UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Offline and Online Planning and Control Strategies for the Multi-Contact and Biped Locomotion of Humanoid Robots

by

Luca Rossini

September 2022

Nikolaos G. Tsagarakis — Supervisor
Enrico Mingo Hoffmann — Co-Supervisor
Paolo Massobrio — Head of the PhD program

Istituto Italiano di Tecnologia (*HHCM* lab) and
Department of Informatics, Bioengineering, Robotics and Systems Engineering

Abstract

In the past decades, the Research on humanoid robots made progress forward accomplishing exceptionally dynamic and agile motions. Starting from the DARPA Robotic Challenge in 2015, humanoid platforms have been successfully employed to perform more and more challenging tasks with the eventual aim of assisting or replacing humans in hazardous and stressful working situations. However, the deployment of these complex machines in realistic domestic and working environments still represents a high-level challenge for robotics. Such environments are characterized by unstructured and cluttered settings with continuously varying conditions due to the dynamic presence of humans and other mobile entities, which cannot only compromise the operation of the robotic system but can also pose severe risks both to the people and the robot itself due to unexpected interactions and impacts. The ability to react to these unexpected interactions is therefore a paramount requirement for enabling the robot to adapt its behavior to the task needs and the characteristics of the environment. Further, the capability to move in a complex and varying environment is an essential skill for a humanoid robot for the execution of any task. Indeed, human instructions may often require the robot to move and reach a desired location, e.g., for bringing an object or for inspecting a specific place of an infrastructure. In this context, a flexible and autonomous walking behavior is an essential skill, study of which represents one of the main topics of this Thesis, considering disturbances and unfeasibilities coming both from the environment and dynamic obstacles that populate realistic scenarios. Locomotion planning strategies are still an open theme in the humanoids and legged robots research and can be classified in sample-based and optimization-based planning algorithms. The first, explore the configuration space, finding a feasible path between the start and goal robot's configuration with different logic depending on the algorithm. They suffer of a high computational cost that often makes difficult, if not impossible, their online implementations but, compared to their counterparts, they do not need any environment or robot simplification to find a solution and they are probabilistic complete, meaning that a feasible solution can be certainly found if at least one exists. The goal of this thesis is to merge the two algorithms in a coupled offline-online planning framework to generate an offline global trajectory with a sample-based approach

to cope with any kind of cluttered and complex environment, and online locally refine it during the execution, using a faster optimization-based algorithm that more suits an online implementation. The offline planner performances are improved by planning in the robot contact space instead of the whole-body robot configuration space, requiring an algorithm that maps the two state spaces. The framework proposes a methodology to generate whole-body trajectories for the motion of humanoid and legged robots in realistic and dynamically changing environments. This thesis focuses on the design and test of each component of this planning framework, whose validation is carried out on the real robotic platforms CENTAURO and COMAN+ in various loco-manipulation tasks scenarios.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

CD	Centroidal Dynamics
CoM	Center of Mass
DCM	Divergent Component of Motion
DMS	Direct Multiple Shooting
LIPM	Linear Inverted Pendulum
MCP	Multi-Contact Planner
MCPC	Multi-Contact Planning and Control
MPC	Model Predictive Control
NLLSO	Non-Linear Least Square Optimization
NLP	Non-Linear Program
NSPG	Null-Space Posture Generator
OCP	Optimal Control Problem
SRBD	Single Rigid Body Dynamics
TO	Trajectory Optimization
WBC	Whole-Body Controller

Chapter 1

Introduction

1.1 Motivations

The aim of robotics has always been the design of automatic systems able to help and simplify tedious and dangerous tasks for humans. The first robots to be introduced, and that still make up the majority of the robotic systems in the industry, are fixed-based manipulators. Indeed, thanks to their flexibility, they can be employed in a wide set of cyclic and heavy tasks relaxing the load on the human operator. However, fixed-base manipulators work in a restricted and fixed workspace and they are not designed to be moved around.

Modern companies are moving towards a more flexible automation, with autonomous warehouses that can move goods, limiting human responsibilities to supervision only. To this end, mobile-base robots, generally made by mounting a manipulator on a wheeled base, offer a theoretically unlimited workspace thanks to their mobility, and it is not surprising that one of the most important e-commerce companies in the world has one of the most advanced research teams in the field. Indeed, Amazon Robotics in Fig. 1.1, has extensively utilized mobile robots in structured industrial and warehouse environments with flat terrains and specifically dedicated paths. On the contrary, legged robots can move in more complex environments by stepping and interacting with the environment.

A special class of legged robots are the humanoid robots, multi-articulated robots that associate a kinematic chain to a limb, with a structure similar to a human. Problems in controlling these complex machines come from their bipedal nature. Humanoid robots have a small support polygon compared to their size, and the CoM is usually higher than any other robot, making them prone to fall or fail during the execution of a task. The Darpa Robotic Challenge (DRC) held in 2015 (Fig. 1.2), pushed the interest and research on these machines, which are now able to execute complex and spectacular motions, outwitting humans in

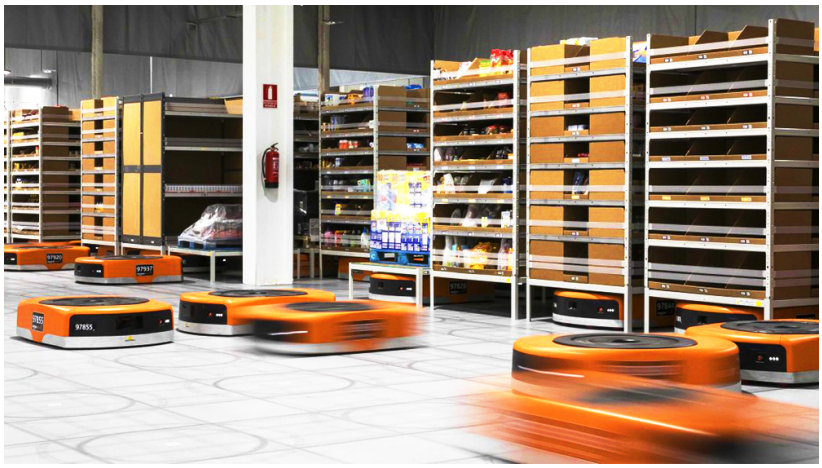


Figure 1.1 The mobile base robot Kiva designed and built by Amazon Robotics. This image was retrieved from https://aws.amazon.com/it

certain cases [Atkeson et al., 2018]. Despite the incredible progress of the last decade, the deployment of these complex legged robots in a domestic or working environment is still hard to accomplish. Indeed, these kinds of environments are characterized by cluttered and continuously varying conditions due to the dynamic presence of humans and other mobile obstacles, which can not only compromise the operation of the robotic system but can also pose severe risks both for the people and the robot itself, due to unexpected interactions and impacts.

Recently, companies like Apptronik[1] with Astro, Agility Robotics[2] with Digit and Halodi[3] with Eve, are focusing on building humanoids able to be employed in this kind of environment to work side by side with humans in structured and controlled working environments. Not for nothing, their mottos *"Robots for Humans"*, and *"Made for Work"* reflect their intentions.

When thinking to humanoids, the lower body has always been assigned to navigation tasks while the upper-body manipulates objects interacting with the environment. However, humanoids have the unique capability to take advantage also of the upper body to move in particularly narrow and challenging passages, for instance, crawling below short obstacles or climbing high ones switching between the bipedal and multi-contact locomotion. Since, historically, locomotion and manipulation tasks were accomplished using the lower and upper body respectively, this peculiar kind of multi-contact locomotion belongs to the family of *loco-manipulation tasks*, shown in Fig. 1.3, and it is still a challenging problem both

[1]https://apptronik.com/
[2]https://agilityrobotics.com/
[3]https://www.halodi.com/

(a) Atlas (MIT) (b) WALK-MAN (IIT) (c) RoboSimian (JPL)

Figure 1.2 Humanoid robots participating in the Darpa Robotics Challenge (DRC) in 2015

from a planning and a control point of view. From the planning perspective, most of the difficulties derive from solving the dual problem of (i) finding a contact sequence e.g., in which location and when a contact must be established, and (ii) generating a dynamically feasible motion compliant with the contact sequence. Indeed, it is easy that this dual problem can explode when considering at the same time the contact sequence and all the possible feasible whole-body paths. At the same time, the execution of the planned trajectory is a non-trivial issue that must guarantee robustness from external disturbances and early/late contact establishment w.r.t. the planned contact timing.

This thesis tackles this problem proposing a solution for the generation of acyclic multi-contact trajectories for loco-manipulation tasks coupled with a robust control layer able to adjust the planned motions to react to changes in the robot and environment state during its execution.

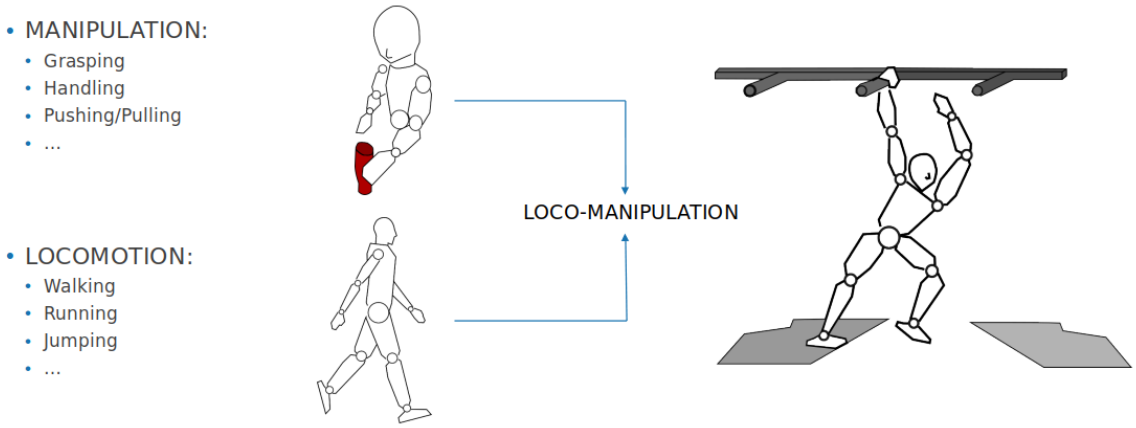


Figure 1.3 A loco-manipulation task represented as the joining of a manipulation task and a locomotion task.

1.2 Previous Work

After years of research, humanoid robots can safely walk in an environment with flat and cluttered terrains with a certain autonomy, but still, their locomotion is an open research theme in the scientific community. Humanoid and legged robots offer severe challenges from their modeling and control point of view like under-actuation and varying constraints deriving from the sequence of interactions with the environment. For this reason, some simplification of the complicated multi-body dynamics to incorporate the essential information can be useful. Examples of these simplifications are the Linear Inverted Pendulum (LIP) [Englsberger et al., 2015] and Centroidal Dynamics (CD) [Winkler, 5 14].

Humanoids, as humans, can access arguably more environments compared to wheeled and other mobile robots by jumping, running, swimming, and climbing. Humanoid robots can potentially accomplish complex locomotion tasks by breaking and establishing multiple non-coplanar contacts with their hand and possibly any other parts of their body. Compared to the more classic walking planners where the gait generation is implicitly defined by the alternation of left and right steps, using multiple contacts improves the motion capability of the robot allowing more complex gait sequences. Most of the complexity derives from contact planning, i.e., the decision process that picks the contact sequence and their location. Compared with the walking case where the choice of the new end-effector in contact with the ground is trivial, an acyclic multi-contact problem has every time infinite possibilities to establish a new contact. Additionally, each contact defines dynamic and kinematic constraints, and we say that the contact manifold is foliated [Siméon et al., 2004] and the robot motion moves through the intersection of these manifolds. Due to this complexity, many previous works solved the contact generation problem separately from the dynamically feasible motion generation compliant with the planned contact sequence. This procedure, called *contact-before-motion planning*, is based on a sample-based planner that expands a search to connect a user-defined start and goal configuration [Escande et al., 2013]. The main issue of this approach is the computational cost, but it guarantees theoretical probabilistic completeness, meaning that a solution is certainly found if at least a feasible one exists [Karaman and Frazzoli, 2011]. Sample-based planners explore the robot's configuration space by expanding a search tree sampling configurations probabilistic or randomly [LaValle, 2006]. However, the sampled configurations must belong to the foliated topology to be acceptable, resulting in an inefficient random process where many samples are rejected. A solution consists in explicitly projecting the sampled configuration in contact with the environment and consequently belongs to the contact manifold, but this is a very costly and inefficient operation since

many “free-flying” configurations share the same projection. More advanced techniques reduced the inefficiency of the process sampling states directly on manifold tangent atlases or spaces [Kingston et al., 2018]. Recently, Tonneau et al. [2018] presented an efficient acyclic contact planner for multiped robots breaking the complexity of the problem and introducing the concept of the *reachability condition*.

Other previous work explored alternatives for solving the dual problem of finding the sequence of contact position and a trajectory for the base link using local optimizations. To keep the computational cost low, these methods use simplifications of the robot dynamics and simplify the environment using simpler convex shapes to include them in the optimization problem. Compared to their counterparts, the optimization-based planners rely on more and more performing linear and non-linear solvers but they are affected by local minima that may hinder the optimality of the solution. Additionally, the solver may fail to find a solution due to the non-convexity of the planning problem, which especially derives from the collision avoidance constraint. Thus, the algorithm is non-probabilistic complete. [Deits and Tedrake, 2014] took advantage of the binary state of the contact to solve the combinatory problem in one stage, but it is limited to bipedal walking and does not consider any interaction with the upper limbs. Some other approaches solve an optimal control problem to find a continuous global trajectory to move the robot in an arbitrarily complex environment, namely *trajectory optimization* (TO). This last approach has been used both assuming the contact sequence known as in [Ruscelli et al., 2022] and adapting contemporary the gait sequence and contact position as in [Winkler et al., 2018]. Even if the optimization-based motion planner were able to find a solution in a considerably lower amount of time compared to the counterpart sample-based approach (order of seconds vs order of minutes), this approach is still far from being an online implementation, particularly useful for realistic environments in presence of dynamic environmental conditions.

Recently Model Predictive Controllers (MPCs) gained attention for their capability to online generate walking trajectories to react to external disturbances closing the loop on the robot’s state. Romualdi et al. [2022], Ding et al. [2022], and Scianca et al. [2020] showed a consistent methodology based on a simplified dynamic template model for planning contact and CoM trajectories and a whole-body tracking controller. Galliker et al. [2022] presented preliminary work on a non-linear whole-body MPC-based planning strategy. Despite the promising results obtained, MPCs consider a fixed gait sequence and are not suitable for temporally global plans because they plan on a short receding time horizon.

1.3 Contributions

Previous works highlighted the complementary between sample-based and optimization-based planners regarding computational cost and probabilistic completeness. Sample-based planners have a high theoretical complexity leading to a high computational cost and are difficult to implement on floating-base robots with a high number of DoF that move interacting with the surrounding environment establishing and breaking contacts. On the other hand, optimization-based planners manage the contact alternation by adding and removing Cartesian and force constraints, speeding up the planning process, but requiring a user-defined gait sequence to be real-time implementable. Also, these planners are prone to local minima and failures due to the non-convexity of the problem and their performances are strictly related to the provided initial guess.

This thesis aims to take advantage of this complementary by designing and testing a complete framework that combines the sample and optimization-based planning techniques to generate and adapt acyclic multi-contact trajectories for humanoid and multi-limbed robots. Recently, a planning algorithm that merges sampling and optimization-based approaches has been proposed in Jelavic et al. [2021] generating a contact sequence and a whole-body feasible trajectory for a hybrid wheeled-legged excavator with information computed both offline and online.

The sample-based planner acts offline, finding a complete and global contact and joint space trajectory for the execution of an acyclic loco-manipulation task in a general complex and cluttered scenario. To tackle the computational problem deriving from the high number of DoF (i.e., *curse of dimensionality*) the configuration space is downsized considering the pose of the contacts only. Of course, this cost reduction comes with a drawback: at each iteration, the planner requires a map that moves back to the robot configuration space to find a whole-body robot's configuration compliant with a specific set of contacts. Apart from the clear necessity of moving back to the joint space to generate references that can be executed by the robot, collision, self-collision, and stability are constraints that the planner must consider when generating a posture projected onto the planned contacts, with the risk of invalidating the sampled contact state if a feasible configuration can not be found. Thus, the problem of posture generation is a key aspect of the offline sample-based planner with a strong influence on the efficiency of the planner, and it is widely discussed throughout the thesis.

