current humanoid configuration $\hat{\boldsymbol{q}}$.[4] Obviously, for end-effectors equipped with force/torque sensors (as in the case of COMAN+ feet), the corresponding contact wrenches in vector $\widehat{\boldsymbol{W}}_U$ are simply set to the measured values.

In the next sections, we will discuss in detail the mentioned modules constituting the proposed MCPC framework.

5. Stance planning

In this section, we propose a novel randomized method for planning the sequence $\mathcal{S}_\sigma$ of adjacent stances leading to the desired one, together with the sequences $\mathcal{S}_q$ and $\mathcal{S}_W$ of associated transitions and contact wrenches.

[4] The position of the floating-base, whose estimation is not foreseen in our framework, is left unspecified in $\hat{\boldsymbol{q}}$; the rationale being that none of the configuration-dependent terms depends on $\boldsymbol{p}_{\text{fb}}$.

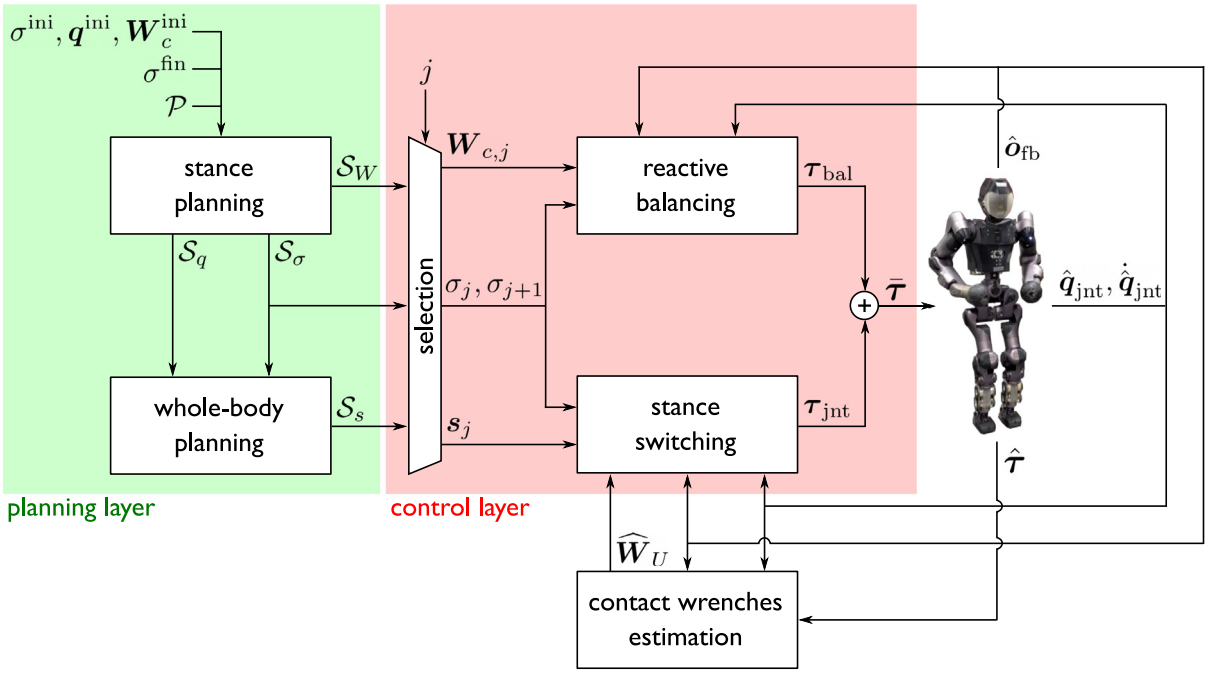


Fig. 3. Block scheme of the proposed MCPC framework. The single modules are described in Sections 5–7.

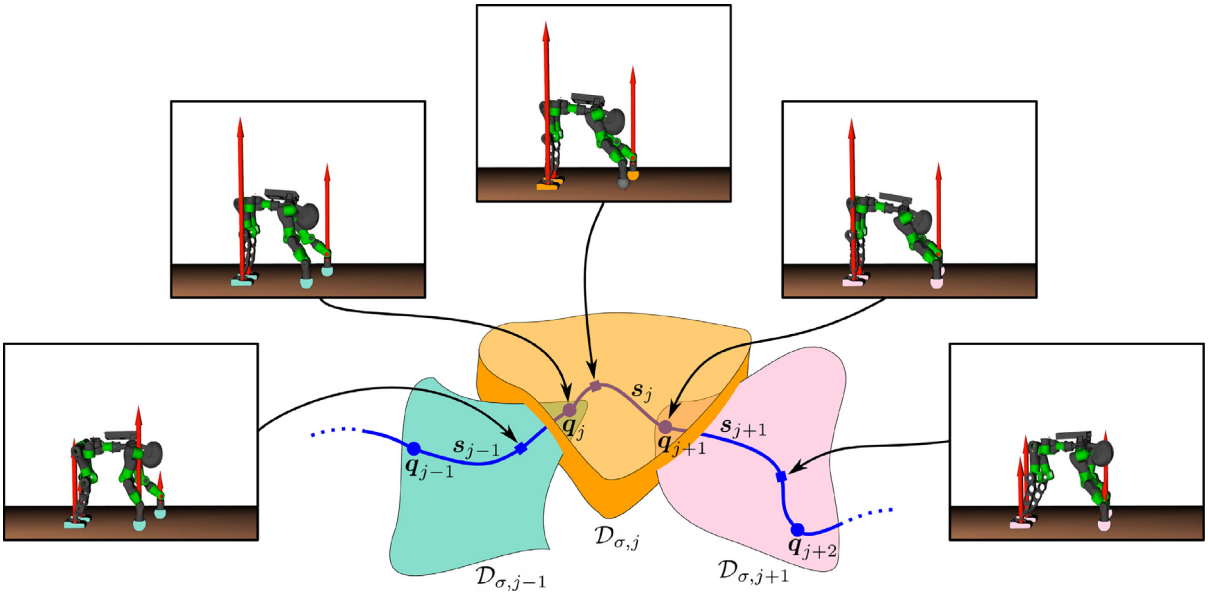


Fig. 4. Role of the two planning modules. The stance planner generates the sequences of adjacent stances, associated transitions, and contact wrenches. Here, the feasible subspaces associated with three consecutive stances are depicted in cyan, orange, and pink. Transitions (blue bullets) belong to the intersection of feasible subspaces associated with adjacent stances. The whole-body planner generates the sequence of trajectories (blue paths) between consecutive transitions. Sample configurations (blue squares) along them are also shown. In the five snapshots: the colored end-effectors satisfy the kinematic constraints that define the feasible subspace having the same color, while red arrows indicate non-null contact forces (moments are not shown). In the considered example, the robot is performing a quadrupedal walk: in particular, it is moving the right hand. Note how at transitions, four end-effectors are kinematically constrained but static balance is guaranteed using only three of them (the contact wrench exerted at the right hand is null). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To compute these sequences, our stance planner receives in input the humanoid initial stance σ^{ini}, configuration $\boldsymbol{q}^{\text{ini}}$ and contact wrenches $\boldsymbol{W}_c^{\text{ini}}$, together with the point cloud $\mathcal{P}$ and the desired final stance σ^{fin}.

5.1. Tree construction

The proposed stance planner, whose pseudocode is given in Algorithm 1, uses an RRT-like strategy to iteratively construct a tree $\mathcal{T}$ in the search space. In this tree, a vertex

$$v = \langle \sigma, \boldsymbol{q}, \boldsymbol{W}_c \rangle$$

consists of a stance σ, a configuration $\boldsymbol{q} \in \mathcal{D}_\sigma$, and a vector $\boldsymbol{W}_c \in \mathbb{R}^{6m}$ of contact wrenches, with $m = |\sigma|$. An edge going from vertex v to vertex v' indicates that σ and σ' are adjacent (in the sense formally defined in Section 2.1); consequently, $\boldsymbol{q}'$ specified in v' is a transition belonging to $\mathcal{D}_\sigma \cap \mathcal{D}_{\sigma'}$. At the beginning, the tree $\mathcal{T}$ is rooted at the vertex v^{ini} (lines 1–2). The generic

iteration of the planner consists of the four steps described in the following.

Selecting a vertex for expansion (lines 5–6): The iteration starts by choosing an end-effector $\mathcal{F}^{\text{rand}}$ and a point $\boldsymbol{p}^{\text{rand}}$. The planner is allowed to randomly choose between exploration and exploitation, to bias the growth of the tree toward unexplored regions of the search space and the goal, respectively. In the first case, $\mathcal{F}^{\text{rand}}$ and $\boldsymbol{p}^{\text{rand}}$ are randomly picked from the set U and the workspace, respectively. In the second case, a contact c^{fin} is randomly picked among those in σ^{fin}; then, $\mathcal{F}^{\text{rand}}$ and $\boldsymbol{p}^{\text{rand}}$ are chosen as $\mathcal{F}^{\text{fin}}$ and $\boldsymbol{p}_c^{\text{fin}}$, respectively.

Then, the planner assigns to each vertex v in $\mathcal{T}$ a probability that is inversely proportional to the Euclidean distance $\|\boldsymbol{k}_p^{\text{rand}}(\boldsymbol{q}) - \boldsymbol{p}^{\text{rand}}\|$ between the position of $\mathcal{F}^{\text{rand}}$ at $\boldsymbol{q}$ (the configuration specified by v) and $\boldsymbol{p}^{\text{rand}}$. The resulting probability distribution is used to randomly choose a vertex v^{near} of $\mathcal{T}$ for a tree expansion attempt. ◄

Generating a candidate stance (lines 7–15): Once v^{near} has been selected, the planner decides whether to attempt the expansion of $\mathcal{T}$ from v^{near} by removing or adding a contact. To this end, it checks if σ^{near} contains a contact involving $\mathcal{F}^{\text{rand}}$, i.e., $\mathcal{F}^{\text{rand}} \in \Psi(\sigma^{\text{near}})$. Based on the outcome of this check, a candidate stance σ^{cand} is generated (and subsequently validated) as follows.