During the execution of the global offline planned trajectory, an optimization-based planner locally refines the nominal solution depending on the dynamic conditions of the

environment and the robot itself. The sample-based multi-contact planner generates a solution both in the joint and contact space. Locally planning on the robot's joint space means solving an optimization problem in a space whose dimension is proportional to the number of DoF of the robot, increasing the computational cost and making harder to satisfy the requirements for the online applicability. But the joint state space vector defines the position, and eventually velocity, of each robot's joint, simplifying the definition of stability and collision avoidance. On the other hand, planning in contact space reduces the state vector dimension to the only pose of each contact, widely simplifying the problem but also losing information about the robot's state. Indeed, differently from the previous method, the collision avoidance constraint uses heuristics, such as relative distance constraints to avoid feet crossing or long and unfeasible steps. Additionally, re-planning the contact sequence requires an additional step that computes a new whole-body trajectory compliant with the new contacts. In this thesis, the optimization of both the state spaces has been carried out, highlighting the benefits and drawbacks of each methodology.

This thesis contributes by:

- Solving the posture generation problem of Multi-Contact Planners (MCP) by proposing a novel and efficient posture generation algorithm named Null-Space Posture Generation (NSPG) presented in Rossini et al. [2021].
- Proposing and validating a sample-based acyclic multi-contact planner for humanoids robots presented in Chapter 3.
- Coupling the offline sample-based planner with an online optimization-based local planner presented in Rossini and Tsagarakis [2022] that adjust the nominal trajectory computed by the MCP both in the joint and contact space.
- Proposing a Model Predictive Control (MPC) in Chapter 5 that generates a whole-body trajectory on the adjusted contact sequence, and improves the robustness of the motion execution to external disturbances.

The introduced framework proposes an algorithm that can plan an acyclic multi-contact trajectory and adapt it online to avoid moving obstacles and react to external disturbances. This work goes beyond the state of the art considering the real-time adaptation of complex and acyclic multi-contact trajectories instead of simpler cyclic walking ones, to execute loco-manipulation tasks with humanoid robots.

1.4 Structure of the Thesis

The thesis is divided into two parts, each one divided into two chapters, with each chapter describing a layer of the proposed framework. Part I introduces the offline layer addressing the problem of the posture generation in contact space sample-based planners proposing, in Chapter 2, the Null Space Posture Generator (NSPG) as an effective way of mapping whole-body robot's configurations with the planned set of contacts. The algorithm has been tested on the hybrid wheeled-legged robot CENTAURO. Chapter 3 describes the Multi-Contact Planner and Controller (MCPC) that generates the discrete set of contacts and whole-body motions composing the solution of the offline planning layer. The MCPC has been tested both in several simulation use cases and validated with an experimental campaign using the biped humanoid robot COMAN+.

Part II addresses the online local re-planning problem solved with the optimization-based approach both in the joint and contact space. Chapter 4 focuses on the joint space local re-planning, validating the proposed methodology with simulations and experiments carried out on the CENTAURO robot. Chapter 5, instead, solves the same problem in the contact space, presenting an MPC-based algorithm to generate whole-body trajectories compliant with the adjusted contact sequence. Simulations were carried out using the biped humanoid robot DRACO 3, during my 5-month long internship spent in the Humanoid Centered Robotics Lab (HCRL) at the University of Texas at Austin, hosted by Professor Luis Sentis.

The CENTAURO robot, as well as the COMAN+ robot, has been designed and built at the Humanoid and Human-Centered Mechatronics Laboratory (HHCM) at the Italian Institute of Technology (IIT). DRACO 3 has been designed and built at the HCRL together with Apptronik.

References

Atkeson, C. G., Benzun, P. W. B., Banerjee, N., Berenson, D., Bove, C. P., Cui, X., DeDonato, M., Du, R., Feng, S., Franklin, P., Gennert, M., Graff, J. P., He, P., Jaeger, A., Kim, J., Knoedler, K., Li, L., Liu, C., Long, X., Padir, T., Polido, F., Tighe, G. G., and Xinjilefu, X. (2018). *What Happened at the DARPA Robotics Challenge Finals"*, pages 667–684. Springer International Publishing", Cham.

Deits, R. and Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 279–286.

Ding, Y., Khazoom, C., Chignoli, M., and Kim, S. (2022). Orientation-aware model predictive control with footstep adaptation for dynamic humanoid walking.

Englsberger, J., Ott, C., and Albu-Schäffer, A. (2015). Three-dimensional bipedal walking control based on divergent component of motion. *IEEE Transactions on Robotics*, 31(2):355–368.

Escande, A., Kheddar, A., and Miossec, S. (2013). Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5):428–442.

Galliker, M. Y., Csomay-Shanklin, N., Grandia, R., Taylor, A. J., Farshidian, F., Hutter, M., and Ames, A. D. (2022). Planar bipedal locomotion with nonlinear model predictive control: Online gait generation using whole-body dynamics.

Jelavic, E., Farshidian, F., and Hutter, M. (2021). Combined sampling and optimization based planning for legged-wheeled robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8366–8372.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894.

Kingston, Z., Moll, M., and Kavraki, L. E. (2018). Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):159–185.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

Romualdi, G., Dafarra, S., L'Erario, G., Sorrentino, I., Traversaro, S., and Pucci, D. (2022). Online non-linear centroidal mpc for humanoid robot locomotion with step adjustment. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10412–10419.

Rossini, L., Mingo Hoffman, E., Laurenzi, A., and Tsagarakis, N. (2021). NSPG: An efficient posture generator based on null-space alteration and kinetostatics constraints. *Frontiers in Robotics and AI*, 8:715325.

Rossini, L. and Tsagarakis, N. (2022). From offline to online: A perception-based local planner for dynamic obstacle avoidance. In *2022 IEEE-RAS International Conference on Humanoid Robots*.

Ruscelli, F., Laurenzi, A., Tsagarakis, N. G., and Mingo Hoffman, E. (2022). Horizon: A trajectory optimization framework for robotic systems. *Frontiers in Robotics and AI*, 9.

Scianca, N., De Simone, D., Lanari, L., and Oriolo, G. (2020). Mpc for humanoid gait generation: Stability and feasibility. *IEEE Transactions on Robotics*, 36(4):1171–1188.

Siméon, T., Laumond, J.-P., Cortés, J., and Sahbani, A. (2004). Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):729–746.

Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., and Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics*, 34(3):586–601.

Winkler, A. W. (2018-05-14). *Optimization-based motion planning for legged robots*. Doctoral thesis, ETH Zurich, Zurich. This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics (NCCR Robotics).

Winkler, A. W., Bellicoso, C. D., Hutter, M., and Buchli, J. (2018). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567.

Part I

Offline Planning Strategies

Chapter 2

The Problem of Posture Generation

2.1 Introduction

Effective motion planning on highly redundant robots with a large number of Degrees of Freedom (DoFs) requires to satisfy multiple objectives and constraints concerning loco-manipulation and stability, while avoiding internal (self) and external collisions with the environment. As a consequence, the computational cost of a motion planning algorithm dramatically increases depending on the dimension of the state-space (i.e. DoFs number). For motion planners which directly search on joint space configurations, this could often lead to the impossibility to find a solution within a reasonable time (a.k.a. *curse of dimensionality*).

To overcome this issue, previous works adopt simplifying assumptions to reduce the state space dimension or use a *discrete control space* (i.e., *actions*) [Cognetti et al., 2015; Ferrari et al., 2018; Hauser et al., 2008]. However, these methods suffer from the trade-off between a small action set, which can reduce the branching factor of the search tree inhibiting specific motions, and a large action set, which increases the branching factor of the search tree that becomes harder to explore. Alternatively, continuous optimization based approach planners are used [Deits and Tedrake, 2014; Kuindersma et al., 2016; Ratliff et al., 2009] but they do not guarantee *completeness* or *global optimality* and it is non-trivial to generate optimal collision-free trajectories within time-frames acceptable for online planning, especially in complex environments.

A further possibility is to use *footstep/multi-contact planners* [Bouyarmane and Kheddar, 2012; Hauser et al., 2005; Kuffner et al., 2001; Tonneau et al., 2018], which have been widely applied in biped robots. For these approaches, the state space is reduced to consider only position and orientation of each contact as working variable. The price to pay is the necessity to move back to the configuration space through a map that associates a whole-body

configuration of the robot to a specific set of contacts (i.e., stance) coming from the planner. Indeed, collision with the ambient, self-collisions and equilibrium are constraints to be considered when generating a posture projected onto the planned contacts, with the risk to invalidate the sampled state if a feasible configuration cannot be found. Furthermore, adjacent configurations, with small differences between one another should be preferred in order to minimize avoidable motions during the transition between two consecutive stances.

Hence, footstep and multi-contact planners relies heavily on posture generators that have to satisfy not only the aforementioned constraints, but also being able to efficiently generate new postures. This Chapter introduces a novel and computationally efficient method to generate collision-free whole-body configurations of any kind of multi-limbed robots, while satisfying both kinematics, i.e. contacts and joint limits, static balance, and contacts stability constraints. The algorithm has been designed to be used coupled with footsteps and multi-contact planners in order to speed-up the whole contact and motion planning phase. The proposed method is first validated in simulations on the hyper-redundant hybrid wheeled-legged quadrupedal robot CENTAURO [Kashiri et al., 2019], and on the biped robot COMAN+ [Ruscelli et al., 2020] to prove the generality of the algorithm independently from the number of considered end-effectors or complexity of the robotic platform. An experimental assessment of the proposed method is also carried out employing CENTAURO's perception system (Lidar sensor) for perceiving the environment in front of the robot.

2.2 Related Work

The generation of feasible whole-body configurations for a legged robot, coupled with footsteps or multi-contact planners has been widely investigated in the past years. Previous works are based on pre-computed paired forward-inverse Dynamic Reachability Maps (DRM/iDRM) to sample among the reachable workspace those configurations that could accomplish a loco-manipulation task while guaranteeing stability and collision safeness on flat [Yang et al., 2017] and cluttered [Ferrolho et al., 2018] terrains. This method is characterized by big computational and memory costs, which are limited solving for the upper and lower body separately. Additionally, it requires the computation and discretiazion of the reachable workspace which becomes computationally heavier when the number of contacts increase.

In Hauser et al. [2005] an Iterative Constraint Enforcement algorithm (ICE) was used to generate statically-stable and collision free configurations using Newton-Raphson. The generated postures are subject to Cartesian constraints for the contacts and CoM position to

guarantee stability, starting from randomly sampled initial configurations. However, random seed configurations do not take into account the problem of minimal displacement between adjacent postures and this could lead to unfeasibilities during the transition motion.

Other approaches explore fixed-size [Deits and Tedrake, 2014; Gutmann et al., 2005] or variable-size [Buchanan et al., 2019] bounding-boxes to find the best collision free walking-posture. This whole-body posture is then projected onto the contacts found during the planning phase. However, these methods do not take advantage of the capability to reshape the whole body of the robot to facilitate and eventually permit locomotion in scenarios where the dimensions of the free passage are closely the physical dimensions of the robot body.

Bouyarmane and Kheddar [2012] used a non-linear optimization to compute IK with static stability, collision avoidance, torque limits and joint limits as constraints. In this work, no further modifications are carried out when the solver is not able to find a solution, leading to a possible avoidable discard of the sampled contact state.

A different approach was used by Tonneau et al. [2018]. The contact planner problem is addressed first finding a guide path for the floating base in the $SE(3)$ configuration space while satisfying a reachable condition to guarantee collision safeness and workspace reachability of the end-effectors. Then, a sequence of discrete configurations is computed using an iterative algorithm that satisfies a specific contact transition, stability and collision safeness starting from the root guide path. Ultimately, the contact sequence is retrieved from the configuration sequence. However, this method relies on a pipeline which may suffer of a necessary fine tuning of its parameters, especially for the effectiveness of the reachable condition which strictly depends on the kinematic characteristics of the robot.

Recently, Shigematsu et al. [2019] developed a posture generator to plan whole-body trajectories for a humanoid robot moving heavy suitcases. The approach is based on a non-linear program where several key postures are optimized all together with the centroidal statics, joint limits and self-collision constraints. Despite the impressive results obtained on the real platform, the method does not account for environmental collisions and it still needs several minutes to compute a sequence of configurations.

To address these concerns, this Chapter proposes a novel posture generator algorithm based on the hierarchical inverse kinematics (HIK), called *Null-Space Posture Generator* (NSPG), able to generate collision-free and statically balanced configurations for arbitrarily complex floating-base robots. In particular, the NSPG exploits the null-space of the robot, which is used to locally correct its posture around a *nominal configuration*, generated starting from the history of previously computed feasible postures. Further, the previous configuration becomes also the seed configuration of the HIK solver, guaranteeing minimal differences

between adjacent postures. If a chain of the robot is in contact with the environment or in self-collision, instead of generating a new whole posture, only the involved kinematic chain is moved in order to avoid the collision, in the neighborhood of the nominal configuration. Hence, the algorithm takes advantage of the kinematic structure of multi-limbed robots to restore the feasibility. Differently, when the generated posture is statically unstable, only the floating base is moved in order to restore feasibility.

The method, compared with previous works, contributes with a smart selection of the seed and nominal configurations for the HIK solver, seeking for a new feasible one that keeps minimum differences between adjacent stances. Specifically, it exploits the robot workspace in random directions, moving in the neighborhood of the nominal configuration. This allows the posture generator to look for a feasible configuration instead of discarding the state as soon as the first computed configuration is unfeasible, improving the performance of the planner and the algorithm itself. Last, the number of parameters to be tuned is kept minimum and is independent from the environment or the robot (a larger discussion will be given in Sec. 2.5). A similar approach has been used by Yang et al. [2016] in which feasible postures where generated using the IK randomly sampling seed configurations from the balanced manifold. However, stability is checked using the projection of the CoM onto the support polygon drawn by the feet, thus not considering non-co-planar contacts.

2.3 Background

In this section, the main tools used to develop the posture generator will be described.