- If $\mathcal{F}^{\text{rand}} \in \Psi(\sigma^{\text{near}})$, σ^{cand} is generated by removing from σ^{near} the contact c^{rem} involving the selected end-effector $\mathcal{F}^{\text{rand}}$. Contact c^{rem} is simply that in σ^{near} such that $\mathcal{F}^{\text{rem}} = \mathcal{F}^{\text{rand}}$.
- If $\mathcal{F}^{\text{rand}} \notin \Psi(\sigma^{\text{near}})$, σ^{cand} is generated by adding to σ^{near} a novel contact c^{add} involving the selected end-effector $\mathcal{F}^{\text{rand}}$. To build the additional contact c^{add}, the planner first identifies the portion of the workspace that the humanoid, at configuration $\boldsymbol{q}^{\text{near}}$, can reach with end-effector $\mathcal{F}^{\text{rand}}$. Such reachable workspace $\mathcal{W}$ is approximated as the set of points of the point cloud $\mathcal{P}$ that lie inside a sphere[5] centered at $\boldsymbol{k}_p^{\text{rand}}(\boldsymbol{q}^{\text{near}})$, with radius r determined by the planner itself: starting from a lower bound $r^{\min}$, it iteratively checks increasing values for r up to an upper bound $r^{\max}$, until the resulting reachable workspace $\mathcal{W}$ is not empty.[6] Bounds $r^{\min}$ and $r^{\max}$ are predefined for each end-effector according to the kinematic limits of the robot. Once the workspace $\mathcal{W}$ has been computed, the point $\boldsymbol{p}_c^{\text{add}} \in \mathcal{W}$ that is the closest to $\boldsymbol{p}^{\text{rand}}$ is selected, and c^{add} is accordingly built, with the contact orientation purposely left unspecified, as it will be automatically determined by the planner once σ^{cand} is validated. ◄

Generating an associate transition (line 16): At this point, the transition generation procedure described in Section 5.2 is invoked with the aim of producing a new stance σ^{new} and a transition $\boldsymbol{q}^{\text{new}} \in \mathcal{D}_\sigma^{\text{near}} \cap \mathcal{D}_\sigma^{\text{new}}$, with $\mathcal{D}_\sigma^{\text{near}}$ and $\mathcal{D}_\sigma^{\text{new}}$ the feasible subspace at σ^{near} and σ^{new}, respectively. In particular, the stance σ^{new} will consist of an updated version of σ^{cand} in which the orientation of contact frames involved in point contacts already present in σ^{near} (if any) and the (possibly) newly added contact c^{add} are updated to their values attained at the simultaneously generated $\boldsymbol{q}^{\text{new}}$. ◄

Adding a new vertex (lines 17–21): If the transition generation procedure succeeds, σ^{near} and σ^{new} are adjacent, as both the related conditions (see Section 2.1) are satisfied. In fact, σ^{new} differs from σ^{near} by a single contact (by construction of σ^{cand}), and $\mathcal{D}_\sigma^{\text{near}} \cap \mathcal{D}_\sigma^{\text{new}} \neq \emptyset$ thanks to the existence of $\boldsymbol{q}^{\text{new}}$. The procedure described in Section 5.3 is invoked to compute the vector $\boldsymbol{W}_c^{\text{new}}$, collecting the contact wrenches that guarantee static balance at $\boldsymbol{q}^{\text{new}}$ using the contacts specified by σ^{new}. A new vertex v^{new} is then created and added in tree $\mathcal{T}$ as a child of v^{near}. ◄

Construction of $\mathcal{T}$ stops when the desired final stance is reached, i.e., $\sigma^{\text{new}} = \sigma^{\text{fin}}$, or a maximum number $l_\sigma^{\max}$ of iterations has been performed. In the first case, the sequences $\mathcal{S}_\sigma$, $\mathcal{S}_q$, and $\mathcal{S}_W$ are directly retrieved from the vertices along the branch of $\mathcal{T}$ joining the root vertex v^{ini} to v^{new} (line 25) and passed to the whole-body planner. In the second case, the stance planner returns a failure.

Algorithm 1: Stance Planner

```
1  v^ini ← ⟨σ^ini, q^ini, W_c^ini⟩;
2  AddVertex(T, v^ini, ∅);
3  l ← 0;
4  repeat
5      [F^rand, p^rand] ← PickRandom();
6      v^near ← FindNearestVertex(T, F^rand, p^rand);
7      if F^rand ∈ Ψ(σ^near) then
8          c^rem ← ExtractContact(σ^near, F^rand);
9          σ^cand ← σ^near \ c^rem;
10     else
11         W ← ComputeReachableWorkspace(q^near, F^rand);
12         p_c^add ← PickRandomPoint(W, p^rand);
13         c^add ← ⟨φ(F^rand), F^rand, [(p_c^add)^T, ∅]^T⟩;
14         σ^cand ← σ^near ∪ c^add;
15     end
16     [σ^new, q^new] ← GenerateTransition(σ^near, σ^cand, q^near);
17     if q^new ≠ ∅ then
18         W_c^new ← ComputeContactWrenches(σ^new, q^new);
19         v^new ← ⟨σ^new, q^new, W_c^new⟩;
20         AddVertex(T, v^new, v^near);
21     end
22     l ← l + 1;
23 until σ^new = σ^fin or l = l_σ^max;
24 if σ^new = σ^fin then
25     [S_σ, S_q, S_W] ← RetrieveSolution(T);
26     return [S_σ, S_q, S_W];
27 end
28 return [∅, ∅, ∅];
```

5.2. Transition generation

The proposed transition generator consists of an adaptation of the method presented in [41] for the specific case in which adjacency conditions between stances must be accounted for. It receives in input the stance σ^{near}, its associated transition $\boldsymbol{q}^{\text{near}}$, and the candidate stance σ^{cand}. The aim is to produce a configuration $\boldsymbol{q}^{\text{cand}}$ that belongs to $\mathcal{D}_\sigma^{\text{near}} \cap \mathcal{D}_\sigma^{\text{cand}}$, and is close to $\boldsymbol{q}^{\text{near}}$.

Let σ^{lar} and σ^{sma} be, respectively, the largest and smallest stance between σ^{near} and σ^{cand}. Denote by $\mathcal{C}_\sigma^{\text{lar}}$ and $\mathcal{C}_\sigma^{\text{sma}}$ their associated stance submanifolds, and by $\mathcal{D}_\sigma^{\text{lar}}$ and $\mathcal{D}_\sigma^{\text{sma}}$ their corresponding feasible subspaces. Recall that, a configuration $\boldsymbol{q}$ belongs to $\mathcal{D}_\sigma^{\text{lar}} \cap \mathcal{D}_\sigma^{\text{sma}}$ if it (*i*) satisfies the $|\sigma^{\text{lar}}|$ kinematic constraints yielded by σ^{lar}, (*ii*) respects the joint limits, (*iii*) avoids (self-)collisions, and (*iv*) guarantees humanoid static balance using the $|\sigma^{\text{sma}}|$ contacts in σ^{sma}, all at the same time.

Our transition generator, whose pseudo-code is given in Procedure 1, works in an iterative fashion by repeatedly invoking an Inverse Kinematics (IK) solver. In the following, we first present

[5] More sophisticated methods for approximating the reachable workspace (e.g., [39,40]) can be involved without affecting the overall planning strategy.

[6] In case the upper bound $r^{\max}$ is exceeded, the current expansion attempt is aborted, and the planner starts a new iteration. For the sake of illustration, this is not explicitly shown in Algorithm 1.

Procedure 1: GenerateTransition(σ^{near}, σ^{cand}, $\boldsymbol{q}^{\text{near}}$)

1 **if** $|\sigma^{\text{cand}}| > |\sigma^{\text{near}}|$ **then**
2 $\quad \sigma^{\text{lar}} \leftarrow \sigma^{\text{cand}}$, $\sigma^{\text{sma}} \leftarrow \sigma^{\text{near}}$;
3 **else**
4 $\quad \sigma^{\text{lar}} \leftarrow \sigma^{\text{near}}$, $\sigma^{\text{sma}} \leftarrow \sigma^{\text{cand}}$;
5 **end**
6 $\boldsymbol{q}^{\text{nom}} \leftarrow$ SolveIK(σ^{lar}, $\boldsymbol{q}^{\text{near}}$, $\boldsymbol{q}^{\text{near}}$);
7 $\boldsymbol{q}^{\text{cand}} \leftarrow \boldsymbol{q}^{\text{nom}}$;
8 $l \leftarrow 0$;
9 **while not** (CollisionFree($\boldsymbol{q}^{\text{cand}}$) **and** Balanced($\boldsymbol{q}^{\text{cand}}$, σ^{sma})) **do**
10 $\quad$ **if** mod(l, $l^{\max}_{\text{tran}}$) $= 0$ **then**
11 $\quad\quad \boldsymbol{q}^{\text{ref}} \leftarrow \boldsymbol{q}^{\text{nom}}$;
12 $\quad\quad \dot{\boldsymbol{q}}^{\text{rand}} \leftarrow$ GenerateRandomVelocity();
13 $\quad$ **end**
14 $\quad \boldsymbol{q}^{\text{ref}} \leftarrow$ Integrate($\boldsymbol{q}^{\text{ref}}$, $\dot{\boldsymbol{q}}^{\text{rand}}$);
15 $\quad \boldsymbol{q}^{\text{cand}} \leftarrow$ SolveIK(σ^{lar}, $\boldsymbol{q}^{\text{cand}}$, $\boldsymbol{q}^{\text{ref}}$);
16 $\quad l \leftarrow l + 1$;
17 **end**
18 $\sigma^{\text{cand}} \leftarrow$ UpdateStance(σ^{cand}, $\boldsymbol{q}^{\text{cand}}$);
19 **return** [σ^{cand}, $\boldsymbol{q}^{\text{cand}}$];

the overall transition generation procedure and then discuss the details of the IK solver.

5.2.1. Procedure

Once σ^{lar} and σ^{sma} have been identified between σ^{near} and σ^{cand} (line 1–5), the procedure invokes the IK solver (line 6) to generate a nominal configuration $\boldsymbol{q}^{\text{nom}}$ that complies with requirements (*i*)-(*ii*) and is as close as possible to $\boldsymbol{q}^{\text{near}}$. Such $\boldsymbol{q}^{\text{nom}}$ represents the initial candidate configuration $\boldsymbol{q}^{\text{cand}}$.

Then, the procedure enters a cycle that repeatedly generates new versions of $\boldsymbol{q}^{\text{cand}}$, exploiting the humanoid kinematic redundancy, to explore $\mathcal{C}^{\text{lar}}_{\sigma}$ with the objective of finding a configuration that also satisfies requirements (*iii*)-(*iv*). The generic iteration starts by checking whether the current $\boldsymbol{q}^{\text{cand}}$ satisfies such requirements (line 9). In particular, to evaluate (*iv*), the procedure described in Section 5.3 is invoked. It verifies if there exists a vector $\boldsymbol{W}^{\text{cand}}_{c}$, collecting the contact wrenches that guarantee static balance at $\boldsymbol{q}^{\text{cand}}$ using the contacts specified by σ^{sma}.

In case (*iii*)-(*iv*) are not both satisfied, the IK solver is invoked (line 15) to generate a new version of $\boldsymbol{q}^{\text{cand}}$ that complies with requirements (*i*)-(*ii*) and is as close as possible to a reference configuration $\boldsymbol{q}^{\text{ref}}$, which is obtained by numerically integrating a constant velocity $\dot{\boldsymbol{q}}^{\text{rand}}$ (line 14). Every $l^{\max}_{\text{tran}}$ iterations (line 10), $\boldsymbol{q}^{\text{ref}}$ is reset to $\boldsymbol{q}^{\text{nom}}$ and a new velocity $\dot{\boldsymbol{q}}^{\text{rand}}$ is randomly generated with bounded norm (line 12). With this strategy, $\dot{\boldsymbol{q}}^{\text{rand}}$ will be maintained fixed over $l^{\max}_{\text{tran}}$ iterations, while $\boldsymbol{q}^{\text{ref}}$ will consequently change in the neighborhood of $\boldsymbol{q}^{\text{nom}}$. This, combined with the fact that $\boldsymbol{q}^{\text{nom}}$ is generated as close as possible to $\boldsymbol{q}^{\text{near}}$, aids the overall planning layer to produce qualitatively graceful motions. More informed strategies for generating the velocity $\dot{\boldsymbol{q}}^{\text{rand}}$ can also be employed in our transition generator; for example, one may generate a non-null velocity only for those kinematic chains that are in collision at the current $\boldsymbol{q}^{\text{cand}}$. For further details about efficient strategies for selecting $\dot{\boldsymbol{q}}^{\text{rand}}$ we refer the reader to [41].

In case (*iii*)-(*iv*) are both satisfied, the cycle is left and, for each $c_i \in \sigma^{\text{cand}}$, the procedure updates the pose $\boldsymbol{r}_{c,i}$ of contact frame $\mathcal{F}_i$ to $\boldsymbol{k}_i(\boldsymbol{q}^{\text{cand}})$ (line 18). Then, the updated stance σ^{cand} and the configuration $\boldsymbol{q}^{\text{cand}}$, which now represents a transition in $\mathcal{D}^{\text{near}}_{\sigma} \cap \mathcal{D}^{\text{new}}_{\sigma}$, are returned to the stance planner.

The cycle is interrupted as soon as a predefined time budget ΔT_{tran} expires without finding a transition or the IK solver returns a failure. In these cases, which for sake of illustration are not shown in Procedure 1, a failure is reported to the stance planner.

5.2.2. IK solver

At the generic invocation, the IK solver is provided with a stance σ, a starting configuration $\boldsymbol{q}^{\text{start}}$, and a reference configuration $\boldsymbol{q}^{\text{ref}}$. It aims to compute a configuration $\boldsymbol{q}^{\text{cand}}$ that lies in $\mathcal{C}_{\sigma}$ and is as close as possible to $\boldsymbol{q}^{\text{ref}}$. To compute $\boldsymbol{q}^{\text{cand}}$, the IK solver proceeds iteratively; at each iteration, it numerically integrates the velocities $\dot{\boldsymbol{q}}$ produced by solving two sequential QP problems. Integration starts from $\boldsymbol{q}^{\text{start}}$.

The first QP problem is

$$\begin{cases} \min\limits_{\dot{\boldsymbol{q}}} \ \|\boldsymbol{J}_{\sigma}(\boldsymbol{q})\dot{\boldsymbol{q}} - \boldsymbol{K}_{\sigma}\boldsymbol{e}_{\sigma}(\boldsymbol{q})\|^2 + \alpha\|\dot{\boldsymbol{q}}\|^2 \\ \text{subject to:} \\ \bullet \ \text{joint limits constraint} \end{cases} \tag{15}$$

where the first term of the cost function aims at fulfilling the kinematic constraints yielded by σ, while the second is included for regularization purposes. Here, α is a positive scalar, $\boldsymbol{K}_{\sigma}$ a positive definite matrix, $\boldsymbol{J}_{\sigma}(\boldsymbol{q})$ and $\boldsymbol{e}_{\sigma}(\boldsymbol{q})$ given by

$$\boldsymbol{J}_{\sigma}(\boldsymbol{q}) = [\boldsymbol{J}^T_{\sigma,1}(\boldsymbol{q}), \ldots, \boldsymbol{J}^T_{\sigma,m}(\boldsymbol{q})]^T$$

and

$$\boldsymbol{e}_{\sigma}(\boldsymbol{q}) = [\boldsymbol{e}^T_{\sigma,1}(\boldsymbol{q}), \ldots, \boldsymbol{e}^T_{\sigma,m}(\boldsymbol{q})]^T,$$

with $m = |\sigma|$. Moreover, $\boldsymbol{J}_{\sigma,i}(\boldsymbol{q})$ and $\boldsymbol{e}_{\sigma,i}(\boldsymbol{q})$ are defined according to the contact $c_i \in \sigma$ and the current configuration $\boldsymbol{q}$, respectively, as

$$\boldsymbol{J}_{\sigma,i}(\boldsymbol{q}) = \begin{cases} \boldsymbol{J}_{p,i}(\boldsymbol{q}), & \text{if } \mathtt{t}_i = \mathtt{P} \text{ or } \boldsymbol{o}_{c,i} = \emptyset \\ \boldsymbol{J}_i(\boldsymbol{q}), & \text{otherwise,} \end{cases}$$

and

$$\boldsymbol{e}_{\sigma,i}(\boldsymbol{q}) = \begin{cases} \boldsymbol{p}_{c,i} - \boldsymbol{k}_{p,i}(\boldsymbol{q}), & \text{if } \mathtt{t}_i = \mathtt{P} \text{ or } \boldsymbol{o}_{c,i} = \emptyset \\ \boldsymbol{r}_{c,i} - \boldsymbol{k}_i(\boldsymbol{q}), & \text{otherwise.} \end{cases}$$

Note the following points.

- If the stance planner generated σ^{cand} by adding a contact to σ^{near}, the stance σ provided to the IK solver coincides with σ^{cand}. In this case, the orientation of the contact frame corresponding to the added contact (even in case it is a surface one) is not accounted for in the cost function, letting the IK solver explore the different possibilities.
- In view of the choice made about the contact frames placement (see assumption A2, Section 4), for a configuration $\boldsymbol{q} \in \mathcal{C}_{\sigma}$ to be collision-free it is necessary (but not sufficient) that the z-axis of each contact frame $\mathcal{F}_i$ involved either in a point or surface contact $c_i \in \sigma$, is coincident with the normal at point $\boldsymbol{p}_{c,i}$ pointing from the environment to the robot. Based on this observation, in order to increase the chances of the IK solver finding a collision-free configuration, we included in the cost function of the first QP an extra term that attempts to align the z-axis of those contact frames involved in σ whose orientation is either not explicitly constrained (i.e., point contacts) or unspecified (i.e., an added surface contact) with the associated normal. For sake of illustration, we omit the formulation of such a term, whose structure is similar to that of the first.

The second QP problem is

$$\begin{cases} \min\limits_{\dot{\boldsymbol{q}}} \ \|\dot{\boldsymbol{q}} - \boldsymbol{K}_q\boldsymbol{e}_q(\boldsymbol{q})\|^2 \\ \text{subject to:} \\ \bullet \ \boldsymbol{J}_{\sigma}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{J}_{\sigma}(\boldsymbol{q})\dot{\boldsymbol{q}}^*_1 \\ \bullet \ \text{joint limits constraint} \end{cases}$$

and attempts to bring the current configuration $\boldsymbol{q}$ toward the reference one $\boldsymbol{q}^{\text{ref}}$. Here, $\boldsymbol{K}_q$ is a positive definite matrix, $\boldsymbol{e}_q(\boldsymbol{q}) = \boldsymbol{q}^{\text{ref}} - \boldsymbol{q}$, and $\dot{\boldsymbol{q}}_1^*$ is the solution of the first QP. The first constraint ensures that the minimization of the cost function produces a solution that does not perturb the objectives pursued by the first QP. The joint limits constraint included in both QP problems is readily obtained by rewriting (3) in terms of the decision variables $\dot{\boldsymbol{q}}$.

The IK solver stops integrating when the norm of $\boldsymbol{e}_\sigma(\boldsymbol{q})$ lowered a small threshold ε_σ or a maximum number $I_{\text{IK}}^{\max}$ of integration steps have been performed. In the first case, the kinematic constraints yielded by σ are considered satisfied, and the last obtained configuration is returned to the transition generator; in the second case σ is considered kinematically unrealizable, and a failure is returned to the transition generator.

It is worth noticing that a solution to the two QP problems above will always exist. Indeed, the first QP is initialized at the starting configuration $\boldsymbol{q}^{\text{start}}$ which surely satisfies the unique constraint, in view of the fact that it is a configuration extracted from the tree $\mathcal{T}$ (see Section 5.1); similar arguments holds for the second QP problem, whose additional constraint is trivially satisfied by the solution of the first QP. By construction, the QP problems formulated at the successive integration steps will be feasible as well.

5.3. Contact wrenches computation

Given a generic stance σ and a configuration $\boldsymbol{q} \in \mathcal{C}_\sigma$, in order to guarantee static balance, the vector $\boldsymbol{W}_c$ collecting the contact wrenches at all $c_i \in \sigma$ must satisfy the conditions C1–C4 described in Section 2.2. To compute such $\boldsymbol{W}_c$, we solve the following QP problem

$$\begin{cases} \min\limits_{\boldsymbol{W}_c} \ \|\boldsymbol{g}_u(\boldsymbol{q}) - \boldsymbol{J}_{c,u}^T(\boldsymbol{q})\boldsymbol{W}_c\|^2 + \beta\|\boldsymbol{W}_c\|^2 \\ \text{subject to:} \\ \bullet \text{ torque limits constraint (16)} \\ \bullet \text{ friction cone constraints (11)} \\ \bullet \text{ CoP constraints (12)} \\ \bullet \text{ yaw moment constraints (13)} \end{cases}$$

Here, the four kinds of constraints emerge from the four conditions C1–C4, in the same order. In particular, we account separately for the centroidal statics condition (C1) on the underactuated and actuated parts of the system. For the underactuated part, the condition is formulated as a soft objective by the first term of the cost function (see (7)), in which the second term is instead included for regularization purposes. For the actuated part, the condition results in a simple constraint on the actuated torques (first constraint) of the form

$$\boldsymbol{\tau}^{\min} \le \boldsymbol{g}_a(\boldsymbol{q}) - \boldsymbol{J}_{c,a}^T(\boldsymbol{q})\boldsymbol{W}_c \le \boldsymbol{\tau}^{\max} \tag{16}$$

that is obtained by rewriting (9) in terms of the decision variables. Note that, with this formulation the actuated torques $\boldsymbol{\tau}$ are not explicitly treated as decision variables in the QP, as they are not required during the planning stage.

For each contact in $c_i \in \sigma$, while a friction cone constraint is enforced regardless of the contact type, a CoP and a yaw-moment constraint are enforced only if it is a surface contact.

Recall that the computation of contact wrenches is required within our stance planner both to check the static balance at a candidate transition $\boldsymbol{q}^{\text{cand}}$ (line 9 of Procedure 1), and obtain the remaining information needed to construct a new vertex once a transition $\boldsymbol{q}^{\text{new}}$ has been generated (line 19 of Algorithm 1). While in the second case a solution for the QP certainly exists (since $\boldsymbol{q}^{\text{new}}$ is a transition), in the first case the QP could prove infeasible. If this is the case, static balance is not satisfied at $\boldsymbol{q}^{\text{cand}}$ and the procedure continues generating a new candidate transition. Otherwise, once the QP is solved, it is verified that the residual value of the first term of the cost function is below a small threshold ε_u, situation in which static balance at $\boldsymbol{q}^{\text{cand}}$ is considered satisfied

A remark is in order here about the choice of formulating the centroidal statics condition on the underactuated subsystem as a term of the cost function, rather than as a constraint. With this choice, since the resulting QP always admits a solution, the transition generator is more likely to validate candidate transitions, but in a weaker sense with respect to C1, depending on the chosen value for ε_u. However, the planned contact wrenches will act as suitable reference values for the controller, which will appropriately adjust them in order to ensure reactive balance.

6. Whole-body planning

The whole-body planner receives in input the sequences $\mathcal{S}_\sigma$ and $\mathcal{S}_q$ of, respectively, stances and associated transitions, produced by the stance planner. In the output, it provides the sequence $\mathcal{S}_s$ of whole-body motions between consecutive transitions.