2.3.1 Stance Generation

The aim of the NSPG is to generate whole-body configurations realizing both kinematic and statics constraints that will be detailed in Sec. III-B and III-C. These constraints arise from the contacts that the robot is required to establish with the environment in order to execute an assigned task, and are generally the output of a contact planner. To this end, this section introduces the basic notions that will be used in the following:

- A *configuration* $\boldsymbol{q} \in \mathbb{SE}(3) \times \mathbb{R}^n = \mathscr{Q}$ is an element of the robot configuration space containing the n joint positions and the pose of the floating-base w.r.t. the inertial frame. $\boldsymbol{q}_j \in \mathbb{R}^n$ expresses the joint positions. Additionally, $\mathscr{Q}$ is partitioned by two sub-set $\mathscr{Q}_{\text{feas}}$ and $\mathscr{Q}_{\text{unfeas}}$ containing the feasible and unfeasible configurations respectively so that $\mathscr{Q}_{\text{feas}} \cap \mathscr{Q}_{\text{unfeas}} = \emptyset$;

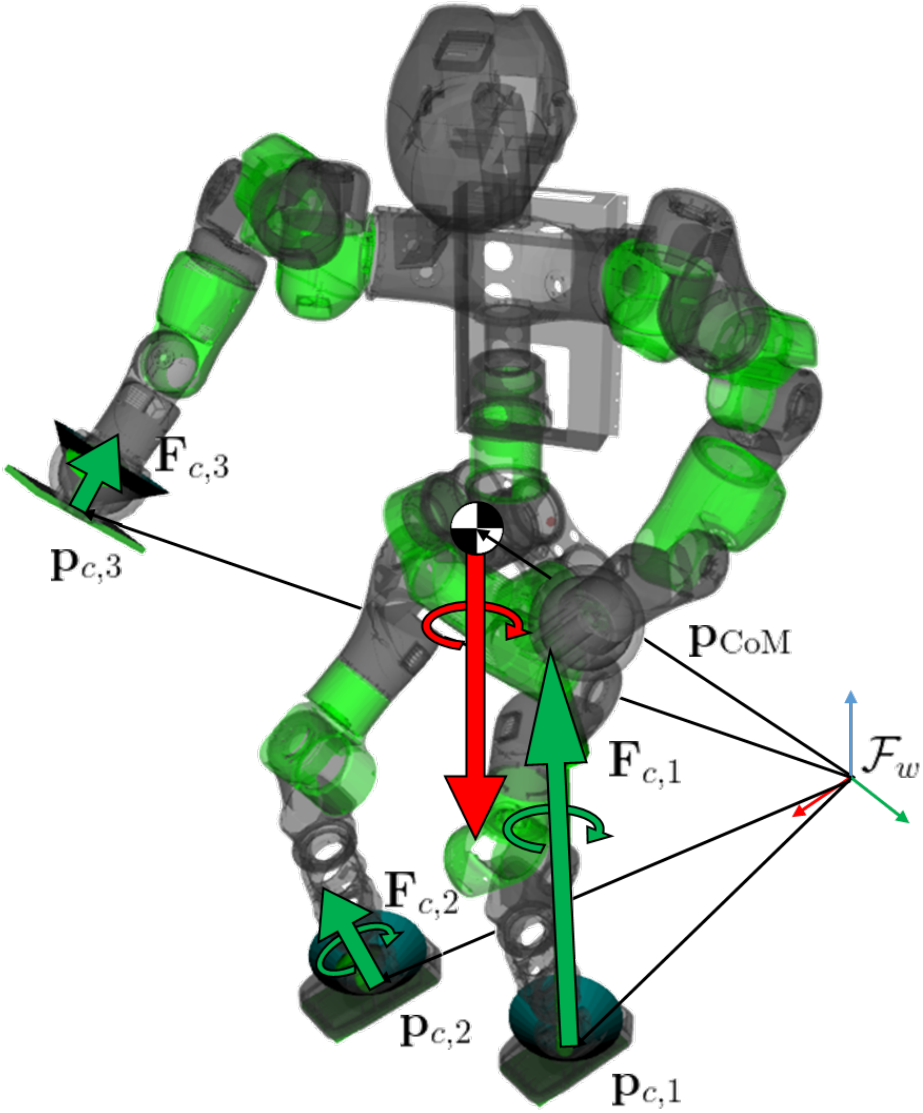


Figure 2.1 Centroidal statics and robot configuration starting from planned contacts. The green arrows represent the contact forces and torques. These lasts are limited to the surface contacts only (feet). The friction cones are also shown for the three active contacts. The red arrow are the weight force and the derivative of the angular momentum exerted directly on the CoM of the robot. Position vectors of each contact and CoM are highlighted w.r.t. the inertial frame $\mathcal{F}_w$.

- A *stance* $\sigma = \{\mathtt{c}_1, \ \ldots \ , \mathtt{c}_k\}$ is a set of k contacts where each $\mathtt{c}_k = \langle {}^w\boldsymbol{T}_{\mathrm{c,k}}, \mathrm{ID}_{\mathrm{c,k}}, \mathrm{CT}_{\mathrm{c,k}} \rangle$ contains the pose of the k-th contact ${}^w\boldsymbol{T}_{\mathtt{c}_i}$ w.r.t. the inertial frame, the contact's name $\mathrm{ID}_{\mathrm{c,k}}$ and its type $\mathrm{CT}_{\mathrm{c,k}}$ (i.e., point or surface contact);
- A configuration $\boldsymbol{q}$ is *compliant* with a stance σ if it realizes all the contact poses specified by σ, i.e., $\boldsymbol{k}(\mathtt{c}_i, \boldsymbol{q}) = {}^w\boldsymbol{T}_{\mathtt{c}_i}$ for all $\mathtt{c}_i \in \sigma$. A pair consisting of a stance σ and a compliant configuration $\boldsymbol{q}$ defines a *state* s:

$$s = \langle \sigma, \boldsymbol{q} \rangle. \tag{2.1}$$

For each state s_i, given the stance σ_i, the posture generator aims to find a feasible configuration $\boldsymbol{q}_i$ compliant with σ_i. The path of stances is instead found by a generic footstep or multi-contact planner. However, the description of the planner algorithm is out of the scope of this work.

The configuration space velocities associated to the configurations $\boldsymbol{q}$ are denoted with $\boldsymbol{v} \in \mathbb{R}^{n+6}$ that contains the joint space velocities and the linear and angular base velocities:

$$\boldsymbol{v} = \begin{bmatrix} \dot{\boldsymbol{p}}_b \\ \boldsymbol{\omega}_b \\ \dot{\boldsymbol{q}}_j \end{bmatrix}. \tag{2.2}$$

In addition, proper validity functions ensures to sample stances with poses of the contacts in the workspace of the robots and not inside any obstacle.

2.3.2 Hierarchical IK

The Cartesian velocity ${}^w\boldsymbol{v}_e \in \mathbb{R}^6$ of an end-effector frame $\mathscr{F}_e$ w.r.t. a reference $\mathscr{F}_w$ is related to $\boldsymbol{v}$ through the relation

$${}^w\boldsymbol{v}_e = {}^w\boldsymbol{J}_{w,e}\,\boldsymbol{v}, \tag{2.3}$$

where ${}^w\boldsymbol{J}_{w,e} \in \mathbb{R}^{6\times(6+n)}$ is the Jacobian[1] of the frame $\mathscr{F}_e$ w.r.t. $\mathscr{F}_w$ expressed in $\mathscr{F}_w$.

The inverse problem of (2.3), a.k.a. *differential inverse kinematics*, permits to compute the configuration velocities $\boldsymbol{v}^*$ which realizes a desired Cartesian velocity $\boldsymbol{v}_d$ for a certain end-effector. The computation of $\boldsymbol{v}^*$ is classically found solving a *least square* problem in

[1] Here and in what follows, for the sake of brevity of the notation, the dependence of the matrices on the configuration $\mathbf{q}$ will not be expressed.

the form:

$$\boldsymbol{\nu}^* \in \text{argmin} \quad \|\boldsymbol{J}\boldsymbol{\nu} - \boldsymbol{\nu}_d\|^2_{\boldsymbol{W}}, \tag{2.4}$$

with $\boldsymbol{W} \in \mathbb{R}^{6\times 6}$ a weight matrix. In order to track Cartesian poses as well, a Closed Loop IK (CLIK) scheme is often employed were the desired Cartesian velocity $\boldsymbol{\nu}_d$ is set as:

$$\boldsymbol{\nu}_d = \boldsymbol{\nu}_r + \lambda \boldsymbol{e}(\boldsymbol{T}_r, \boldsymbol{T}), \tag{2.5}$$

with $\boldsymbol{\nu}_r$ a feed-forward Cartesian velocity reference, $\boldsymbol{T}_r$ a reference Cartesian pose, $\boldsymbol{T}$ the actual Cartesian pose and λ a gain which ensures exponential convergence of the Cartesian error $\boldsymbol{e}(\cdot)$ to zero. The configuration velocities computed using (2.4) can be integrated to obtain the new robot configuration, through the integration function $\boldsymbol{I}$:

$$\boldsymbol{q}_k = \boldsymbol{I}(\boldsymbol{q}_{k-1}, \boldsymbol{\nu}, dt). \tag{2.6}$$

The problem in (2.4) can be formulated as a Quadratic Programming (QP) problem with the main advantage to consider equality and inequality *constraints* as well [Kanoun et al., 2011]:

$$\begin{aligned} \min_{\boldsymbol{\nu}} \ & \|\boldsymbol{J}\boldsymbol{\nu} - \boldsymbol{\nu}_d\|^2_{\boldsymbol{W}} + \varepsilon\|\boldsymbol{\nu}\|^2 \\ s.t. \ & \boldsymbol{A}_{eq}\boldsymbol{\nu} = \boldsymbol{b}_{eq} \\ & \boldsymbol{A}\boldsymbol{\nu} \leq \boldsymbol{b}. \end{aligned} \tag{2.7}$$

Furthermore, *hard* priorities between tasks can be enforced in the QP-based IK by means of a cascade of QPs [Kanoun et al., 2009] or using particular hierarchical orthogonal decomposition of the aggregated task matrices [Escande et al., 2014].

The methodology requires the definition of the following tasks and constraints:

- **Contact Task** $\mathscr{T}_c$ that projects the robot into the manifold defined by the contact stances σ. For example, the surface contact task is defined as:

$$\mathscr{T}_c^s := \|{}^w\boldsymbol{J}_c^s\boldsymbol{\nu} - \lambda_c \boldsymbol{e}({}^w\boldsymbol{T}_{c,d}, {}^w\boldsymbol{T}_c)\|^2 \tag{2.8}$$

  with ${}^w\boldsymbol{J}_c^s \in \mathbb{R}^{6\times(n+6)}$, while the point contact task is defined as:

$$\mathscr{T}_c^p := \|{}^w\boldsymbol{J}_c^p\boldsymbol{\nu} - \lambda_c({}^w\boldsymbol{p}_{c,d} - {}^w\boldsymbol{p}_c)\|^2 \tag{2.9}$$

  with ${}^w\boldsymbol{J}_c^p \in \mathbb{R}^{3\times(n+6)}$ and ${}^w\boldsymbol{p}_c \in \mathbb{R}^3$ the position of the contact.

- **Postural Task** $\mathscr{T}_{\boldsymbol{\nu}}$ that tracks a desired configuration velocity $\boldsymbol{\nu}_d$ of the robot. The postural task is defined as:

$$\mathscr{T}_{\boldsymbol{\nu}} := \|\boldsymbol{\nu} - \boldsymbol{\nu}_d\|^2. \tag{2.10}$$

  As done in the Cartesian case (2.5), it is possible to define a desired configuration velocity with a term that tracks a reference robot configuration $\boldsymbol{q}_r$:

$$\boldsymbol{\nu}_d = \boldsymbol{\nu}_r + \lambda_{\boldsymbol{\nu}} \boldsymbol{e}(\boldsymbol{q}_r, \boldsymbol{q}). \tag{2.11}$$

- **Joint Limits Constraint** $\mathscr{C}_{\boldsymbol{q}_j}$ permits to take into account hardware joint limits present in the considered robotic platform. The joint limits constraint is an inequality constraint in the form:

$$\mathscr{C}_{\boldsymbol{q}_j} := \frac{\underline{\boldsymbol{q}}_j - \boldsymbol{q}_j}{dt} \leq \dot{\boldsymbol{q}}_j \leq \frac{\overline{\boldsymbol{q}}_j - \boldsymbol{q}_j}{dt}, \tag{2.12}$$

  with $\underline{\boldsymbol{q}}_j \in \mathbb{R}^n$ and $\overline{\boldsymbol{q}}_j \in \mathbb{R}^n$ respectively the lower and upper joint limits. The dt is the integration time used in (2.6).

Tasks and constraint are organized in the following stack $\mathscr{S}$:

$$\mathscr{S} := \left[\left(\sum_{i=1}^{k} \mathscr{T}_{c,i} \right) / \mathscr{T}_{\boldsymbol{\nu}} \right] \ll \mathscr{C}_{\boldsymbol{q}_j}, \tag{2.13}$$

where the "$\sum$" symbol means that all the contact tasks are summed at the same priority level, the "/" symbol means that the postural task acts in the null-space of the contact tasks. The "$\ll$" symbol means that all the tasks are subject to the joint limits constraint. This formulation, known as *Math of Tasks*, follows the work done in [Hoffman and Tsagarakis, 2021].

2.3.3 Centroidal Statics

To grant quasi-static stability for a given robot configuration $\boldsymbol{q}_i$, compliant with a stance σ_i, a critical role is played by the interaction forces. Static stability is checked by solving another QP based on the stances' information of contact position $\boldsymbol{p}_c$, its associated normal $\boldsymbol{n}_c$ and CoM position $\boldsymbol{p}_{\text{CoM}}$ computed from the configuration to be checked.

The resulting QP in the variables $\boldsymbol{x} = \boldsymbol{F}_c$, with $\boldsymbol{F}_c$ being all the contact wrenches w.r.t. the inertial frame[2], is formulated as:

$$\min_{\boldsymbol{x}} \|m\boldsymbol{g} + \boldsymbol{G}_{\text{CD}} \boldsymbol{F}_c\|^2_{\boldsymbol{W}_1} + \|\boldsymbol{F}_c\|^2_{\boldsymbol{W}_2} \tag{2.14a}$$

[2]For a point contact $\boldsymbol{F}_{c,i} \in \mathbb{R}^3$, while for a surface contact $\boldsymbol{F}_{c,i} \in \mathbb{R}^6$.

s.t.

$$\underline{\boldsymbol{F}} \leq \boldsymbol{F}_c \leq \overline{\boldsymbol{F}} \tag{2.14b}$$

$$\boldsymbol{C}\boldsymbol{F}_c \leq \boldsymbol{0} \tag{2.14c}$$

if $\mathtt{c_i}$ is surface contact :

$$\boldsymbol{P}\boldsymbol{F}_{c,i} \leq \boldsymbol{0} \tag{2.14d}$$

$$\boldsymbol{N}\boldsymbol{F}_{c,i} \leq \boldsymbol{0}. \tag{2.14e}$$

The first term in (5.28) ensures static stability, under quasi-static conditions, based on the Centroidal Statics (CS) of the robot, where the terms $\boldsymbol{g} \in \mathbb{R}^6$ and $m \in \mathbb{R}$ are the vector of the gravity acceleration and momentum variation, and the mass of the robot, respectively, and $\boldsymbol{G}_{\text{CD}} \in \mathbb{R}^{6\times k}$ is the the centroidal dynamics grasp matrix:

$$\boldsymbol{G}_{\text{CD}} = \begin{bmatrix} \boldsymbol{I}_3 & \cdots & \boldsymbol{I}_3 \\ \boldsymbol{S}_\times(\boldsymbol{p}_{\text{CoM}} - \boldsymbol{p}_{c,1}) & \cdots & \boldsymbol{S}_\times(\boldsymbol{p}_{\text{CoM}} - \boldsymbol{p}_{c,k}) \end{bmatrix}, \tag{2.15}$$

with $\boldsymbol{S}_\times$ the skew-symmetric matrix operator.

This reduced description assumes fixed contact placements with associated linearized friction models and unilaterality of the contact force (5.21), Center of Pressure (CoP) inside contact surface (3.12) and bounded contact yaw torque (2.14e) [Caron et al., 2015][3], to obtain the interaction forces required to compensate for gravity, achieve static balancing and non-slippage of surface contacts. Matrices $\boldsymbol{C}$, $\boldsymbol{P}$ and $\boldsymbol{N}$ are expressed as $\boldsymbol{C} = \boldsymbol{C}_i \cdot \boldsymbol{R}_{\text{adj}}$, $\boldsymbol{N} = \boldsymbol{N}_i \cdot \boldsymbol{R}_{\text{adj}}$ and $\boldsymbol{P} = \boldsymbol{P}_i \cdot \boldsymbol{R}_{\text{adj}}$ with:

$$\boldsymbol{C}_i = \left[\begin{array}{ccc|c} 1 & 0 & -\mu_i & \\ -1 & 0 & -\mu_i & \\ 0 & 1 & -\mu_i & \boldsymbol{0}_{5\times 3} \\ 0 & -1 & -\mu_i & \\ 0 & 0 & -1 & \end{array}\right], \tag{2.16a}$$

$$\boldsymbol{P}_i = \begin{bmatrix} 0 & 0 & x & 0 & 1 & 0 \\ 0 & 0 & -x & 0 & -1 & 0 \\ 0 & 0 & y & -1 & 0 & 0 \\ 0 & 0 & -y & 1 & 0 & 0 \end{bmatrix}, \tag{2.16b}$$

[3]Constraints (3.12) and (2.14e) are considered only for surface contacts

$$\boldsymbol{N}_i = \begin{bmatrix} -y & -x & -\mu(x+y) & -\mu & -\mu & 1 \\ -y & x & -\mu(x+y) & -\mu & \mu & 1 \\ y & -x & -\mu(x+y) & \mu & -\mu & 1 \\ y & x & -\mu(x+y) & \mu & \mu & 1 \\ -y & -x & -\mu(x+y) & \mu & \mu & -1 \\ -y & x & -\mu(x+y) & \mu & -\mu & -1 \\ y & -x & -\mu(x+y) & -\mu & \mu & -1 \\ y & x & -\mu(x+y) & -\mu & -\mu & -1 \end{bmatrix}, \tag{2.16c}$$

$$\boldsymbol{R}_{\text{adj,i}} = \begin{bmatrix} {}^{w}\boldsymbol{R}_i & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & {}^{w}\boldsymbol{R}_i \end{bmatrix}, \tag{2.16d}$$

where $\boldsymbol{C}_i$ and $\boldsymbol{P}_i$ are the coefficient matrices of the constraints inequalities expressed in local force frame, and $\boldsymbol{R}_{\text{adj,i}}$ being the adjoint rotation matrix that transforms the wrench from the i-th local frame $\mathscr{F}_i$ to the inertial frame $\mathscr{F}_w$, computed from the contact normal $\boldsymbol{n}_{c,i}$. In particular, μ_i is the static friction coefficient associated with the i-th contact, x and y are half the size of the surface contact[4], and ${}^{w}\boldsymbol{R}_i$ is the rotation matrix that moves from the i-th contact frame $\mathscr{F}_i$ to the inertial frame $\mathscr{F}_w$.