The pseudocode of the whole-body planner is given in Algorithm 2. For each pair of consecutive configurations $\boldsymbol{q}_j$ and $\boldsymbol{q}_{j+1}$ ($j = 0, \dots, N-1$) in $\mathcal{S}_q$, a configuration space trajectory that connects the two is planned. To this end, the two-stage approach presented in [20] is adopted. First, a path consisting of a sequence $\mathcal{S}_{q,j}$ of via points (configurations) in $\mathcal{D}_{\sigma,j}$ joining $\boldsymbol{q}_j$ to $\boldsymbol{q}_{j+1}$ is found (line 3); then, a continuous (with continuous first time derivative) trajectory

$$\boldsymbol{s}_j(t), \quad t \in [t_j, t_j + \delta_j),$$

which interpolates the configurations in $\mathcal{S}_{q,j}$ is computed (line 4), simultaneously determining its duration δ_j. Here, t_j indicates an arbitrary time instant at which execution of $\boldsymbol{s}_j$ will start during the control phase. These two stages are separately discussed in detail in the next two subsections.

The sequentially generated trajectories form the sequence $\mathcal{S}_s$. These, together with the sequences $\mathcal{S}_W$ and $\mathcal{S}_\sigma$, produced by the stance planner, are finally sent to the control layer.

6.1. Via points computation

Given two consecutive configurations $\boldsymbol{q}_j, \boldsymbol{q}_{j+1}$ in $\mathcal{S}_q$, and stance σ_j in $\mathcal{S}_\sigma$, the connecting sequence

$$\mathcal{S}_{q,j} = \{\boldsymbol{q}_{j,0}, \dots, \boldsymbol{q}_{j,M_j}\},$$

where $\boldsymbol{q}_{j,0} = \boldsymbol{q}_j$, $\boldsymbol{q}_{j,M_j} = \boldsymbol{q}_{j+1}$, and each generic element $\boldsymbol{q}_{j,k}$ ($k = 0, \dots, M_j$) belongs to $\mathcal{D}_{\sigma,j}$, is computed using the AtlasRRT* method [42].

While constructing a tree of configurations belonging to $\mathcal{D}_{\sigma,j}$ using an RRT*-based strategy, the algorithm incrementally builds an atlas of the stance submanifold $\mathcal{C}_{\sigma,j}$ defined by σ_j, i.e., a collection of charts, each one represented by a tangent space that locally approximates $\mathcal{C}_{\sigma,j}$ to a Euclidean space. Hence, the advantage of adopting such a method is twofold: first, its asymptotic-optimality property allows to minimize the length of the path between $\boldsymbol{q}_j$ and $\boldsymbol{q}_{j+1}$; second, it efficiently generates new configurations by directly sampling $\mathcal{C}_{\sigma,j}$ using its atlas.

Every time a new configuration is generated, the algorithm verifies that it belongs to the feasible subspace $\mathcal{D}_{\sigma,j}$ before adding it to the current tree. To this end, the satisfaction of joint limits, collision avoidance, and static balance is checked. The latter check

Algorithm 2: Whole-Body Planner

```
S_s ← {};
for j ← 0 to N − 1 do
    S_{q,j} ← ComputeViaPoints(q_j, q_{j+1}, σ_j);
    s_j(t), t ∈ [t_j, t_j + δ_j) ← OptimizeTrajectory(S_{q,j});
    S_s ← S_s ∪ {s_j};
end
return S_s;
```

is performed using the method discussed in Section 5.3, providing the generated configuration and stance σ_j.

The algorithm is allowed to run for a predefined time budget ΔT_{via}, at the end of which the solution sequence $\mathcal{S}_{q,j}$ is retrieved from the constructed tree and passed to the trajectory optimization procedure.

6.2. Trajectory optimization

The trajectory $\boldsymbol{s}_j(t)$ interpolating the $M_j + 1$ configurations in $\mathcal{S}_{q,j}$ is generated by finding a spline consisting of the concatenation of M_j cubic polynomials, i.e.,

$$\boldsymbol{s}_j(t) = \boldsymbol{\varrho}_{j,k}(t), \quad t \in [t_{j,k}, t_{j,k+1} = t_{j,k} + \delta_{j,k}), \tag{17}$$

with $k = 0, \dots, M_j - 1$, $t_{j,0} = t_j$, and $\delta_{j,k}$ the duration of the kth polynomial (subtrajectory), which is defined as

$$\boldsymbol{\varrho}_{j,k}(t) = \sum_{l=0}^{3} \boldsymbol{\lambda}_{j,k,l}(t - t_{j,k})^l, \tag{18}$$

with $\boldsymbol{\lambda}_{j,k,l}$ $(l = 0, \dots, 3)$ its coefficients. Moreover, according to (17), the overall duration of trajectory $\boldsymbol{s}_j(t)$ will be

$$\delta_j = \sum_{k=0}^{M_j-1} \delta_{j,k}. \tag{19}$$

Collect in vectors

$$\boldsymbol{\zeta}_j = [\delta_{j,0}, \dots, \delta_{j,M_j-1}]^T,$$
$$\boldsymbol{\xi}_j = [\boldsymbol{\lambda}_{j,0,0}^T, \dots, \boldsymbol{\lambda}_{j,0,3}^T, \dots, \boldsymbol{\lambda}_{j,M_j-1,0}^T, \dots, \boldsymbol{\lambda}_{j,M_j-1,3}^T]^T,$$

the durations and coefficients of all polynomials. To obtain these, the following nonlinear programming (NLP) problem is solved

$$\min_{\boldsymbol{\zeta}_j, \boldsymbol{\xi}_j} \ \|\boldsymbol{\zeta}_j\|^2 + \gamma \|\boldsymbol{\xi}_j\|^2$$

subject to:

$$\boldsymbol{\zeta}_j \geq \boldsymbol{0}, \tag{20a}$$
$$\boldsymbol{\varrho}_{j,k}(t_{j,k}) = \boldsymbol{q}_{j,k}, \ k = 0, \dots, M_j - 1, \tag{20b}$$
$$\boldsymbol{\varrho}_{j,k}(t_{j,k+1}) = \boldsymbol{q}_{j,k+1}, \ k = 0, \dots, M_j - 1, \tag{20c}$$
$$\dot{\boldsymbol{\varrho}}_{j,k}(t_{j,k+1}) = \dot{\boldsymbol{\varrho}}_{j,k+1}(t_{j,k+1}), \ k = 0, \dots, M_j - 2, \tag{20d}$$
$$\dot{\boldsymbol{\varrho}}_{j,0}(t_{j,0}) = \boldsymbol{0}, \tag{20e}$$
$$\dot{\boldsymbol{\varrho}}_{j,M_j-1}(t_{j,M_j}) = \boldsymbol{0}, \tag{20f}$$
$$\dot{\boldsymbol{q}}_{\text{jnt}}^{\min} \leq \boldsymbol{S}^T \dot{\boldsymbol{\varrho}}_{j,k}(t_{j,k}) \leq \dot{\boldsymbol{q}}_{\text{jnt}}^{\max}, \ k = 1, \dots, M_j - 1, \tag{20g}$$
$$\ddot{\boldsymbol{q}}_{\text{jnt}}^{\min} \leq \boldsymbol{S}^T \ddot{\boldsymbol{\varrho}}_{j,k}(t_{j,k}) \leq \ddot{\boldsymbol{q}}_{\text{jnt}}^{\max}, \ k = 1, \dots, M_j - 1. \tag{20h}$$

Here, the first term of the cost function aims at minimizing the duration of the resulting trajectory, while the second is included for regularization purposes. The constraints enforce, respectively, non-negativity of the durations (20a), passage through the via points (20b)–(20c), continuity of the velocity at the internal via points (20d), null velocity at the initial and final points (20e)–(20f), limits on the resulting joint velocities and accelerations (20g)–(20h), with matrix $\boldsymbol{S}$ defined in (6).

Note the following points.

- Despite a large number of decision variables, the presented NLP can be rapidly solved thanks to its sparse structure.
- It can be reasonably assumed that each sub-trajectory $\boldsymbol{\varrho}_{j,k}(t)$ is completely contained in $\mathcal{D}_{\sigma,j}$ when using a small stepsize in the AtlasRRT* employed in the first stage, as consecutive configurations $\boldsymbol{q}_{j,k}$, $\boldsymbol{q}_{j,k+1}$ in $\mathcal{S}_{q,j}$ will be close to each other.

7. Control layer

The control layer adopted in the presented framework is mainly based on authors' previous work [20,32,36]. In the following, we briefly discuss it as a possible option for executing the references produced through the offline planning phase. However, other implementations of the control layer, e.g. based on full-body MPC [43] or Task Space Inverse Dynamics [44], – which is not the main focus of this paper – can potentially be involved.

As anticipated in Section 4, a naive feedforward execution of the planned motions $\mathcal{S}_s$ is generally not sufficient to guarantee the successful completion of the task. This typically occurs due to environment and model inaccuracy, which can lead to an early or late contact establishment/removal compromising the execution of the whole motion. For this reason, proprioceptive and exteroceptive sensing must be used, closing the control loop and increasing the robustness of the overall framework.

While moving between two adjacent stances σ_j and σ_{j+1}, the control layer is in charge to track both the planned motion $\boldsymbol{s}_j$ and the contact wrench $\boldsymbol{W}_{c,j}$. This is done by generating a torque reference $\bar{\boldsymbol{\tau}}$ that takes into account two contributions:

$$\bar{\boldsymbol{\tau}} = \boldsymbol{\tau}_{\text{bal}} + \boldsymbol{\tau}_{\text{jnt}}. \tag{21}$$

The first term $\boldsymbol{\tau}_{\text{bal}}$ in (21) is a feedforward term to track the planned contact wrench $\boldsymbol{W}_{c,j}$ while moving toward the next stance σ_{j+1}. This is produced by the reactive balancing module by solving the following QP problem

$$\min_{\boldsymbol{W}_c} \ \|\boldsymbol{W}_c - \boldsymbol{W}_{c,j}\|^2$$

subject to:

- centroidal statics constraint (7)
- torque limits constraint (16)
- friction cone constraints (11)
- CoP constraints (12)
- yaw moment constraints (13)

Such QP is obtained by minor modifications of that used to compute the contact wrenches during the planning phase (see Section 5.3). Differently from the latter, here the centroidal statics condition on the underactuated subsystem is formulated as a constraint, while the cost function simply attempts to keep the contact wrenches as close as possible to the planned ones.

Then, $\boldsymbol{\tau}_{\text{bal}}$ is computed as:

$$\boldsymbol{\tau}_{\text{bal}} = \boldsymbol{g}_a(\hat{\boldsymbol{q}}) - \boldsymbol{J}_{c,a}^T(\hat{\boldsymbol{q}})\overline{\boldsymbol{W}}_c, \tag{22}$$

in such a way to realize the computed contact wrenches $\overline{\boldsymbol{W}}_c$ and compensate for the robot links gravity, under quasi-static conditions. In (22), the term $\hat{\boldsymbol{q}} = [\boldsymbol{0}_{3\times1}^T \quad \hat{\boldsymbol{o}}_{\text{fb}}^T \quad \boldsymbol{q}_{\text{jnt}}^T]^T$ is the current humanoid configuration, which is reconstructed using the IMU base orientation measurement $\hat{\boldsymbol{o}}_{\text{fb}}$, and the measured joint position $\boldsymbol{q}_{\text{jnt}}$.

The second term $\boldsymbol{\tau}_{\text{jnt}}$ in (21) consists instead in a feedback term for tracking the planned motion $\boldsymbol{s}_j$ and is produced by the stance switching module as

$$\boldsymbol{\tau}_{\text{jnt}} = \boldsymbol{K}_P(\bar{\boldsymbol{q}}_{\text{jnt}} - \hat{\boldsymbol{q}}_{\text{jnt}}) + \boldsymbol{K}_D(\dot{\bar{\boldsymbol{q}}}_{\text{jnt}} - \dot{\hat{\boldsymbol{q}}}_{\text{jnt}}), \tag{23}$$

with $\bar{\boldsymbol{q}}_{\text{jnt}}$ and $\dot{\bar{\boldsymbol{q}}}_{\text{jnt}}$ being the joint position and velocity references extracted from the planned motion $\boldsymbol{s}_j$.

After the execution of $\boldsymbol{s}_j$, to ensure that the stance σ_{j+1} is eventually achieved, a finishing motion is performed in case $c^{\text{diff}} = \langle \mathfrak{t}^{\text{diff}}, \mathcal{F}^{\text{diff}}, \boldsymbol{r}_c^{\text{diff}} \rangle$, i.e., the contact by which σ_j and σ_{j+1} differ, failed to be correctly established/broken. To this end, the stance switching module continuously checks if the measured/estimated contact wrench $\boldsymbol{w}^{\text{diff}}$ (extracted from $\hat{\boldsymbol{W}}_U$) at c^{diff} reached a certain threshold. In particular, when $\sigma_{j+1} \supset \sigma_j$ ($\sigma_{j+1} \subset \sigma_j$), c^{diff} is considered correctly established (broken) if $\|\hat{\boldsymbol{w}}^{\text{diff}}\| \geq w^{\max}$ ($\|\hat{\boldsymbol{w}}^{\text{diff}}\| \leq w^{\min}$). While such check is not passed, the references $\bar{\boldsymbol{q}}_{\text{jnt}}$ and $\dot{\bar{\boldsymbol{q}}}_{\text{jnt}}$ for (23) are generated via kinematic control using a scheme similar to that discussed in Section 5.2.2 with the idea of imposing a linear velocity $\dot{\boldsymbol{p}}^{\text{ref}}$ to end-effector $\mathcal{F}^{\text{diff}}$ to make it advance/retract toward/from the environment surface on which the contact must be established/broken while maintaining all the other contacting end-effectors at their current pose. This is obtained by simply choosing $\dot{\boldsymbol{p}}^{\text{ref}} = \pm \boldsymbol{K}_c \boldsymbol{n}_c$, with $\boldsymbol{K}_c$ a positive definite matrix, $\boldsymbol{n}_c$ the unit normal at point $\boldsymbol{p}_c^{\text{diff}}$ specified by $\boldsymbol{r}_c^{\text{diff}}$, and the sign positive/negative if the contact must be established/broken.

8. Implementation details

Our implementation of the proposed multi-contact motion planning and control framework relies on various tools which belong to the ROS ecosystem; those tools are briefly discussed in the following.

Computations related to inverse kinematics and contact wrenches (Sections 5.2.2, 7, and 5.3) are managed through the *CartesI/O* framework [45] which relies on the *OpenSoT* library [46] for the formulation and resolution of the corresponding QP problems. In particular, such problems can be defined through a simple syntax called *Math of Tasks* [47] and solved using efficient QP solvers such as *qpOASES* [48] or *OSQP* [49]. Specifically, in our validation, we adopted the active-set method provided by *qpOASES*.

In the whole-body planner, the first stage (Section 6.1) is realized using the implementation of AtlasRRT* provided by the *OMPL* library [50], which is appropriately customized in order to include the described feasibility checks. The NLP constituting the second stage (Section 6.2) is formulated using *CasADi* [51] and solved via *IPOPT* [52].

Collision checks, in both the stance and whole-body planners, are performed using the *Planning Scene* component of the *MoveIT!* framework [53], which exploits the *Flexible Collision Library* (*FCL*) [54]. To this purpose, the point cloud $\mathcal{P}$ is converted into an *OctoMap* [55], encoding free and occupied space, which is compared with the link meshes from the robot *URDF*.

Finally, we have also developed a UI, still based on *CartesI/O*, that allows the user to perform two important operations using RVIZ as a visualization interface: the creation of desired final, or even intermediate (see Section 9.1), stances describing the tasks assigned to the humanoid, and the inspection of solutions found by the single modules of the planning layer.

9. Validation

The validation of the proposed MCPC framework has been performed using COMAN+, a torque-controlled humanoid robot designed at Istituto Italiano di Tecnologia. COMAN+ is 1.70 m tall, its mass is about 70 kg and has 28 DoFs: 7 DoFs for each arm, 6 DoFs for each leg, with the two related to the ankle provided by a particular four-bar actuation mechanism [56,57], and 2 DoFs for the torso, allowing roll and yaw rotations. For the purpose of our validation, we replaced the anthropomorphic hands of COMAN+ with spherical end-effectors.

In the next two subsections, in order to provide a comprehensive validation, we first analyze the performance of the planning layer via numerical results and then showcase the effectiveness of the overall framework through experiments on the actual robot.

9.1. Planning results

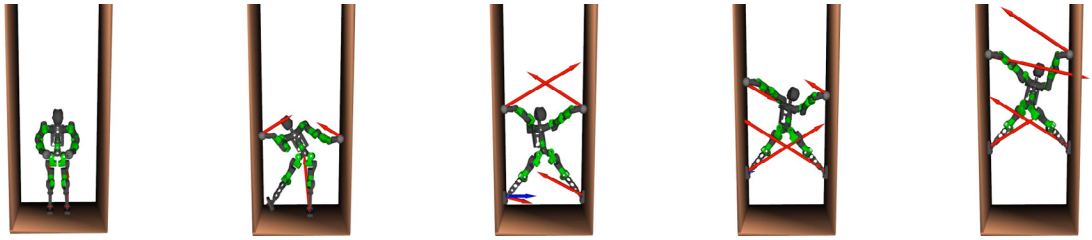
To illustrate the performance of the planning layer, we present numerical results obtained on an Intel Core i7-7500U CPU running at 2.70 GHz. We considered the four different multi-contact loco-manipulation tasks shown in Fig. 5, which for each of them provides some snapshots of a typical solution produced by the planning layer. We invite the reader to watch the accompanying video, which contains animated clips of such solutions, to better appreciate the effectiveness of the planned motions. The four tasks are described in the following.

1. *Ladder climbing*. The robot, starting from its homing configuration, must climb a ladder that is located in front of it. The ladder has 10 rungs and is inclined by 63°. The rungs are vertically equispaced by 0.22 m, and each of them is 1 m long and 0.08 m wide. The desired final stance requires placing both hands on the topmost rung and both feet six rungs below.
2. *Parallel walls climbing*. The robot, starting from its homing configuration, must climb vertically between two parallel walls located on its left and right flanks. The walls are 1.4 m apart. The desired final stance requires placing the left/right hand on the left/right wall at a height of 2.5 m above the ground, while the left/right foot must be placed 1.4 m below the left/right hand.
3. *Quadrupedal walking*. The robot, starting from its homing configuration, must navigate a narrow passage that is 0.3 m long and 1.1 m wide/high. Note that, due to its characteristics, the passage cannot be navigated by bipedal walking. The desired final stance requires both hands and feet to be on the ground at the exit of the narrow passage and is thus realizable only by a quadruped-like configuration.
4. *Standing up*. The robot, starting from a quadruped-like configuration, must stand upright, possibly exploiting a wall located in front of it as support. In particular, the robot's initial configuration may represent the final configuration reached after completing the previous task, e.g., performing a motion aimed at recovering the upright posture.

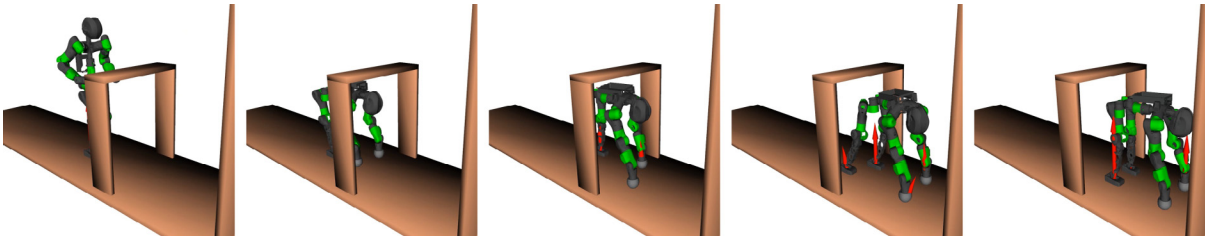
The point clouds representing the three scenarios in which such tasks take place (quadrupedal walking and standing up are shown in the same scenario) have a resolution of 0.025 m. For all the tasks, the set U of end-effectors contains both feet and hands, i.e., $U = \{\mathcal{F}^{\text{lf}}, \mathcal{F}^{\text{rf}}, \mathcal{F}^{\text{lh}}, \mathcal{F}^{\text{rh}}\}$, which can establish, respectively, surface and point contacts. The placement of potential contact frames is that shown in Fig. 2, except for the ladder climbing task where we have chosen a slightly different placement for the hands, shown in Fig. 6, which allows establishing contacts with the rungs in a more natural fashion.

For the second and third tasks, in order to speed up the stance planner, we provided it with an intermediate stance σ^{int} (illustrated, respectively, in snapshot 3 of Fig. 5-(b) and in snapshot 2 of Fig. 5-(c)) to bias the growth of the tree first toward such intermediate stance, and then toward the desired final stance. In particular, when the stance planner chooses to perform a tree

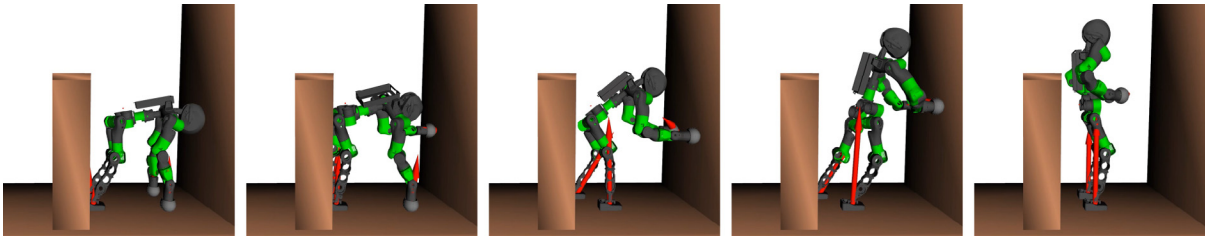
(a) Ladder climbing.



(b) Parallel walls climbing.



(c) Quadrupedal walking.



(d) Standing up.

Fig. 5. A typical solution of the planning layer for each considered task. The Red and blue arrows represent the force and moment sub-vectors of the planned contact wrenches respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

expansion attempt via exploitation (see Section 5.1), contact c^{fin} is randomly picked among those in σ^{int} (σ^{fin}) if a vertex containing σ^{int} does not exist (exists) in $\mathcal{T}$.

The values used for the parameters involved in the planning layer for the four tasks are reported above. For ease of reading, we report in Table 2 a synthetic recap of the most relevant parameters. For an exhaustive description of each of them, the reader is referred to the specific section.