It is worth noticing that the modeling a surface contact using forces and moments, together with the constraints (5.21), (3.12) and (2.14e) permit to save variables and constraints. In fact, assuming 4 contact points per surface contact leads to a total of 12 pure contact forces variables and 20 constraints in the form of (2.16a). On the other hand, assuming a single wrench leads to 6 variables to describe contact forces and torques, and 17 constraints. A graphical representation of the centroidal statics' components, is given in Fig. 2.1.

The residual of the first term in the cost function (5.28) is used to decide whether a configuration is stable or not when satisfying a specific stance, depending on a threshold value.

2.4 Generating Transition Configurations

In a contact planner application case, a feasible sequence of adjacent postures is required to be connectable in order to build a configuration path that moves the robot safely from a start configuration $\boldsymbol{q}_{\text{start}}$ to a goal configuration $\boldsymbol{q}_{\text{goal}}$. In particular, two configurations $\boldsymbol{q}_i$ and $\boldsymbol{q}_{i+1}$ are connectable if there exist a continuous path $\varphi(l)$ satisfying σ_i, σ_{i+1}, and the requirements of stability and collision avoidance. Further, a local planner interpolator

[4]Here the contact frame is assumed at the center of the surface for simplicity and modeled as rectangular.

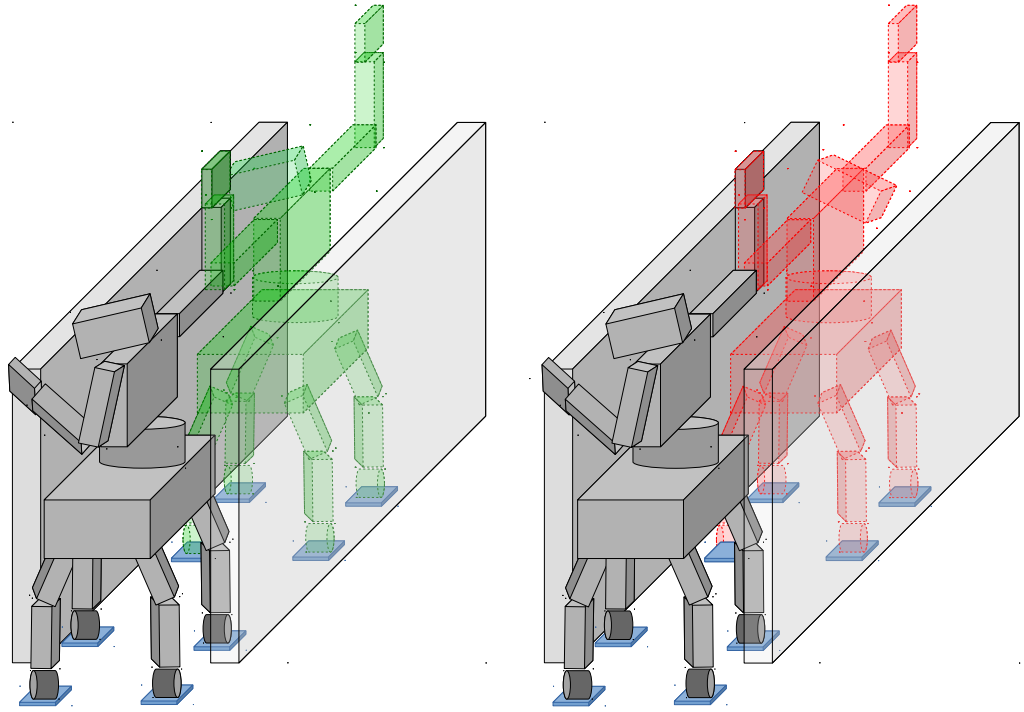


Figure 2.2 An example of how generating adjacent poses with large differences can lead to an unfeasible colliding transition during their connection. Starting from the same robot configuration in grey, the next posture is generated minimizing the differences w.r.t. the parent state, on the left, or facing the opposite side of the corridor with the upper body, on the right. Even though all the generated poses are feasible, the right scenario will lead to a collision with the environment while connecting the two consecutive states.

guarantees a feasible trajectory between two consecutive configurations. Having defined $\mathscr{Q}_{\sigma_i}$ the set of all configurations that satisfies σ_i, two consecutive configurations are connectable if $\exists\ \varphi : [0,l] \to \mathscr{Q}_{\text{free}}$ such that:

$$\varphi \in C^0 \tag{2.17a}$$

$$\varphi(0) \in \mathscr{Q}_{\sigma_i} \tag{2.17b}$$

$$\varphi(l) \in \mathscr{Q}_{\sigma_{i+1}}. \tag{2.17c}$$

Collision avoidance is sought generating similar adjacent poses, thus minimizing the transition motion that moves the robot from $\boldsymbol{q}_i$ to $\boldsymbol{q}_{i+1}$. In order to better understand this last requirement, imagine a robot side-walking in a narrow space. In this scenario, feasible postures can be both the one with the robot facing left-ward and right-ward. However, if two adjacent σ_i and σ_{i+1} contains configurations that face opposite sides of the narrow passage, the transition motion between $\boldsymbol{q}_i$ and $\boldsymbol{q}_{i+1}$ will probably collide with the environment, see Fig. 2.2. This issue is solved using the parent state's configuration $\boldsymbol{q}_{i-1}$ as nominal configuration for the generation of $\boldsymbol{q}_i$ thus forcing $\boldsymbol{q}_i$ to be in a small neighborhood of the $\boldsymbol{q}_{i-1}$ (Sec. 2.5). This assumption works in the hypothesis of small changes of the environment seen by the robot, which covers the most of the considered scenarios. Indeed, when moving in such environment, the previous feasible configuration is a first good guess to generate the next feasible configuration. In this way, any transition motion is generated only if required.

Furthermore, in the quasi-static assumption that near stances differ by exactly one active contact, the generated configuration $\boldsymbol{q}_i$ must be statically stable w.r.t. the minimum contacts number stance between σ_{i-1} and σ_i [Escande et al., 2013].

2.5 Null-Space Posture Generator

The approach is based on a complete reshape of the robot configuration, obtained by adjusting the pose of kinematic chains in collision, or moving the floating base to recover the static stability, in the null-space of the Cartesian (contact) tasks. Specifically, each detected unfeasibility will generate random velocities components aiming to recover the feasibility. This section follows a pipeline going through each component of the algorithm: first, the nominal configuration and the random velocity vector $\boldsymbol{v}$ are generated. The NSPG exploits the neighborhood of the nominal configuration in the random direction defined by $\boldsymbol{v}$. Then, the procedure to adapt the velocity vector and the candidate configuration procedures are described. The strategy used is described in Algorithm 1, while a graphical representation is

Algorithm 1: NSPG()

input: s_{i-1}, σ_i;
parameters: N, dt, T;
take $\boldsymbol{q}_{i-1} \in s_{i-1}$;
$\boldsymbol{q}_{\text{pose}} = \text{solveIK}(\sigma_{\text{i}}, \boldsymbol{q}_{\text{i}-1})$;
$C \leftarrow \text{collidingChains}(\boldsymbol{q}_{\text{pose}})$;
iter = 0;
t = 0;
repeat
  if iter % N == 0 **then**
    $\boldsymbol{q}_{\text{old}} = \boldsymbol{q}_{i-1}$;
    for *i in C* **do**
      if *i is constrained* **then**
        $\dot{\boldsymbol{p}}_b = \text{random}()$;
        $\boldsymbol{v} = \begin{bmatrix} \dot{\boldsymbol{p}}_b & \boldsymbol{0}_{1\times 3} & \boldsymbol{0}_{1\times n} \end{bmatrix}$;
      else
        $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{0}_{1\times 6} & C(i)\cdot \text{random}() \end{bmatrix}$;
      end
    end
    res_{CC} =checkStability($\boldsymbol{q}_{\text{old}}$);
    if $\text{res}_{\text{CC}} > \varepsilon_{\text{CC}}$ **then**
      $\boldsymbol{v} = \begin{bmatrix} \dot{\boldsymbol{p}}_b & \boldsymbol{0}_{1\times 3} & \boldsymbol{0}_{1\times n} \end{bmatrix}$;
    end
  else
    $\boldsymbol{q}_{\text{new}} = \boldsymbol{I}(\boldsymbol{q}_{\text{old}}, \boldsymbol{v}, dt)$;
    $[\boldsymbol{q}_{\text{pose}}, \text{res}_{\text{IK}}] \leftarrow \text{solveIK}(\sigma_{\text{i}}, \boldsymbol{q}_{\text{new}})$;
    if $\text{res}_{\text{IK}} > \varepsilon_{\text{IK}}$ **then**
      iter++;
      continue;
    end
  end
  $\boldsymbol{q}_{\text{old}} = \boldsymbol{q}_{\text{new}}$;
  $\boldsymbol{v} = \text{updateVel}(C, \boldsymbol{v}, \boldsymbol{q}_{\text{pose}})$;
  iter++;
until checkCollision($\boldsymbol{q}_{\text{pose}}$) $\wedge$ checkStability($\boldsymbol{q}_{\text{pose}}$) $\wedge$ t < T;
if t < T **then**
  $\boldsymbol{q}_i = \boldsymbol{q}_{\text{pose}}$;
  return true;
else
  return false;
end

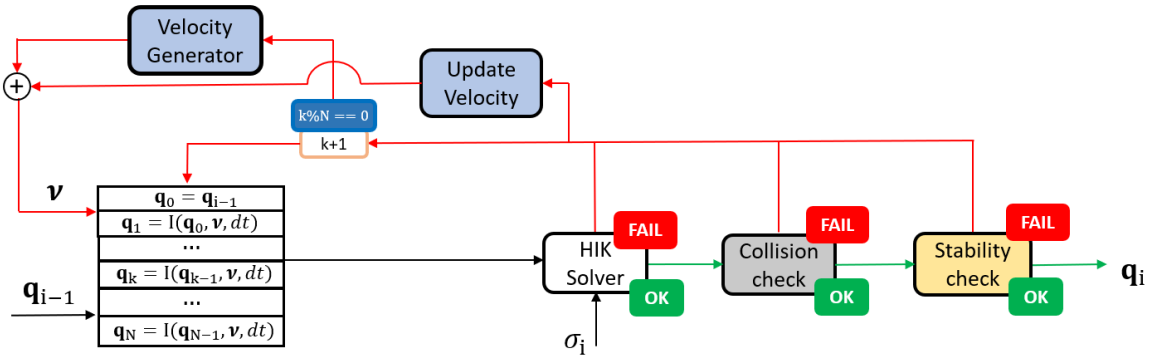


Figure 2.3 NSPG flow diagram: the configuration $\boldsymbol{q}_{i-1}$ is used as nominal posture for the HIK solver and at each iteration it is modified depending on the random velocity vector $\boldsymbol{\nu}$ generated, until a feasible configuration is found. At each iteration, the velocity vector $\boldsymbol{\nu}$ is updated following Sec. 2.5.4. Notice that the nominal posture is reset to $\boldsymbol{q}_{i-1}$ every N iterations. σ_i defines the manifold for the HIK solver while green and red arrows define the flow when the validity checks and the HIK projector succeed or fail respectively.

given in Fig. 2.3.

2.5.1 Nominal Configuration Generation

First, the candidate configuration $\boldsymbol{q}_{\text{pose}}$ is computed projecting the *seed configuration* $\boldsymbol{q}_{i-1}$ onto the manifold defined by the stance σ_i. The projection is performed by the HIK solving the stack in (2.13) with the same $\boldsymbol{q}_{i-1}$ used as reference for the postural task (line 4)[5]. In the case the first candidate configuration $\boldsymbol{q}_{\text{pose}}$ is feasible, and no further adjustment are required, the NSPG will return $\boldsymbol{q}_i = \boldsymbol{q}_{\text{pose}}$. Oppositely, $\boldsymbol{q}_{\text{pose}}$ will be used as nominal configuration in which neighborhood the NSPG will look for a new feasible configuration.

2.5.2 Adaptive Random Velocity Vector Generation

In the following, the procedure to generate the random velocity vector $\boldsymbol{\nu}$ will be described, depending on the unfeasibility.

Collisions

This phase takes advantage of the kinematic structure of a multi-limbed robot to avoid collisions while keeping minimum differences between adjacent configurations. Indeed, the motion of a kinematic chain is independent with respect to the others, assuming that its

[5]The $\boldsymbol{q}_{\text{start}} \in s_{\text{start}}$ is chosen as the *homing* configuration

end-effector is not constrained onto a contact pose. In this way, only the chain(s) involved in the unfeasibility will be moved, preventing avoidable motions that could lead to other collisions. In the details, when $\boldsymbol{q}_{\text{pose}}$ is in (self-)collision, the colliding kinematic chains are detected (line 5) and collected into the set C. For each of the joints belonging to those chains, a random bounded velocity vector, is generated as written in line 16. The random velocity vector comes from a uniform random distribution bounded between the joint velocity limits. $C(i)$ returns a $1 \times n$ vector that extracts the aforementioned joints from the whole joints' list.

It is worth mentioning that collisions/self-collisions can be avoided also integrating specific cost functions or constraints in the HIK [Fang et al., 2015; Stasse et al., 2008]. Despite appealing, this inclusion may have a not negligible computational cost which is avoidable when HIK is used as posture generator. For kinematic chains fully constrained onto the set of contacts defined by σ_i, their reshape is obtained through a linear displacement of the floating base as shown in line 14.

With this methodology, the NSPG moves the only joints involved in the unfeasibility preventing from avoidable motions of the rest of the body.

Stability

A second scenario occurs when $\boldsymbol{q}_{\text{pose}}$ is not statically stable (line 21). The static stability is checked solving the QP problem as written in (2.14) comparing the residual of the first term of the cost function with a threshold value ε_{CS}.

In this case linear velocities of the floating base are generated from a uniform random distribution bounded between two arbitrarily big numbers ($\pm 50 \frac{m}{s}$).

It has been chosen to move the floating base joints instead of the CoM directly since a motion of this last could imply an undesired whole-body motion involving non-colliding kinematic chains or deviating the motion of the colliding ones in an umpredictable way.

2.5.3 Candidate Configuration Update

The postural task reference is then updated according to the new velocity vector (line 24) and the HIK is solved generating a new robot configuration $\boldsymbol{q}_{\text{pose}}$ according to the new postural reference $\boldsymbol{q}_{\text{new}}$ (line 25). This procedure is repeated every N iterations, until a feasible pose is found, according to ε_{IK} and ε_{CS} which are chosen to be sufficiently small in order to guarantee the stable configurations, well projected onto the contacts manifold. The algorithm exploits the robot workspace in the direction defined by $\boldsymbol{v}$ for N iterations, after which the reference posture of the robot is reset to the starting one $\boldsymbol{q}_{i-1}$ (line 10).

2.5.4 Velocity Vector Adaptation

While running, the algorithm will generate configurations with arbitrarily small differences which depends on its parameters and velocity vector $\boldsymbol{\nu}$.