- For the tree construction (Section 5.1), the stance planner uses $l_{\sigma}^{\max} = 5000$; $r^{\min}$ and $r^{\max}$ are set to 0.25 m and 1.5 m, both for hands and feet, for all tasks except the forth, where $r^{\min}$ is set to 0.8 m for the hands and 0.3 m for the feet.
- The transition generator (Section 5.2) works with $l_{\text{tran}}^{\max} = 100$ and $\Delta T_{\text{tran}} = 1$ s.
- In the IK solver (Section 5.2.2), $\boldsymbol{K}_{\sigma}$ and $\boldsymbol{K}_{q}$ are identities, $\alpha = 10^{-2}$, $\varepsilon_{\sigma} = 10^{-4}$ and $l_{\text{IK}}^{\max} = 1000$.
- For the computation of the contact wrenches (Section 5.3), the static friction coefficient is considered equal for all end-effectors and set to 0.8 for the first two tasks and to 0.5 for the last two tasks; half-dimensions of the CoP admissible region d^x and d^y for both feet are set to 0.1 m and 0.05 m for all tasks except the fourth, where they are set to 0.04 m; moreover $\beta = 10^{-4}$ and $\varepsilon_u = 0.05$.
- Finally, the whole-body planner uses $\Delta T_{\text{via}} = 1$ s for all tasks except the first, where we set $\Delta T_{\text{via}} = 2$ s.

Since our planning layer is randomized (as both the stance and whole-body planners rely on probabilistic strategies), we performed 100 runs for each of the four tasks. Thus, in each run, we first produced the sequences $\mathcal{S}_{\sigma}$, $\mathcal{S}_{q}$, $\mathcal{S}_{W}$ of stances, transitions, and contact wrenches using the stance planner; then, we passed $\mathcal{S}_{\sigma}$, $\mathcal{S}_{q}$ to the whole-body planner to compute the sequence $\mathcal{S}_{s}$ of trajectories.

Table 1
Averaged performance data of the stance planner.

Task	Planning time (s)	Transition generation time (s)	Number of iterations	Number of vertices in $\mathcal{T}$	Number of stances in $\mathcal{S}_\sigma$
Ladder climbing	43.60	37.64	1926.14	205.34	39.04
Parallel walls climbing	175.37	171.01	1557.37	151.28	44.12
Quadrupedal walking	62.69	55.99	2540.10	228.32	53.19
Standing up	7.56	6.02	459.24	62.19	17.00

Table 2
Recap of the most relevant parameters involved in the planning layer. For each parameter, we report the adopted symbol, the procedure in which it is used, and a synthetic description of its role.

Symbol	Procedure	Role
$l_\sigma^{\max}$	Stance planner	Maximum number of iterations
$r^{\min}$		Minimum radius of the spherical approximation of the end-effector workspace
$r^{\max}$		Maximum radius of the spherical approximation of the end-effector workspace
$l_{\text{tran}}^{\max}$	Transition generator	Number of iterations with fixed reference configuration
ΔT_{tran}		Time budget
α	IK solver	Regularization term weight
ε_σ		Threshold for the kinematic constraints error
$l_{\text{IK}}^{\max}$		Maximum number of integration steps
d^x	Whole-body planner	Half-length of the CoP admissible region
d^y		Half-width of the CoP admissible region
β		Regularization term weight
ε_u		Threshold for the centroidal statics term
ΔT_{via}		Time budget

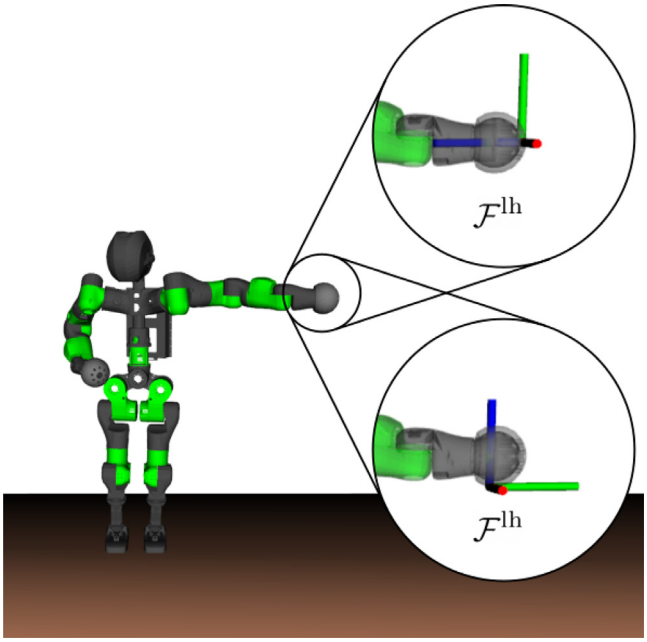


Fig. 6. The different placement of the potential contact frame on the left hand between the ladder climbing task (bottom) and the other tasks (top). The same applies to the right hand.

Table 1 collects the most significant performance data, averaged over the 100 runs, of the proposed stance planner. For each task, we report the time needed by the stance planner to find a solution, the time spent in generating transitions, the number of performed iterations, the number of vertices in the constructed tree $\mathcal{T}$, and the number of stances in the solution sequence $\mathcal{S}_\sigma$.

It is worth mentioning that the time needed by the whole-body planner to find a solution, provided in input a sequence $\mathcal{S}_q$ containing $N+1$ transitions, is the sum of the N times needed to plan the N trajectories between consecutive configurations in $\mathcal{S}_q$. Each of these times is constituted by those employed by the two stages which are, respectively, fixed at ΔT_{via} and not significant (as we have observed in our runs that the NLP described in Section 6.2 is always solved in few milliseconds). Then, the generic planning time of the whole-body planner is about $N \cdot \Delta T_{\text{via}}$. Eventually, the average time to complete the overall planning phase is the sum of the average planning times of the stance and whole-body planners.

We emphasize that our planning layer exhibited its capability of generating appropriate humanoid motions for four different multi-contact loco-manipulation tasks without requiring significant parameter modifications. This proves its versatility and confirms its applicability to different contexts.

9.2. Experiments

The experimental validation was carried out using the *XBot* framework [58] as control middleware for the COMAN+ humanoid. *XBot* works at a frequency of 1 kHz and guarantees deterministic hard real-time performance. All the robot joints are equipped with torque sensors which make available the measured joint torques $\hat{\boldsymbol{\tau}}$ at each time instant. These measures are used to estimate via (14) the contact wrenches at the spherical hands, which are sensorless, while the contact wrenches at the feet are directly provided by their force/torque sensors. Both hands and feet soles are covered by a thin layer of rubber.

In our experiment, the robot torque commands are computed via (21). It is worth noticing how the choice of the gain matrices $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ in (23) affects the behavior of the overall control layer. Increasing the gain values, the contribution of $\boldsymbol{\tau}_{\text{jnt}}$ in the computation of the torque commands (21) becomes the most significant; as a consequence, the robot will better track the reference joint trajectory while degrading its capability to track the reference contact wrenches $\overline{\boldsymbol{W}}_c$ that guarantee to maintain static balance. Obviously, an opposite behavior is obtained when reducing the gain values. As a trade-off, in our experiments, we adopted a simple variable stiffness strategy to adaptively choose the gain values in the diagonal matrices $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$; in particular, when moving from one stance to another, gain values are increased for those joints belonging to the kinematic chain ending with the end-effector by which the two stances differ. As regards the thresholds $w^{\min}$ and $w^{\max}$ used by the stance

Fig. 7. Experiment: COMAN+ sequentially performs the quadrupedal walking and standing-up tasks. The first snapshot reports the measures of the experimental area. See the accompanying video.

switching module to check the achievement of a certain stance, we set them to 0 N and 30 N, respectively, for each end-effector in U.

Our main experiment aims at showcasing the feasibility of the planned motions on the real robotic platform and assessing the reliability of the presented control layer. To this end, we reconstructed the third scenario (see Figs. 5-(c) and 5-(d)) described in the previous section; in particular, the narrow passage is constituted by wooden panels, and the wall is made of bricks. As mentioned before, in this scenario the robot must first traverse a corridor-like passage to reach a destination $\approx$ 1.65 m far from the initial location. A passage is placed at $\approx$ 0.75 m from such location and, due to its narrow dimensions, 1.1 m wide/high, it can be overcome by the robot only by passing below it. At $\approx$ 1.4 m beyond the passage, there is a wall that can be exploited as support for the final stand-up. The environment is reconstructed as a synthetic point cloud and used in our planning layer. Then, COMAN+ was requested to sequentially perform the quadrupedal walking and standing-up motions resulting from the planning stage.

Fig. 7 shows screenshots of the whole experiment in which COMAN+ successfully managed to navigate the narrow passage and then stand up exploiting contacts with the wall. Fig. 8 collects the plots representing the evolution of the reference contact forces computed by the reactive balancing module at each end-effector in U and their corresponding measured/estimated values. The vertical lines separate the three phases of the task execution. In the first phase, the robot starts in the homing configuration, in which all its weight is supported by the feet only, and moves to a quadrupedal-like configuration, in which its weight is equally distributed among the four contacts. In the second phase, the robot performs multiple steps repeatedly establishing and breaking contacts using different end-effectors. In the third phase, the robot reaches the wall and starts to push against it to recover the bipedal posture. In the final part of the plot, it is clear the work performed by the feet tangential forces in counterbalancing the pushes of the hands on the wall while standing up. No evident slippery or drifting phenomena were observed during motion execution and the experiment was successfully carried out multiple times despite the lack of a visual perception layer, confirming the strong reliability of the planning and control layers.

The accompanying video includes the full movie clip of the whole experiment, together with another experiment where the humanoid successfully completes the quadrupedal walking task even in presence of external disturbances, which shows the robustness of the control layer granted by the joint action of the stance switching and reactive balancing modules.

10. Discussion

In this section, we provide some additional comments about the proposed MCPC framework.

10.1. Considerations w.r.t. previous works

The two tasks that we successfully addressed in our experimental validation have been rarely considered in previous works.