Collision and stability checks strictly depends on the current candidate configuration of the robot that changes at each iteration of the algorithm. Thus, the velocity vector $\boldsymbol{\nu}$ must be updated and adapted depending on the state of the robot throughout each iteration.

Specifically, at each iteration, the colliding chains are updated, and two new sub-sets are defined:

$$\mathscr{C}_{\text{new}} \leftarrow \text{collidingChains}(\boldsymbol{q}_{\text{pose}}) \tag{2.18a}$$

$$\mathscr{C}_{\text{more}} = \mathscr{C}_{\text{new}} \setminus \mathscr{C} \tag{2.18b}$$

$$\mathscr{C}_{\text{less}} = \mathscr{C} \setminus \mathscr{C}_{\text{new}} \tag{2.18c}$$

with $\mathscr{C}_{\text{more}}$ and $\mathscr{C}_{\text{less}}$ containing the set of the new and old colliding chains, respectively, depending on the new candidate configuration. In correspondence of $\mathscr{C}_{\text{more}}$, random velocity components are added to $\boldsymbol{\nu}$, while velocity components are removed depending on $\mathscr{C}_{\text{less}}$:

$$\boldsymbol{\nu} \mathrel{+}= \begin{bmatrix} \mathbf{0}_{1\times 6} & C_{\text{more}}(i) \cdot \text{random}() \end{bmatrix} \tag{2.19a}$$

$$\boldsymbol{\nu} = \begin{bmatrix} \dot{\boldsymbol{p}}_b & \mathbf{0}_{1\times 3} & C_{\text{less}}(i) \cdot \dot{\boldsymbol{q}}_j \end{bmatrix} \tag{2.19b}$$

with $\mathscr{C}_{\text{more}}(i)$ returning a $1 \times n$ vector containing 1 in correspondence of the joints belonging to the new colliding chains and 0 elsewhere, while $\mathscr{C}_{\text{less}}(i)$ returns a $1 \times n$ vector containing 0 in correspondence of the old colliding chains' joints and 1 elsewhere. Additionally, stability can be lost or recovered while generating new candidate configurations. In these cases, the velocity vector must be updated adding or removing velocity components corresponding to linear floating base velocities as follows:

$$\boldsymbol{\nu} \mathrel{+}= \begin{bmatrix} \dot{\boldsymbol{p}}_b = \text{random}() & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times n} \end{bmatrix} \tag{2.20a}$$

$$\boldsymbol{\nu} = \begin{bmatrix} \mathbf{0}_{1\times 6} & \dot{\boldsymbol{q}}_j \end{bmatrix} \tag{2.20b}$$

At each iteration, this method seeks minimal differences between the generated posture and the parent one. The NSPG algorithm moves only the joints involved in the unfeasibility of about a quantity that depends on the NSPG parameters, listed in Table 2.1. Increasing N

N	Reset condition
dt	Integration time
T	NSPG timeout
ε_{CS}	Centroidal statics threshold
ε_{IK}	IK threshold

Table 2.1 NSPG parameters

	dt	N	T	$\bar{t}$ [s]	$\bar{t}_{HIK}$ [ms]	$\bar{t}_{CS}$ [ms]	$\bar{t}_{CC}$ [ms]	$\bar{i}$	% success
CENTAURO	0.005	10	0.5	0.1248	0.4296	0.3585	0.1301	136	89.9%
			1	0.1337				146	95.0%
			2	0.3029				330	92.3%
COMAN+	0.005	10	0.5	0.1617	0.5173	0.1875	0.2375	172	85.0%
			1	0.2161				230	91.6%
			2	0.2927				311	93.9%

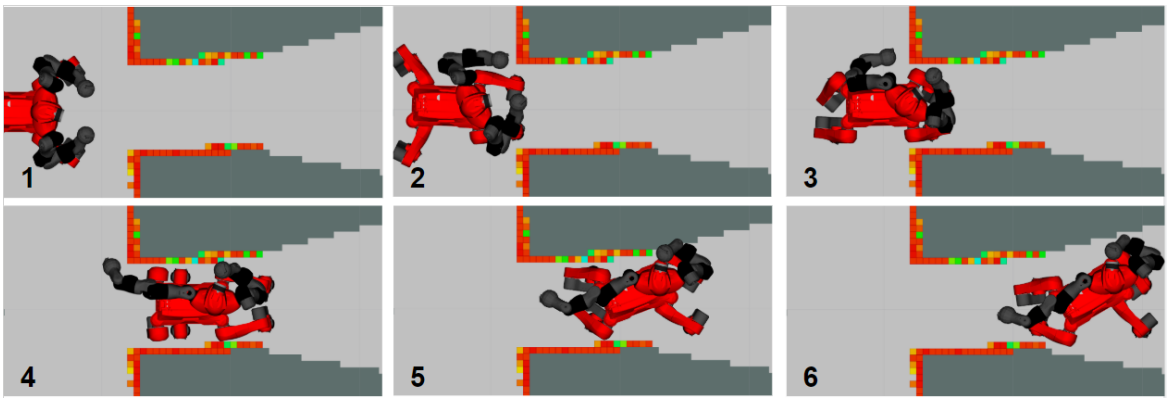
Table 2.2 Results from the scenario in Section 2.6.2: dt, N and T are the three parameters of the NSPG and $\bar{t}_{HIK}$, $\bar{t}_{CS}$, and $\bar{t}_{CC}$ are the average time required by the HIK solver, the Centroidal Statics and the Collision Check respectively, averaged on the three experiments, which do not depend on the parameter T. The average number of NSPG iterations to find a feasible solution are shown in the second-last column.

allows the robot to explore a larger range of motion around the nominal configuration. The parameter dt is the integration time involved in line 24: the smaller this parameter, the smaller will be the motion between two generated configurations during a single call of the NSPG. In addition, keeping N constant, the integration time dt will also influence the maximum range of motion around the nominal configuration. Ultimately, the timeout T sets a time threshold for the search of a feasible configuration.

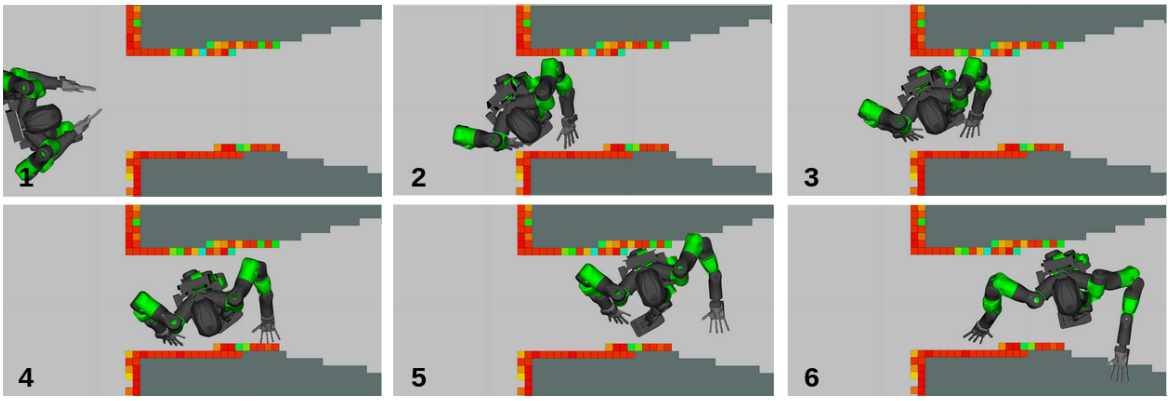
2.6 Results

The proposed NSPG algorithm has been tested in two scenarios with increasing difficulty, applied on two different types of legged redoundant robots: the hybrid wheeled-legged quadrupedal robot CENTAURO, and the biped robot COMAN+. CENTAURO is a robot with 39 DoFs distributed between a quadrupedal lower body and a bimanual humanoid upper body, while COMAN+ is a biped humanoid robot with 28 DoFs.

The first considered scenario consists in multiple tiles, placed at different heights and orientations, where the robot has to step on, or place its limbs, while the second one is a narrow corridor on flat terrain. Additionally, an experiment of this last scenario has been carried out on CENTAURO.



(a)



(b)

Figure 2.4 Screenshots of the CENTAURO 2.4a) and COMAN+ 2.4b) robots passing through the high narrow corridor

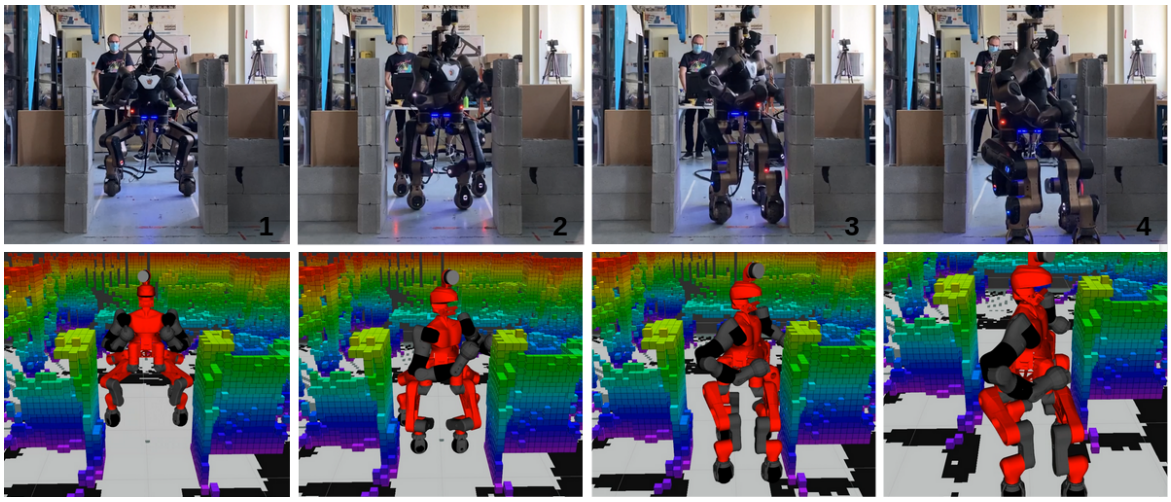


Figure 2.5 Screenshots from the experiment carried out on CENTAURO robot. From top left to bottom right: the robot first approaches the narrow corridor keeping the homing position as far as no collisions are detected. As soon as the robot starts entering the corridor, the NSPG reshapes the support polygon and the upper body at the same time.

The NSPG implementation is based on the OpenSoT [Hoffman et al., 2017] and CartesI/O [Laurenzi et al., 2019] frameworks for the computation of the whole-body HIK and centroidal statics QP problems, depicted in Sections 2.3.2 and 2.3.3 respectively. In particular, QPs are efficiently solved using well-known QP solvers such as *qpOASES* [Ferreau et al., 2014] or *OSQP* [Stellato et al., 2020]. Collisions are detected exploiting the *Flexible Collision Library* (FCL) [Pan et al., 2012] using convex-hull approximations of the links of the robot.

All video showing the presented simulations and real experiments are included in the material accompanying this thesis [6].

2.6.1 Non Co-Planar Contacts Scenario

The NSPG has been tested in the scenario where an external planner returns a series of feasible stances only. In this case, the contact state can be written as:

$$s = \langle \sigma \rangle. \tag{2.21}$$

Specifically, the humanoid robot has to climb a stair of three steps on a sequence of 18 stances. The first two steps are flat while the last two are rotated of 0.25 rad along the x-axis

[6] https://youtu.be/eEQbz8r5Z3s

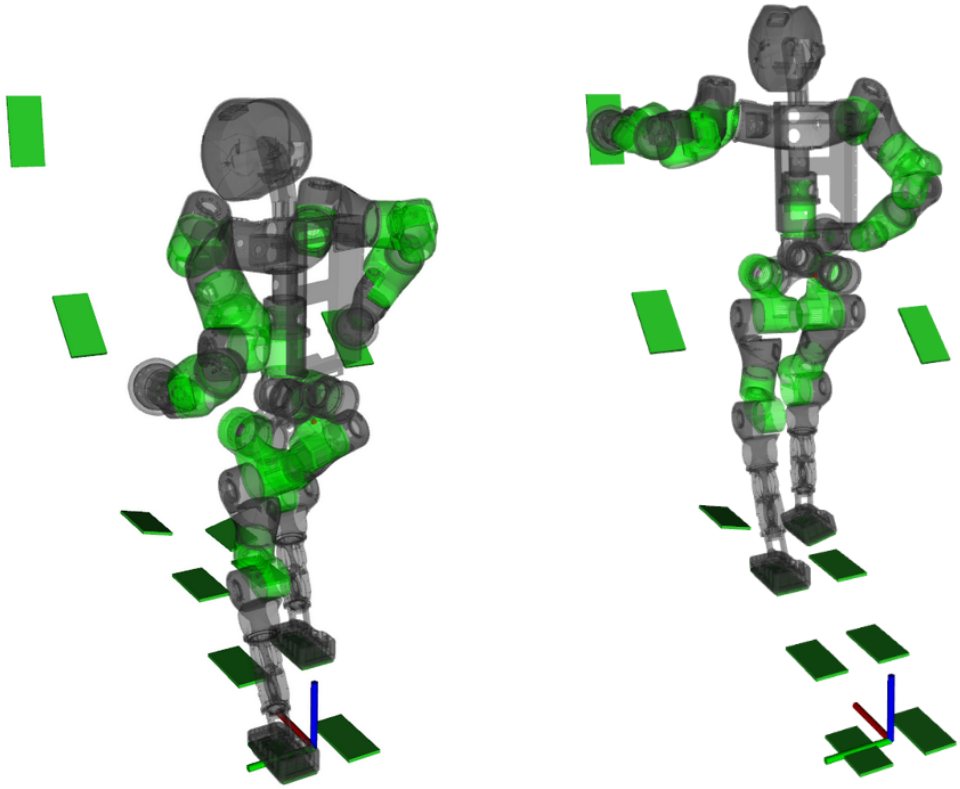


Figure 2.6 Captures of COMAN+ moving on a sequence of predefined stances while the NSPG generates feasible configurations

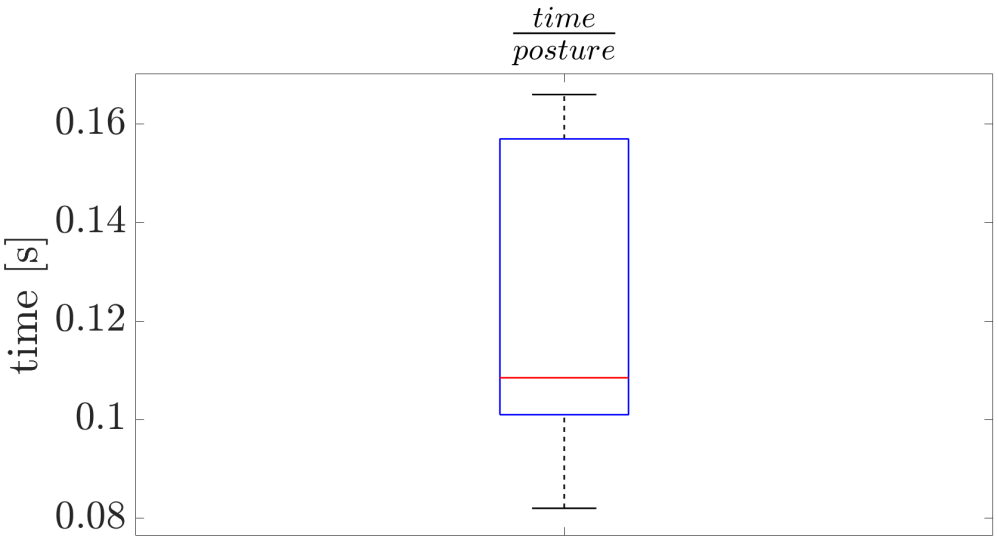


Figure 2.7 Results from the scenario in Section 2.6.1

(see Fig. 2.6). The stances are so that they respect the principle of connectivity described in Section 2.4 and the NSPG has to find a series of feasible configuration $\boldsymbol{q}_i$, each one corresponding to a specific stance σ_i, in the hypothesis that a feasible configuration exists for each stance. Differently from the previous scenario, the sequence of stances includes contacts with both the hands to enhance static stability. The NSPG is used after the planner and generates a sequence of configurations starting from a sequence of stances. Following the algorithm described in Section 2.5, the previous configuration $\boldsymbol{q}_{i-1}$ has been used as nominal configuration for the generation of $\boldsymbol{q}_i$, starting from a known *homing* configuration $\boldsymbol{q}_{\text{start}}$.