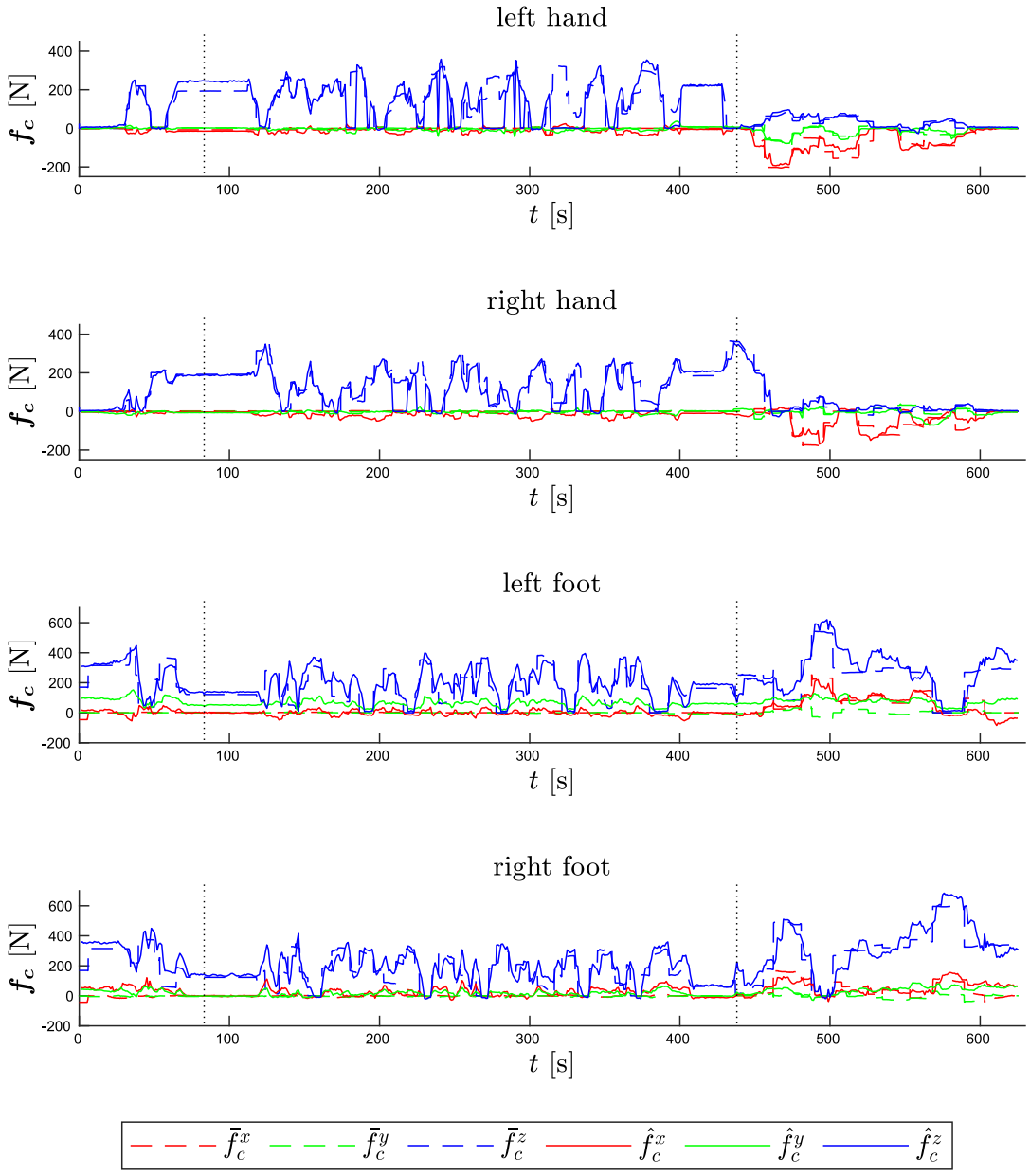


Fig. 8. Evolution of the contact forces at each end-effector during the whole experiment: reference (dashed lines) vs estimated/measured (solid lines) values.

Quadrupedal walking with a real humanoid is achieved in [59] using the E2-DR robot. Different from COMAN+, E2-DR possesses hardware designed to accomplish exactly this type of task. In particular, it is equipped with a wide-range, high-torque pitch joint in the torso which simplifies both the passage from bipedal to quadrupedal configuration and the consequent quadrupedal walk. We emphasize that we succeeded in planning such complex multi-modal loco-manipulation tasks in COMAN+ even though its hardware was not designed for this purpose. Furthermore, multiple experimental trials have confirmed the high reliability of our control layer in executing the planned motions.

Standing up exploiting a wall as support is considered in [7]. That work focuses exclusively on the planning aspects and thus the result is shown solely on a simulated humanoid. In contrast, using the proposed framework, we were able to tackle such challenging tasks both at the planning and control level, ultimately achieving experimental results on a real humanoid. We consider this result particularly relevant.

The stance planner proposed in [7] is, to the best of our knowledge, the most efficient among those existing in the literature. To find a sequence of stances for the standing-up task it needed about 2.5 s on average. Although our stance planner does not match the same performance, its average computational time is still small and is obtained without the need for any kind of pre-computation and heuristic specification. These two operations are instead required with the planner in [7] where, in order to decide possible contacts, feasible configurations for the robot kinematic chains are extracted from a precomputed octree data structure according to user-defined heuristics. Performing precomputations (which are needed also in different forms with other existing planners, e.g., [18]) may in principle require significant time, while designing heuristics may be tedious and task-specific. A similar observation might be done regarding the choice of predesigned possible contacts between robot and environment points [4]. Our stance planner has the advantage of avoiding the need for any of these operations.

10.2. Limitations and possible adaptations

The presented MCPC framework proved to be capable of generating sensible humanoid motions for different multi-contact loco-manipulation tasks and we believe that it can be used for practical applications. However, for the sake of completeness, we list below three main limitations of our framework that will be tackled in future work.

1. Our framework only considers static balance. An extension to cope with dynamic motions is not trivial, but we believe that our scheme still represents a valid template. In the direction of such extension, one possibility consists in keeping the stance planner identical, while the motions between consecutive transitions could be generated online using the common approach (see, e.g., [15]) of computing the humanoid CoM trajectory via Model Predictive Control (using a reduced model of the robot dynamics) and tracking it with a whole-body controller.
2. Our stance planner does not account for the quality of the generated solutions. In principle, this might produce sub-optimal behaviors; for example, the humanoid might move the same end-effector two consecutive times, even in case this is not strictly needed. Accounting for user-defined quality criteria, such as the minimization of the number of stances in the produced sequence, would be possible by applying an RRT*-like strategy.
3. Our planning layer is not equipped with a backtracking strategy to address the case in which the whole-body planner fails to find a connection between two consecutive configurations $\boldsymbol{q}_j$, $\boldsymbol{q}_{j+1}$ in the transition sequence $\mathcal{S}_q$ produced by the stance planner. A simple solution would be to prune the tree $\mathcal{T}$ of the subtree rooted at $\boldsymbol{q}_j$ (in the spirit of lazy planners, see for example [60]) and restart the stance planner from the resulting tree; the investigation of more effective strategies will be part of our future work.

11. Conclusions

In this paper, we have presented a complete multi-contact planning and control framework that allows a torque-controlled humanoid robot to decide and execute its motions to fulfill a generic multi-contact loco-manipulation task. Our framework is constituted by two layers. The planning layer works offline using two modules: the stance planner finds a sequence of stances, together with associated transitions and contact wrenches, and then the whole-body planner computes the sequence of motions to realize them. The control layer involves two modules, i.e., the stance switching and reactive balancing module, to produce the torque commands allowing the humanoid to execute the planned motions while guaranteeing closed-loop balance by absorbing possible execution inaccuracies, external disturbances, and modeling uncertainties. We validate the proposed framework via both numerical and experimental results obtained on the COMAN+ humanoid robot.

In addition to the adaptations that we mentioned in Section 10.2, the presented MCPC framework can be further developed along several lines. A first possibility that we want to consider is its extension to the case of hybrid wheeled–legged quadrupedal robots such as CENTAURO [61], in which also rolling contacts need to be accounted for. Furthermore, the integration of a perception layer would be beneficial, as the full autonomy of the robot inevitably depends on its sensing capabilities. Finally, it would be interesting to employ the proposed stance planner as a local strategy to decide how to establish an additional contact whenever this is required for the humanoid to increase the safety level [62], for example in the presence of an imminent risk of falling.

Declaration of competing interest

All authors declare that they have no conflicts of interest.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was granted by the European Union's Horizon 2020 Research and Innovation Programs under Grant No. 779963 (EU-ROBENCH) and Grant No. 101070596 (euROBIN).

Appendix A. Supplementary data

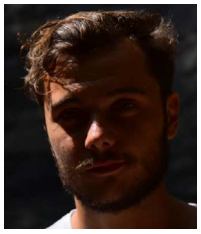
Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104448.

References

[1] K. Bouyarmane, S. Caron, A. Escande, A. Kheddar, Multi-contact motion planning and control, in: Humanoid Robotics: A Reference, Springer, 2019, pp. 1763–1804.
[2] C.G. Atkeson, P.B. Benzun, N. Banerjee, D. Berenson, C.P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, et al., What happened at the DARPA robotics challenge finals, in: The DARPA Robotics Challenge Finals: Humanoid Robots To the Rescue, Springer, 2018, pp. 667–684.
[3] T. Bretl, Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem, Int. J. Robot. Res. 25 (4) (2006) 317–342.
[4] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, B. Wilcox, Motion planning for legged robots on varied terrain, Int. J. Robot. Res. 27 (11–12) (2008) 1325–1349.
[5] K. Bouyarmane, A. Kheddar, Humanoid robot locomotion and manipulation step planning, Adv. Robot. 26 (10) (2012) 1099–1126.
[6] A. Escande, A. Kheddar, S. Miossec, Planning contact points for humanoid robots, Robot. Auton. Syst. 61 (5) (2013) 428–442.
[7] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, N. Mansard, An efficient acyclic contact planner for multiped robots, IEEE Trans. Robot. 34 (3) (2018) 586–601.
[8] A.W. Winkler, C.D. Bellicoso, M. Hutter, J. Buchli, Gait and trajectory optimization for legged systems through phase-based end-effector parameterization, IEEE Robot. Autom. Lett. 3 (3) (2018) 1560–1567.
[9] E.M. Hoffman, M. Parigi Polverini, A. Laurenzi, N.G. Tsagarakis, Modeling and optimal control for rope-assisted rappelling maneuvers, in: IEEE International Conference on Robotics and Automation, ICRA, 2021, pp. 9826–9832.
[10] Z. Manchester, N. Doshi, R.J. Wood, S. Kuindersma, Contact-implicit trajectory optimization using variational integrators, Int. J. Robot. Res. 38 (12–13) (2019) 1463–1476.
[11] A. Patel, S.L. Shield, S. Kazi, A.M. Johnson, L.T. Biegler, Contact-implicit trajectory optimization using orthogonal collocation, IEEE Robot. Autom. Lett. 4 (2) (2019) 2242–2249.
[12] J. Carius, R. Ranftl, V. Koltun, M. Hutter, Trajectory optimization with implicit hard contacts, IEEE Robot. Autom. Lett. 3 (4) (2018) 3316–3323, http://dx.doi.org/10.1109/LRA.2018.2852785.
[13] J.-P. Sleiman, J. Carius, R. Grandia, M. Wermelinger, M. Hutter, Contact-implicit trajectory optimization for dynamic object manipulation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2019, pp. 6814–6821.
[14] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, N. Mansard, A versatile and efficient pattern generator for generalized legged locomotion, in: 2016 IEEE International Conference on Robotics and Automation, 2016, pp. 3555–3561.
[15] S. Caron, A. Kheddar, Multi-contact walking pattern generation based on model preview control of 3D COM accelerations, in: 2016 IEEE-RAS International Conference on Humanoid Robots, 2016, pp. 550–557.
[16] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, N. Mansard, Crocoddyl: An efficient and versatile framework for multi-contact optimal control, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 2536–2542.
[17] F. Ruscelli, A. Laurenzi, N.G. Tsagarakis, E. Mingo Hoffman, Horizon: A trajectory optimization framework for robotic systems, Front. Robot. AI 9 (2022).