This scenario stresses the capability of the NSPG to find collision-free and stable configurations while stepping on non-co-planar stances, using a whole-body approach.

The NSPG performance is tested on 10 runs using the same sequence of stances. Results are collected in Fig. 2.7: the NSPG is always able to find all the 18 feasible configurations, changing the active links accordingly, in approximately 2.2 seconds with an average of 0.12 seconds for each generated configuration.

2.6.2 Corridor Scenario

The NSPG has been also tested in a scenario particularly tricky scenario for a sampled-based planner algorithm: a narrow corridor [Kingston et al., 2018]. In particular, CENTAURO and COMAN+ are asked to traverse a narrow corridor 0.7 m wide and and 1.8 m high. Taking advantage of the capability of CENTAURO to roll through the next stance instead of

taking a step, its active contacts do not change during the whole planning. In this scenario, CENTAURO proves the effectiveness of the methodology to generate (self-)collision free postures being the corridor wide approximately as the robot itself.

When planning with COMAN+ instead, locomotion is achieved by continuously switching between the two feet, i.e., the walking pattern. In particular, a sequence of single and double stances is computed by the planner and connectivity is guaranteed generating single-stance statically stable configurations. Indeed, the next double-stance configuration can be reached if and only if it is single-stance stable, avoiding the robot to go through an unstable region while walking (see Section 2.4). This is particularly challenging both from a stability and collision-safety point of view, since the robot has to move its CoM between the two stance-feet while moving in the corridor.

In this scenario, the NSPG is used inside a planner routine, implemented using OMPL [Şucan et al., 2012], to validate the sampled stances. Every time a new stance σ_i is sampled, the contact state $s_i = \langle \sigma_i, \boldsymbol{q}_i \rangle$ is added to the search tree if the NSPG has been able to compute a feasible whole-body configuration $\boldsymbol{q}_i$ in the given time T.

Three parameter setups were tried in this scenario to test how the performance of the NSPG changes. This was evaluated collecting data about the average time employed to find a feasible posture:

$$\bar{t} = \frac{\sum t_i}{m}, \tag{2.22}$$

with t_i being the time taken by the NSPG in a single call and m being the total NSPG calls. Additionally, the NSPG performance is evaluated considering also its rate of success and the average number of iterations the NSPG takes to find a feasible solution computed as:

$$\bar{i} = \frac{\bar{t}}{t_{\text{HIK}} + t_{\text{CS}} + t_{\text{CC}}}, \tag{2.23}$$

with t_{HIK}, t_{CS} and t_{CC} being the average time required by the HIK and the stability/collision check respectively. The integration time is kept fixed dt $= 0.005$ seconds, as well as the parameter N $= 10$ to guarantee small differences between adjacent configurations. The timeout T is varied between 0.5, 1, and 2 seconds. Intuitively, this variation in T should guarantee a higher success rate of the algorithm that is allowed to search a feasible configuration for a longer time. On the other hand, the mean time to find a single feasible posture increases the more the timeout is increased.

Results are collected in Table 2.2, which confirm the observations just done. Screenshots on the simulations with both COMAN+ and CENTAURO, are shown in Fig. 2.4. Real experiments with CENTAURO in this scenario are shown in Fig. 2.5. Both in the simulated

and real experiments, the surrounding environment is detected using perception data based on a 3D point cloud generated by a Lidar sensor. The data are collected following the work in [Hornung et al., 2013].

2.7 Conclusions

This first chapter presents a novel algorithm, named Null-Space Posture Generator (NSPG), able to efficiently generate stable and (self-)collision free whole-body postures for a generic, multi-limbed, floating-base robot, given a sequence of stances. The NSPG has been developed to speed up the whole-body motion planning of complex robotic systems when passing through particularly challenging environments keeping the tuning procedure as light as possible, proposing a solution for the problem of posture generation in multi-contact planning scenarios. Furthermore, it can be also used independently as a posture generator, given the active contacts as shown in Sec. 2.6.1 Multiple experiments on two profoundly different robotic platforms, COMAN+ and CENTAURO, demonstrated that the NSPG is capable to quickly generate stable and collision-free configurations for a legged robot in contact with the environment, exploiting null-space motions. In particular, CENTURO represents a challenging platform for planning considering the high number of DoFs. Real experiments were also carried out on CENTAURO using a Lidar to perceive the environment, demonstrating the applicability of the proposed approach to a real scenario.

Currently, the NSPG is able to generate approximately 1000 configurations per second, guaranteeing a good exploration despite using a light randomic approach able to adapt while exploiting the robot's workspace depending on the unfeasibility occurrence. The proposed method, even if based on a random approach, presents a good level of reliability, which is observed on the result obtained in the two considered scenarios. Additionally, it does not require big effort to tune its parameters, which do not depend on the robotic platform in use, as it has been seen by the general applicability of the algorithm to two profoundly different robotic platforms.

Comparing the obtained results to the recent work proposed in [Shigematsu et al., 2019], in the cluttered scenario (Section 2.6.1) we were able to double the configurations with an average time that is smaller of 3 orders of magnitude, guaranteeing minimal differences between adjacent postures. Future works will involve the use of the NSPG in a multi-contact planner scenario similar to the one used to generate stances for the scenario 2.6.1. Further, the stability check could be improved considering centroidal dynamics, allowing higher

dynamic motions and enlarging the set of possible feasible configurations and application, i.e. kinodynamic planning.

References

Bouyarmane, K. and Kheddar, A. (2012). Humanoid robot locomotion and manipulation step planning. *Advanced Robotics*, 26(10):1099–1126.

Buchanan, R., Bandyopadhyay, T., Bjelonic, M., Wellhausen, L., Hutter, M., and Kottege, N. (2019). Walking posture adaptation for legged robot navigation in confined spaces. *IEEE Robotics and Automation Letters (RAL)*, 4(2):2148–2155.

Caron, S., Pham, Q., and Nakamura, Y. (2015). Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5107–5112.

Cognetti, M., Mohammadi, P., and Oriolo, G. (2015). Whole-body motion planning for humanoids based on com movement primitives. In *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, pages 1090–1095.

Deits, R. and Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, pages 279–286.

Escande, A., Kheddar, A., and Miossec, S. (2013). Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5):428–442.

Escande, A., Mansard, N., and Wieber, P.-B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research (IJRR)*, 33(7):1006–1028.

Fang, C., Rocchi, A., Hoffman, E. M., Tsagarakis, N. G., and Caldwell, D. G. (2015). Efficient self-collision avoidance based on focus of interest for humanoid robots. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (HUMANOIDS)*, pages 1060–1066.

Ferrari, P., Cognetti, M., and Oriolo, G. (2018). Anytime whole-body planning/replanning for humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 1–9.

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363.

Ferrolho, H., Merkt, W., Yang, Y., Ivan, V., and Vijayakumar, S. (2018). Whole-body end-pose planning for legged robots on inclined support surfaces in complex environments. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (HUMANOIDS)*, pages 944–951.

Gutmann, J.-S., Fukuchi, M., and Fujita, M. (2005). Real-time path planning for humanoid robot navigation. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1232–1237.

Hauser, K., Bretl, T., and Latombe, J. . (2005). Non-gaited humanoid locomotion planning. In *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, pages 7–12.

Hauser, K., Bretl, T., Latombe, J.-C., Harada, K., and Wilcox, B. (2008). Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research (IJRR)*, 27(11-12):1325–1349.

Hoffman, E. M., Rocchi, A., Laurenzi, A., and Tsagarakis, N. G. (2017). Robot control for dummies: Insights and examples using opensot. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (HUMANOIDS)*, pages 736–741.

Hoffman, E. M. and Tsagarakis, N. G. (2021). The math of tasks: A domain specific language for constraint-based task specification. *International Journal of Humanoid Robotics*, 18(03):2150008.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Springer Autonomous Robots*.

Kanoun, O., Lamiraux, F., and Wieber, P.-B. (2011). Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *IEEE Transaction on Robotics (TRO)*, 27(4):785–792.

Kanoun, O., Lamiraux, F., Wieber, P.-B., Kanehiro, F., Yoshida, E., and Laumond, J.-P. (2009). Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In *IEEE international conference on robotics and automation (ICRA)*, pages 2939–2944.

Kashiri, N., Baccelliere, L., Muratore, L., Laurenzi, A., Ren, Z., Hoffman, E. M., Kamedula, M., Rigano, G. F., Malzahn, J., Cordasco, S., Guria, P., Margan, A., and Tsagarakis, N. G. (2019). Centauro: A hybrid locomotion and high power resilient manipulation platform. *IEEE Robotics and Automation Letters (RAL)*, 4(2):1595–1602.

Kingston, Z., Moll, M., and Kavraki, L. E. (2018). Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):159–185.

Kuffner, J. J., Nishiwaki, K., Kagami, S., Inaba, M., and Inoue, H. (2001). Footstep planning among obstacles for biped robots. In *IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 500–505.

Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R. (2016). Optimization-based locomotion planning, estimation, and control design for atlas. *Autonomous Robots*, 40(3):429–455.

Laurenzi, A., Hoffman, E. M., Muratore, L., and Tsagarakis, N. G. (2019). CartesI/O: A ros based real-time capable cartesian control framework. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 591–596.

Pan, J., Chitta, S., and Manocha, D. (2012). Fcl: A general purpose library for collision and proximity queries. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3859–3866.

Polverini, M. P., Laurenzi, A., Hoffman, E. M., Ruscelli, F., and Tsagarakis, N. G. (2020). Multi-contact heavy object pushing with a centaur-type humanoid robot: Planning and control for a real demonstrator. *IEEE Robotics and Automation Letters (RAL)*, 5(2):859–866.

Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S. (2009). Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 489–494.

Ruscelli, F., Parigi Polverini, M., Laurenzi, A., Mingo Hoffman, E., and Tsagarakis, N. G. (2020). A multi-contact motion planning and control strategy for physical interaction tasks. In *IEEE - RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3869–3876.

Shigematsu, R., Murooka, M., Kakiuchi, Y., Okada, K., and Inaba, M. (2019). Generating a key pose sequence based on kinematics and statics optimization for manipulating a heavy object by a humanoid robot. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3852–3859.

Stasse, O., Escande, A., Mansard, N., Miossec, S., Evrard, P., and Kheddar, A. (2008). Real-time (self)-collision avoidance task on a hrp-2 humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3200–3205.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020). OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672.

Şucan, I. A., Moll, M., and Kavraki, L. E. (2012). The Open Motion Planning Library. *IEEE Robotics & Automation Magazine (RAM)*, 19(4):72–82.

Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., and Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics (TRO)*, 34(3):586–601.

Yang, Y., Ivan, V., Merkt, W., and Vijayakumar, S. (2016). Scaling sampling-based motion planning to humanoid robots. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1448–1454.

Yang, Y., Merkt, W., Ferrolho, H., Ivan, V., and Vijayakumar, S. (2017). Efficient humanoid motion planning on uneven terrain using paired forward-inverse dynamic reachability maps. *IEEE Robotics and Automation Letters (RAL)*, 2(4):2279–2286.

Chapter 3

Global Offline Planning Strategy

Thanks to their structure, humanoid robots have the potential to accomplish complex tasks that require to move 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 these *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. 3.1.

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 [Bouyarmane et al., 2019]. 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 [Atkeson et al., 2018].

3.0.1 Previous works

At the planning level, the MCPC problem requires to compute 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 Bretl [2006], 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: the sequence of contact combinations is in fact acyclic, differently from the case of pure biped locomotion in

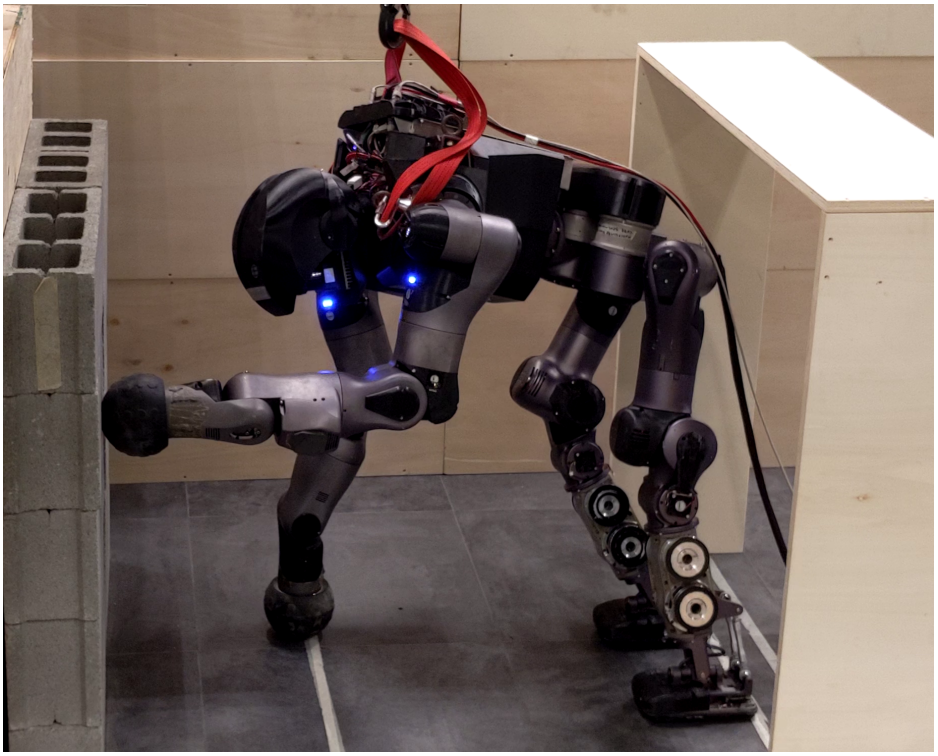


Figure 3.1 An example of multi-contact loco-manipulation task: COMAN+ stands up exploiting the environment, in particular the wall and the ground, as support.

which the identity of the foot in contact with the ground regularly alternates between the right and left. Early approaches, such as Hauser et al. [2008], 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, Bouyarmane and Kheddar [2012]; Escande et al. [2013]) 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 [Tonneau et al., 2018], 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, techniques like contact-consistent elastic strips (CES) [Chung and Khatib, 2015] and contact-invariant optimization (CIO) [Mordatch et al., 2012] 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, while CIO assumes that the number of stances is given and no actuation limits exist.

Except for coupled planners like the latter, 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 online or offline fashion. With online approaches, when moving from the current stance to the next, the robot configuration is regulated to that associated to the latter using a whole-body controller [Bouyarmane and Kheddar, 2011]. Clearly, this approach works only if the consecutive configurations are reasonably close; for more complex motions, the whole-body controller shall track the trajectory of the robot center of mass that is typically produced using Model Predictive Control (see, e.g., Caron and Kheddar [2016]; Carpentier et al. [2016]). On the other hand, offline approaches (e.g., Ruscelli et al. [2020]), involve constrained versions of sampling-based planners (see Kingston et al. [2018] for a recent review) to compute the complete robot motion before execution.

At the control level, the MCPC problem consists of choosing online the inputs for the robot actuators in such a way 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 [Park, 2019], 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 Herzog et al. [2016]; Saab et al. [2013]) simultaneously optimize 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 Henze et al. [2016]; Lee and Goswami [2012]; Stephens and Atkeson [2010]) first solve the optimal contact force distribution problem, and then maps them to joint torques. Such strategy requires the (quite tricky) specification of a reference angular momentum. This is avoided with approaches based on post-optimization (such as Laurenzi et al. [2018]; Polverini et al. [2019]) 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 [Werner et al., 2016], climbing a ladder [Vaillant et al., 2016], or pushing a heavy object [Polverini et al., 2020].

3.1 Background

Before formally defining the problem of interest, this section recalls some basic notions that will be used throughout the chapter.

3.1.1 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[1] coordinates, of the floating-base frame $\mathscr{F}^{\text{fb}}$ w.r.t. the inertial world frame $\mathscr{F}^{\text{w}}$, and $\boldsymbol{q}_{\text{jnt}}$ is the n-vector of joint angles. The dimension of the configuration space $\mathscr{C}$ is $\dim(\mathscr{C}) = 6+n$.

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,

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

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 chapter.

A *stance* is a set $\sigma = \{c_1, \ldots, c_m\}$ of m contacts. Each *contact* is defined by a triplet $c_i = \langle \mathtt{t}_i, \mathscr{F}_i, \boldsymbol{r}_{c,i} \rangle$, 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.

- $\mathscr{F}_i$ 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 $\mathscr{F}_i$ w.r.t. $\mathscr{F}^{\mathrm{W}}$ 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, $\boldsymbol{k}_{p,i}(\cdot)$ and $\boldsymbol{k}_{o,i}(\cdot)$ denote, 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 $\mathscr{F}_i$ w.r.t. $\mathscr{F}^{\mathrm{W}}$ when c_i is established. While maintained, c_i yields a kinematic constraint of the form:

$$\boldsymbol{k}_{p,i}(\boldsymbol{q}) = \boldsymbol{p}_{c,i}, \quad \text{if } \mathtt{t}_i = \mathtt{P}, \tag{3.1}$$

$$\boldsymbol{k}_i(\boldsymbol{q}) = \boldsymbol{r}_{c,i}, \quad \text{if } \mathtt{t}_i = \mathtt{S}. \tag{3.2}$$

In a stance σ, each contact c_i involves a different contact frame. The set $\{\mathscr{F}_1, \ldots, \mathscr{F}_m\}$ of contact frames involved at a stance σ is retrieved by a function $\Psi(\sigma)$.

A stance σ defines a submanifold $\mathscr{C}_\sigma$ of $\mathscr{C}$, called *stance submanifold*, containing all configurations $\boldsymbol{q}$ that satisfy the kinematic constraints (3.1)-(3.2) 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 $\mathscr{C}_\sigma$ is $\dim(\mathscr{C}_\sigma) = \dim(\mathscr{C}) - 3m_{\mathtt{P}} - 6m_{\mathtt{S}}$.

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

- Joint limits are respected, i.e.,

$$\boldsymbol{q}_{\mathrm{jnt}}^{\min} \leq \boldsymbol{q}_{\mathrm{jnt}} \leq \boldsymbol{q}_{\mathrm{jnt}}^{\max}. \tag{3.3}$$

- 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 Sect. 3.1.2).

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.
- $\mathscr{D}_\sigma \cap \mathscr{D}_{\sigma'} \neq \emptyset$, i.e., there exists (at least) one configuration $\boldsymbol{q}$, called *transition*, that belongs to both $\mathscr{D}_\sigma$ and $\mathscr{D}_{\sigma'}$. 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.

3.1.2 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 $\mathscr{F}^{\mathrm{w}}$) 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 $\mathscr{F}_i$. Clearly, $\boldsymbol{\tau}_{c,i} = \boldsymbol{0}$ 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 $\mathscr{F}_i$ at a configuration $\boldsymbol{q}$, i.e.,

$$\boldsymbol{w}_{c,i}^i = \begin{bmatrix} \boldsymbol{R}_i^T & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{R}_i^T \end{bmatrix} \boldsymbol{w}_{c,i}, \tag{3.4}$$

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

Collect in vector $\boldsymbol{W}_c = [\boldsymbol{w}_{c,1}^T, \ldots, \boldsymbol{w}_{c,m}^T]^T$ 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

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}_c^T(\boldsymbol{q})\boldsymbol{W}_c, \tag{3.5}$$

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

$$\boldsymbol{S} = [\boldsymbol{0}_{n\times6} \quad \boldsymbol{I}_{n\times n}]^T \tag{3.6}$$

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

The humanoid at a configuration $\boldsymbol{q} \in \mathscr{C}_\sigma$ 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

$$\boldsymbol{g}_u(\boldsymbol{q}) = \boldsymbol{J}_{c,u}^T(\boldsymbol{q})\boldsymbol{W}_c, \tag{3.7}$$
$$\boldsymbol{g}_a(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{J}_{c,a}^T(\boldsymbol{q})\boldsymbol{W}_c, \tag{3.8}$$

which are obtained by considering (3.5) under quasi-static conditions, i.e., $\ddot{\boldsymbol{q}} = \dot{\boldsymbol{q}} = \boldsymbol{0}$, and explicitly separating the system unactuated (subscript u) and actuated (subscript a) parts associated to, respectively, the first 6 and last n rows of (3.5). Note that, condition (3.8) implicitly requires that the actuated torques $\boldsymbol{\tau}$ are within their limits, i.e.,

$$\boldsymbol{\tau}^{\min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}^{\max}. \tag{3.9}$$

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.,

$$\begin{aligned} &\boldsymbol{f}_{c,i} \cdot \boldsymbol{n}_{c,i} > 0, \\ &\|\boldsymbol{f}_{c,i}^t\| \leq \mu_i(\boldsymbol{f}_{c,i} \cdot \boldsymbol{n}_{c,i}), \end{aligned} \tag{3.10}$$

which imposes unilaterality and non-slippage of c_i, with $\boldsymbol{f}_{c,i}^t = \boldsymbol{f}_{c,i} - (\boldsymbol{n}_{c,i} \cdot \boldsymbol{f}_{c,i}) \cdot \boldsymbol{n}_{c,i}$ 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 Bouyarmane et al. [2019]; Trinkle et al. [1997]), condition (3.10) takes the linear form

$$f_{c,i}^{z,i} > 0, \quad \left|f_{c,i}^{x,i}\right| \leq \tilde{\mu}_i f_{c,i}^{z,i}, \quad \left|f_{c,i}^{y,i}\right| \leq \tilde{\mu}_i f_{c,i}^{z,i}, \tag{3.11}$$

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.,

$$\left|x_{\mathrm{CoP},i}^i\right| \leq d_i^x, \quad \left|y_{\mathrm{CoP},i}^i\right| \leq d_i^y, \tag{3.12}$$

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

$$x^i_{\text{CoP},i} = -\frac{\tau^{y,i}_{c,i}}{f^{z,i}_{c,i}}, \quad y^i_{\text{CoP},i} = \frac{\tau^{x,i}_{c,i}}{f^{z,i}_{c,i}}, \tag{3.13}$$

and d^x_i, d^y_i are the half-dimensions of the CoP rectangular[2], admissible region.

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

$$\tau^{z,\min}_{c,i} \leq \tau^{z,i}_{c,i} \leq \tau^{z,\max}_{c,i}, \tag{3.14}$$

with

$$\begin{aligned} \tau^{z,\min}_{c,i} = -\tilde{\mu}_i(d^x_i + d^y_i) f^{z,i}_{c,i} + \left| d^y_i f^{x,i}_{c,i} - \tilde{\mu}_i \tau^{x,i}_{c,i} \right| \\ + \left| d^x_i f^{y,i}_{c,i} - \tilde{\mu}_i \tau^{y,i}_{c,i} \right|, \end{aligned} \tag{3.15}$$

$$\begin{aligned} \tau^{z,\max}_{c,i} = \tilde{\mu}_i(d^x_i + d^y_i) f^{z,i}_{c,i} - \left| d^y_i f^{x,i}_{c,i} + \tilde{\mu}_i \tau^{x,i}_{c,i} \right| \\ - \left| d^x_i f^{y,i}_{c,i} + \tilde{\mu}_i \tau^{y,i}_{c,i} \right|. \end{aligned} \tag{3.16}$$

C1 is usually referred to as *centroidal statics condition*, while C2-C4 are known as *contact-stability conditions* (for details about their derivation see Caron et al. [2015]).

3.2 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 the proposed 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 $\boldsymbol{q}^{\text{ini}}$ and contact wrenches $\boldsymbol{W}^{\text{ini}}_c$.

The objective of this chapter 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 challenges that will be solved by the MCPC framework, adopting the stance-before-motion paradigm, in the following order:

[2]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.

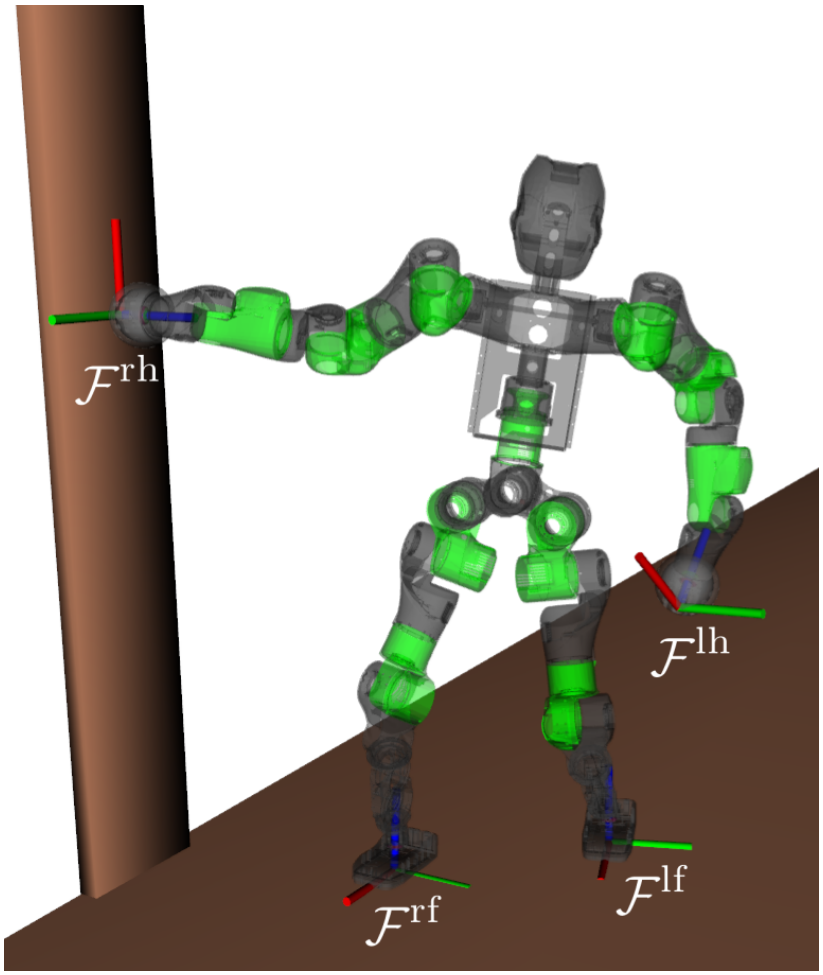


Figure 3.2 The predefined set U of potential contact frames. In this example, $U = \{\mathscr{F}^{\mathrm{lf}}, \mathscr{F}^{\mathrm{rf}}, \mathscr{F}^{\mathrm{lh}}, \mathscr{F}^{\mathrm{rh}}\}$ contains the frames rigidly attached on the left/right foot/hand. Feet and hands can, respectively, establish surface and point contacts, i.e., $\varphi(\mathscr{F}^{\mathrm{lf}}) = \varphi(\mathscr{F}^{\mathrm{rf}}) = \mathtt{S}$ and $\varphi(\mathscr{F}^{\mathrm{lh}}) = \varphi(\mathscr{F}^{\mathrm{rh}}) = \mathtt{P}$. The robot is at a stance σ such that $\Psi(\sigma) = \{\mathscr{F}_1 = \mathscr{F}^{\mathrm{lf}}, \mathscr{F}_2 = \mathscr{F}^{\mathrm{rf}}, \mathscr{F}_3 = \mathscr{F}^{\mathrm{rh}}\}$.

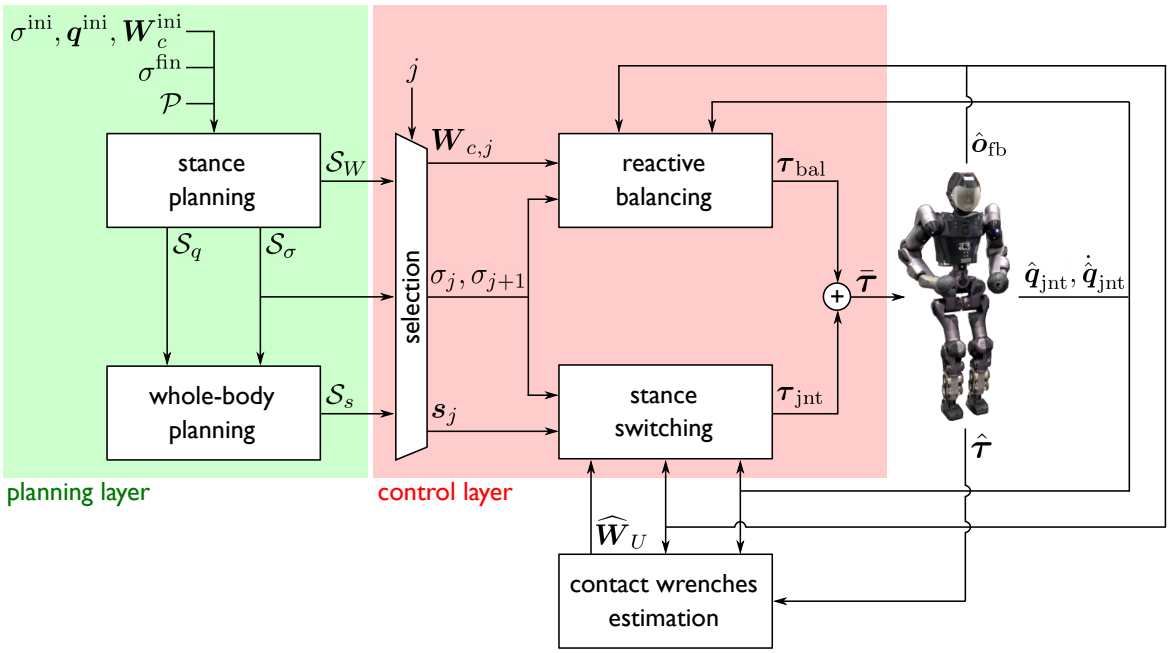


Figure 3.3 Block scheme of the proposed MCPC framework. The single modules are described in Sects. 3.4-3.6.

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. 3.4);

3. to generate torque commands for the humanoid in order to realize the planned motions, while guaranteeing closed-loop balance at any time instant.

Under the following assumptions:

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

A2 Contacts can be established anywhere in the environment by using a predefined set $U = \{\mathscr{F}^{\text{a}}, \mathscr{F}^{\text{b}}, \mathscr{F}^{\text{c}}, \dots\}$ of potential contact frames, henceforth referred to as *end-effectors*. Each $\mathscr{F}_h \in U$ is rigidly attached to a point/surface on the exterior 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. 3.2). The

function $\varphi(\mathscr{F}_h)$ retrieves the type of contact that an end-effector $\mathscr{F}_h \in U$ can establish, with $\varphi(\mathscr{F}_h) = \mathtt{P}$ or $\varphi(\mathscr{F}_h) = \mathtt{S}$ depending on whether $\mathscr{F}_h$ can establish point or surface contacts, respectively.