[18] S.-Y. Chung, O. Khatib, Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots, in: 2015 IEEE International Conference on Robotics and Automation, 2015, pp. 6289–6294.
[19] K. Bouyarmane, A. Kheddar, Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 4414–4419.
[20] F. Ruscelli, M.P. Polverini, A. Laurenzi, E. Mingo Hoffman, N.G. Tsagarakis, A multi-contact motion planning and control strategy for physical interaction tasks using a humanoid robot, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020, pp. 3869–3876.
[21] Z. Kingston, M. Moll, L.E. Kavraki, Sampling-based methods for motion planning with constraints, Ann. Rev. Control Robot. Auton. Syst. 1 (2018) 159–185.
[22] Y. Yang, V. Ivan, W. Merkt, S. Vijayakumar, Scaling sampling-based motion planning to humanoid robots, in: 2016 IEEE International Conference on Robotics and Biomimetics, ROBIO, IEEE, 2016, pp. 1448–1454.
[23] F. Burget, A. Hornung, M. Bennewitz, Whole-body motion planning for manipulation of articulated objects, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 1656–1662.
[24] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, E. Yoshida, Fast humanoid robot collision-free footstep planning using swept volume approximations, IEEE Trans. Robot. 28 (2) (2011) 427–439.
[25] P. Ferrari, M. Cognetti, G. Oriolo, Humanoid whole-body planning for loco-manipulation tasks, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 4741–4746.
[26] J.H. Park, Compliance/impedance control strategy for humanoids, in: Humanoid Robotics: A Reference, Springer, 2019, pp. 1009–1028.
[27] L. Saab, O.E. Ramos, F. Keith, N. Mansard, P. Soueres, J.-Y. Fourquet, Dynamic whole-body motion generation under rigid contacts and other unilateral constraints, IEEE Trans. Robot. 29 (2) (2013) 346–362.
[28] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, L. Righetti, Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid, Auton. Robots 40 (3) (2016) 473–491.
[29] B.J. Stephens, C.G. Atkeson, Dynamic balance force control for compliant humanoid robots, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 1248–1255.
[30] S.-H. Lee, A. Goswami, A momentum-based balance controller for humanoid robots on non-level and non-stationary ground, Auton. Robots 33 (4) (2012) 399–414.
[31] B. Henze, M.A. Roa, C. Ott, Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios, Int. J. Robot. Res. 35 (12) (2016) 1522–1543.
[32] A. Laurenzi, E. Mingo Hoffman, M.P. Polverini, N.G. Tsagarakis, Balancing control through post-optimization of contact forces, in: 2018 IEEE-RAS International Conference on Humanoid Robots, 2018, pp. 320–326.
[33] M.P. Polverini, E. Mingo Hoffman, A. Laurenzi, N.G. Tsagarakis, Sparse optimization of contact forces for balancing control of multi-legged humanoids, IEEE Robot. Autom. Lett. 4 (2) (2019) 1117–1124.
[34] A. Werner, B. Henze, D.A. Rodriguez, J. Gabaret, O. Porges, M.A. Roa, Multi-contact planning and control for a torque-controlled humanoid robot, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016, pp. 5708–5715.
[35] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, A. Escande, K. Bouyarmane, K. Kaneko, M. Morisawa, P. Gergondet, et al., Multi-contact vertical ladder climbing with an HRP-2 humanoid, Auton. Robots 40 (3) (2016) 561–580.
[36] M.P. Polverini, A. Laurenzi, E. Mingo Hoffman, F. Ruscelli, N.G. Tsagarakis, Multi-contact heavy object pushing with a centaur-type humanoid robot: Planning and control for a real demonstrator, IEEE Robot. Autom. Lett. 5 (2) (2020) 859–866.
[37] J.C. Trinkle, J.-S. Pang, S. Sudarsky, G. Lo, On dynamic multi-rigid-body contact problems with Coulomb friction, J. Appl. Math. Mech. 77 (4) (1997) 267–279.
[38] S. Caron, Q.-C. Pham, Y. Nakamura, Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas, in: 2015 IEEE International Conference on Robotics and Automation, 2015, pp. 5107–5112.
[39] L. Jamone, L. Natale, G. Sandini, A. Takanishi, Interactive online learning of the kinematic workspace of a humanoid robot, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 2606–2612.
[40] Y. Guan, K. Yokoi, Reachable space generation of a humanoid robot using the Monte Carlo method, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 1984–1989.
[41] L. Rossini, E. Mingo Hoffman, A. Laurenzi, N. Tsagarakis, NSPG: An efficient posture generator based on null-space alteration and kinetostatics constraints, Front. Robot. AI 8 (2021) 715325.
[42] L. Jaillet, J.M. Porta, Asymptotically-optimal path planning on manifolds, in: Robotics: Science and Systems, 2012, pp. 1–8.
[43] E. Dantec, R. Budhiraja, A. Roig, T. Lembono, G. Saurel, O. Stasse, P. Fernbach, S. Tonneau, S. Vijayakumar, S. Calinon, M. Taix, N. Mansard, Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos, in: IEEE International Conference on Robotics and Automation, ICRA, 2021, pp. 8202–8208.
[44] A. Del Prete, F. Nori, G. Metta, L. Natale, Prioritized motion–force control of constrained fully-actuated robots: "Task Space Inverse Dynamics", Robot. Auton. Syst. 63 (2015) 150–157.
[45] A. Laurenzi, E. Mingo Hoffman, L. Muratore, N.G. Tsagarakis, CartesI/O: A ROS based real-time capable Cartesian control framework, in: 2019 IEEE International Conference on Robotics and Automation, 2019, pp. 591–596.
[46] E. Mingo Hoffman, A. Rocchi, A. Laurenzi, N.G. Tsagarakis, Robot control for dummies: Insights and examples using OpenSoT, in: 2017 IEEE-RAS International Conference on Humanoid Robotics, 2017, pp. 736–741.
[47] E. Mingo Hoffman, N.G. Tsagarakis, The Math of Tasks: a Domain Specific Language for constraint-based task specification, Int. J. Hum. Robotics 18 (3) (2021) 2150008.
[48] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, M. Diehl, qpOASES: A parametric active-set algorithm for quadratic programming, Math. Program. Comput. 6 (4) (2014) 327–363.
[49] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, OSQP: An operator splitting solver for quadratic programs, Math. Program. Comput. 12 (4) (2020) 637–672.
[50] I.A. Sucan, M. Moll, L.E. Kavraki, The open motion planning library, IEEE Robot. Autom. Mag. 19 (4) (2012) 72–82.
[51] J.A.E. Andersson, J. Gillis, G. Horn, J.B. Rawlings, M. Diehl, CasADi: A software framework for nonlinear optimization and optimal control, Math. Program. Comput. 11 (1) (2019) 1–36.
[52] A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math. Program. 106 (1) (2006) 25–57.
[53] M. Görner, R. Haschke, H. Ritter, J. Zhang, MoveIt! Task constructor for task-level motion planning, in: 2019 IEEE International Conference on Robotics and Automation, 2019, pp. 190–196.
[54] J. Pan, S. Chitta, D. Manocha, FCL: A general purpose library for collision and proximity queries, in: 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 3859–3866.
[55] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Auton. Robots 34 (3) (2013) 189–206.
[56] F. Ruscelli, A. Laurenzi, E. Mingo Hoffman, N.G. Tsagarakis, A fail-safe semi-centralized impedance controller: Validation on a parallel kinematics ankle, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018, pp. 1–9.
[57] C. Zhou, N. Tsagarakis, On the comprehensive kinematics analysis of a humanoid parallel ankle mechanism, J. Mech. Robot. 10 (5) (2018).
[58] L. Muratore, A. Laurenzi, E. Mingo Hoffman, N.G. Tsagarakis, The XBot real-time software framework for robotics: From the developer to the user perspective, IEEE Robot. Autom. Mag. 27 (3) (2020) 133–143.
[59] T. Yoshiike, M. Kuroda, R. Ujino, Y. Kanemoto, H. Kaneko, H. Higuchi, S. Komura, S. Iwasaki, M. Asatani, T. Koshiishi, The experimental humanoid robot E2-DR: A design for inspection and disaster response in industrial environments, IEEE Robot. Autom. Mag. 26 (4) (2019) 46–58.
[60] P. Ferrari, M. Cognetti, G. Oriolo, Sensor-based whole-body planning/replanning for humanoid robots, in: 2019 IEEE-RAS International Conference on Humanoid Robots, 2019, pp. 511–517.
[61] N. Kashiri, L. Baccelliere, L. Muratore, A. Laurenzi, Z. Ren, E. Mingo Hoffman, M. Kamedula, G.F. Rigano, J. Malzahn, S. Cordasco, et al., CENTAURO: A hybrid locomotion and high power resilient manipulation platform, IEEE Robot. Autom. Lett. 4 (2) (2019) 1595–1602.
[62] N. Scianca, P. Ferrari, D. De Simone, L. Lanari, G. Oriolo, A behavior-based framework for safe deployment of humanoid robots, Auton. Robots (2021) 1–22.

Paolo Ferrari He received the master's degree in artificial intelligence and robotics and the Ph.D. degree in control engineering from Sapienza University of Rome, Rome, Italy, respectively in 2016 and 2021. His research focuses on motion planning and control for articulated robots, with particular emphasis on humanoids.



Luca Rossini He received the bachelor and master degree (cum Laude) in Mechanical Engineering from Politecnico di Milano in 2017 and 2019 respectively. He joined the Humanoid and Human Centered Mechatronics Laboratory (HHCM) at the Italian Institute of Technology (IIT – CRIS) on November 2019 as a Ph.D student working on offline and online planning and control strategies for the autonomous locomotion of humanoid and legged robots.

Francesco Ruscelli received the Master's degree in Automation Engineering from the University of Bologna in 2016 and the Ph.D. degree in Bioengineering and Robotics from the University of Genova in 2020. His research involves planning and control of legged systems, focusing on locomotion and interaction of humanoid and quadruped robots.

Arturo Laurenzi He received the Master's degree in automation engineering (cum laude) from Università Degli Studi di Firenze, Florence, Italy, in 2015. He is currently working toward the Ph.D. degree with the Humanoid and Human-Centered Mechatronics Laboratory, Istituto Italiano di Tecnologia, Genoa, Italy, on the software architecture and control of the hybrid wheeled–legged quadrupedal robot Centauro. Since 2019, he has been a Senior Technician with the Humanoid and Human-Centered Mechatronics Laboratory.

Giuseppe Oriolo He received his Ph.D. degree in Control Engineering in 1992 from Sapienza University of Rome, Italy. He is currently with the Department of Computer, Control and Management Engineering (DIAG) of the same university, where he is a Full Professor of automatic control and robotics and the director of the DIAG Robotics Lab. His research interests are in the general area of planning and control of robotic systems. Prof. Oriolo has been Associate Editor of the IEEE Transactions on Robotics and Automation from 2001 to 2005 and Editor of the IEEE Transactions on Robotics from 2009 to 2013. He is a Fellow of the IEEE.

Nikolaos Tsagarakis He received the M.Sc. Degree in control engineering and the Ph.D. degree in robotics from the University of Salford, Manchester, U.K., in 1997 and 2000, respectively. He is currently a Tenured Senior Scientist and Principal Investigator of the Humanoids and Human-Centered Mechatronics Research Line at Istituto Italiano di Tecnologia, Genoa, Italy.

Enrico Mingo Hoffman He received the B.Sc. degree in electronics engineering and the M.Sc. degree in AI and robotics, both from the University of Rome "La Sapienza," Rome, Italy, in 2008 and 2012, respectively, and the Ph.D. degree in humanoid robotics from the University of Genoa, Genoa, Italy, in 2016. Since 2020, he has been a Researcher with the Humanoid and Human-Centered Mechatronics Laboratory, Istituto Italiano di Tecnologia, Genoa. His research interests include robot motion planning and control. He is currently working as Senior Researcher in humanoid robotics at PAL Robotics, Barcelona, Spain.