3.3 Proposed framework

To solve the described problem, this chapter proposes the MCPC framework whose underlying architecture is shown in Fig. 3.3. 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*. Figure 3.4 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

$$\begin{aligned}\mathscr{S}_\sigma &= \{\sigma_0, \ldots, \sigma_N\},\\ \mathscr{S}_q &= \{\boldsymbol{q}_0, \ldots, \boldsymbol{q}_N\},\\ \mathscr{S}_W &= \{\boldsymbol{W}_{c,0}, \ldots, \boldsymbol{W}_{c,N}\},\end{aligned}$$

where $\mathscr{S}_\sigma$ is the sequence of $N+1$ stances leading to the desired final stance σ^{fin}, i.e., $\sigma_N = \sigma^{\text{fin}}$, while $\mathscr{S}_q$ and $\mathscr{S}_W$ are sequences of transitions and contact wrenches, respectively. More precisely, for each $j = 1, \ldots, N$, stance $\sigma_j \in \mathscr{S}_\sigma$ is adjacent to σ_{j-1}; configuration $\boldsymbol{q}_j \in \mathscr{S}_q$ is a transition associated to stance σ_j, i.e., $\boldsymbol{q}_j \in \mathscr{D}_{\sigma,j-1} \cap \mathscr{D}_{\sigma,j}$, with $\mathscr{D}_{\sigma,j-1}$ and $\mathscr{D}_{\sigma,j}$ the feasible subspace at σ_{j-1} and σ_j, respectively; vector $\boldsymbol{W}_{c,j} \in \mathscr{S}_W$ 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 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 $\boldsymbol{q}_{j+1}$ ($j = 0, \ldots, N-1$) belonging to $\mathscr{S}_q$, computes a feasible whole-body motion $\boldsymbol{s}_j$, i.e., a configuration space trajectory in $\mathscr{D}_{\sigma,j}$ 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

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

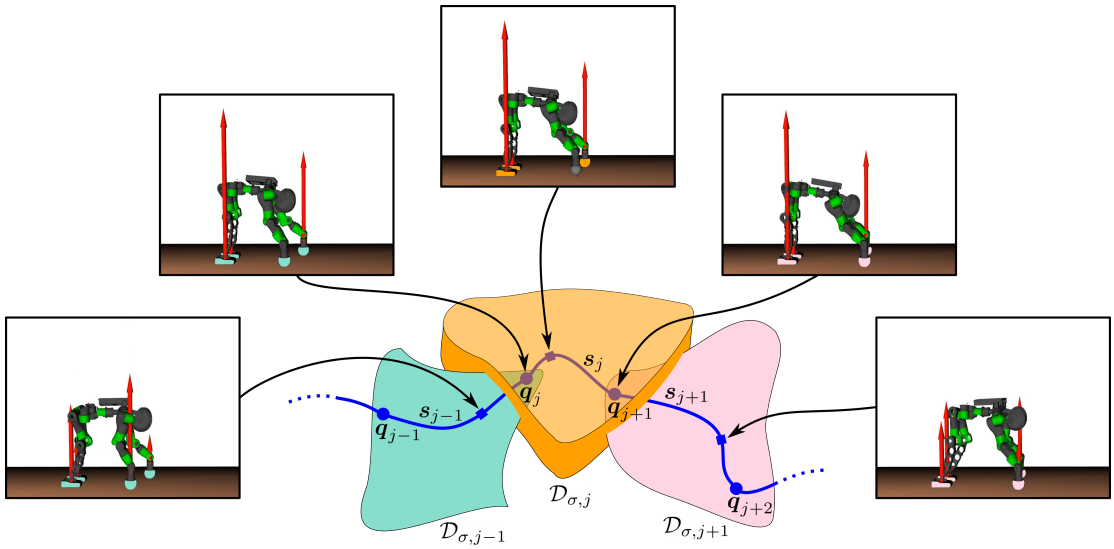


Figure 3.4 Role of the two planning modules. The stance planner generates the sequences of adjacent stances, associated transitions, and contact wrenches. Here, the feasible subspaces associated with three consecutive stances are depicted in cyan, orange, and pink. Transitions (blue bullets) belong to 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).

The control layer is responsible for generating online the torque commands $\bar{\boldsymbol{\tau}}$ for the humanoid actuators to execute the planned sequence $\mathscr{S}_s$ of whole-body motions throughout the planned sequence $\mathscr{S}_\sigma$ 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 $\mathscr{S}_\sigma$. 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

$$\widehat{\boldsymbol{W}}_U = (\boldsymbol{J}_{U,a}^T(\hat{\boldsymbol{q}}))^\dagger(\boldsymbol{g}_a(\hat{\boldsymbol{q}}) - \hat{\boldsymbol{\tau}}), \tag{3.17}$$

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 $\hat{\boldsymbol{q}}$[3]. 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.

The next sections will discuss in detail the mentioned modules constituting the proposed MCPC framework.

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

3.4 Stance planning

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

To compute these sequences, the stance planner receives as 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 $\mathscr{P}$ and the desired final stance σ^{fin}.

3.4.1 Tree construction

The proposed stance planner, whose pseudocode is given in Algorithm 2, uses an RRT-like strategy to iteratively construct a tree $\mathscr{T}$ 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 $\boldsymbol{q} \in \mathscr{D}_\sigma$, 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 Sect. 3.1.1); consequently, $\boldsymbol{q}'$ specified in v' is a transition belonging to $\mathscr{D}_\sigma \cap \mathscr{D}_{\sigma'}$. At the beginning, the tree $\mathscr{T}$ is rooted at the vertex $v^{\text{ini}} = \langle \sigma^{\text{ini}}, \boldsymbol{q}^{\text{ini}}, \boldsymbol{W}_c^{\text{ini}} \rangle$ (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 $\mathscr{F}^{\text{rand}}$ 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 towards unexplored regions of the search space and the goal, respectively. In the first case, $\mathscr{F}^{\text{rand}}$ and $\boldsymbol{p}^{\text{rand}}$ are randomly picked from the set U and the workspace, respectively. In the second case, a contact $c^{\text{fin}} = \langle \mathtt{t}^{\text{fin}}, \mathscr{F}^{\text{fin}}, \boldsymbol{r}_c^{\text{fin}} \rangle$ is randomly picked among those in σ^{fin}; then, $\mathscr{F}^{\text{rand}}$ and $\boldsymbol{p}^{\text{rand}}$ are chosen as $\mathscr{F}^{\text{fin}}$ and $\boldsymbol{p}_c^{\text{fin}}$, respectively.

Then, the planner assigns to each vertex v in $\mathscr{T}$ 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 $\mathscr{F}^{\text{rand}}$ at $\boldsymbol{q}$ (the configuration specified by v) and $\boldsymbol{p}^{\text{rand}}$. The resulting probability distribution is used to randomly choose a vertex $v^{\text{near}} = \langle \sigma^{\text{near}}, \boldsymbol{q}^{\text{near}}, \boldsymbol{W}_c^{\text{near}} \rangle$ of $\mathscr{T}$ 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 $\mathscr{T}$ from v^{near} by removing or adding a contact. To this end, it checks if σ^{near} contains a contact involving $\mathscr{F}^{\text{rand}}$, i.e., $\mathscr{F}^{\text{rand}} \in \Psi(\sigma^{\text{near}})$. Based on the outcome of this check, a candidate stance σ^{cand} is generated (and subsequently validated) as follows.

- If $\mathscr{F}^{\text{rand}} \in \Psi(\sigma^{\text{near}})$, σ^{cand} is generated by removing from σ^{near} the contact $c^{\text{rem}} = \langle \mathtt{t}^{\text{rem}}, \mathscr{F}^{\text{rem}}, \boldsymbol{r}_c^{\text{rem}} \rangle$ involving the selected end-effector $\mathscr{F}^{\text{rand}}$, i.e., $\sigma^{\text{cand}} = \sigma^{\text{near}} \setminus c^{\text{rem}}$. Contact c^{rem} is simply that in σ^{near} such that $\mathscr{F}^{\text{rem}} = \mathscr{F}^{\text{rand}}$.

- If $\mathscr{F}^{\text{rand}} \notin \Psi(\sigma^{\text{near}})$, σ^{cand} is generated by adding to σ^{near} a novel contact $c^{\text{add}} = \langle \mathtt{t}^{\text{add}}, \mathscr{F}^{\text{add}}, \boldsymbol{r}_c^{\text{add}} \rangle$ involving the selected end-effector $\mathscr{F}^{\text{rand}}$, i.e., $\sigma^{\text{cand}} = \sigma^{\text{near}} \cup c^{\text{add}}$. 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 $\mathscr{F}^{\text{rand}}$. Such reachable workspace $\mathscr{W}$ is approximated as the set of points of the point cloud $\mathscr{P}$ that lie inside a sphere[4] 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 $\mathscr{W}$ is not empty [5]. Bounds r^{min} and r^{max} are predefined for each end-effector according to the kinematic limits of the robot. Once the workspace $\mathscr{W}$ has been computed, the point $\boldsymbol{p}_c^{\text{add}} \in \mathscr{W}$ that is the closest to $\boldsymbol{p}^{\text{rand}}$ is selected, and c^{add} is built by setting $\mathtt{t}^{\text{add}} = \varphi(\mathscr{F}^{\text{rand}})$, $\mathscr{F}^{\text{add}} = \mathscr{F}^{\text{rand}}$, and $\boldsymbol{r}_c^{\text{add}} = [(\boldsymbol{p}_c^{\text{add}})^T, \emptyset]^T$, with the orientation $\boldsymbol{o}_c^{\text{add}}$ 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 Sect. 3.4.2 is invoked with the aim of producing a new stance σ^{new} and a transition $\boldsymbol{q}^{\text{new}} \in \mathscr{D}_\sigma^{\text{near}} \cap \mathscr{D}_\sigma^{\text{new}}$, with $\mathscr{D}_\sigma^{\text{near}}$ and $\mathscr{D}_\sigma^{\text{new}}$ 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 Sect. 3.1.1) are satisfied. In fact, σ^{new} differs from σ^{near} by a single contact (by construction of σ^{cand}), and $\mathscr{D}_\sigma^{\text{near}} \cap \mathscr{D}_\sigma^{\text{new}} \neq \emptyset$ thanks to the existence of $\boldsymbol{q}^{\text{new}}$. The procedure described in Sect. 3.4.3 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 $v^{\text{new}} = \langle \sigma^{\text{new}}, \boldsymbol{q}^{\text{new}}, \boldsymbol{W}_c^{\text{new}} \rangle$ is then created and added in tree $\mathscr{T}$ as a child of v^{near}. ◀

[4] More sophisticated methods for approximating the reachable workspace (e.g., Guan and Yokoi [2006]; Jamone et al. [2012]) can be involved without affecting the overall planning strategy.

[5] 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 2.

Algorithm 2: Stance Planner

1 $v^{\text{ini}} \leftarrow \langle \sigma^{\text{ini}}, \boldsymbol{q}^{\text{ini}}, \boldsymbol{W}_c^{\text{ini}} \rangle$;
2 AddVertex($\mathscr{T}$, v^{ini}, $\emptyset$);
3 $l \leftarrow 0$;
4 **repeat**
5 $[\mathscr{F}^{\text{rand}}, \boldsymbol{p}^{\text{rand}}] \leftarrow$ PickRandom();
6 $v^{\text{near}} \leftarrow$ FindNearestVertex($\mathscr{T}$, $\mathscr{F}^{\text{rand}}$, $\boldsymbol{p}^{\text{rand}}$);
7 **if** $\mathscr{F}^{\text{rand}} \in \Psi(\sigma^{\text{near}})$ **then**
8 $c^{\text{rem}} \leftarrow$ ExtractContact(σ^{near}, $\mathscr{F}^{\text{rand}}$);
9 $\sigma^{\text{cand}} \leftarrow \sigma^{\text{near}} \setminus c^{\text{rem}}$;
10 **else**
11 $\mathscr{W} \leftarrow$ ComputeReachableWorkspace($\boldsymbol{q}^{\text{near}}$, $\mathscr{F}^{\text{rand}}$);
12 $\boldsymbol{p}_c^{\text{add}} \leftarrow$ PickRandomPoint($\mathscr{W}$, $\boldsymbol{p}^{\text{rand}}$);
13 $c^{\text{add}} \leftarrow \langle \varphi(\mathscr{F}^{\text{rand}}), \mathscr{F}^{\text{rand}}, [(\boldsymbol{p}_c^{\text{add}})^T, \emptyset]^T \rangle$;
14 $\sigma^{\text{cand}} \leftarrow \sigma^{\text{near}} \cup c^{\text{add}}$;
15 **end**
16 $[\sigma^{\text{new}}, \boldsymbol{q}^{\text{new}}] \leftarrow$ GenerateTransition(σ^{near}, σ^{cand}, $\boldsymbol{q}^{\text{near}}$);
17 **if** $\boldsymbol{q}^{\text{new}} \neq \emptyset$ **then**
18 $\boldsymbol{W}_c^{\text{new}} \leftarrow$ ComputeContactWrenches(σ^{new}, $\boldsymbol{q}^{\text{new}}$);
19 $v^{\text{new}} \leftarrow \langle \sigma^{\text{new}}, \boldsymbol{q}^{\text{new}}, \boldsymbol{W}_c^{\text{new}} \rangle$;
20 AddVertex($\mathscr{T}$, v^{new}, v^{near});
21 **end**
22 $l \leftarrow l + 1$;
23 **until** $\sigma^{\text{new}} = \sigma^{\text{fin}}$ **or** $l = l_\sigma^{\max}$;
24 **if** $\sigma^{\text{new}} = \sigma^{\text{fin}}$ **then**
25 $[\mathscr{S}_\sigma, \mathscr{S}_q, \mathscr{S}_W] \leftarrow$ RetrieveSolution($\mathscr{T}$);
26 **return** $[\mathscr{S}_\sigma, \mathscr{S}_q, \mathscr{S}_W]$;
27 **end**
28 **return** $[\emptyset, \emptyset, \emptyset]$;

Construction of $\mathscr{T}$ 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 $\mathscr{S}_\sigma$, $\mathscr{S}_q$, and $\mathscr{S}_W$ are directly retrieved from the vertices along the branch of $\mathscr{T}$ 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.

3.4.2 Transition generation

The proposed transition generator consists of an adaptation of the method presented in Rossini et al. [2021] 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 $\boldsymbol{q}^{\text{near}}$,

Procedure 3: GenerateTransition(σ^{near}, σ^{cand}, $\boldsymbol{q}^{\text{near}}$)

```
1  if |σ^cand| > |σ^near| then
2  |   σ^lar ← σ^cand, σ^sma ← σ^near;
3  else
4  |   σ^lar ← σ^near, σ^sma ← σ^cand;
5  end
6  q^nom ← SolveIK(σ^lar, q^near, q^near);
7  q^cand ← q^nom;
8  l ← 0;
9  while not (CollisionFree(q^cand) and Balanced(q^cand, σ^sma)) do
10 |   if mod(l, l^max_tran) = 0 then
11 |   |   q^ref ← q^nom;
12 |   |   q̇^rand ← GenerateRandomVelocity();
13 |   end
14 |   q^ref ← Integrate(q^ref, q̇^rand);
15 |   q^cand ← SolveIK(σ^lar, q^cand, q^ref);
16 |   l ← l + 1;
17 end
18 σ^cand ← UpdateStance(σ^cand, q^cand);
19 return [σ^cand, q^cand];
```

and the candidate stance σ^{cand}. The aim is to produce a configuration $\boldsymbol{q}^{\text{cand}}$ that belongs to $\mathscr{D}_{\sigma}^{\text{near}} \cap \mathscr{D}_{\sigma}^{\text{cand}}$, 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 $\mathscr{C}_{\sigma}^{\text{lar}}$ and $\mathscr{C}_{\sigma}^{\text{sma}}$ their associated stance submanifolds, and by $\mathscr{D}_{\sigma}^{\text{lar}}$ and $\mathscr{D}_{\sigma}^{\text{sma}}$ their corresponding feasible subspaces. Recall that, a configuration $\boldsymbol{q}$ belongs to $\mathscr{D}_{\sigma}^{\text{lar}} \cap \mathscr{D}_{\sigma}^{\text{sma}}$ if it (*i*) satisfies the $\left|\sigma^{\text{lar}}\right|$ 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.

The transition generator, whose pseudo-code is given in Procedure 3, works in an iterative fashion by repeatedly invoking an Inverse Kinematics (IK) solver. In the following, first the overall transition generation procedure is presented, then, the details of the IK solver will be discussed.

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 $\mathscr{C}_{\sigma}^{\text{lar}}$ 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 Sect. 3.4.3 is invoked. It verifies if there exists a vector $\boldsymbol{W}_c^{\text{cand}}$, 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 $l_{\text{tran}}^{\max}$ 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 $l_{\text{tran}}^{\max}$ iterations, while $\boldsymbol{q}^{\text{ref}}$ will consequently change in the neighbourhood 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 the 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 $\dot{\boldsymbol{q}}^{\text{rand}}$ refer to Rossini et al. [2021].

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 $\mathscr{F}_i$ 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 $\mathscr{D}_{\sigma}^{\text{near}} \cap \mathscr{D}_{\sigma}^{\text{new}}$, 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 3, a failure is reported to the stance planner.

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 $\mathscr{C}_{\sigma}$ 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

$$\min_{\dot{\boldsymbol{q}}} \|\boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} - \boldsymbol{K}_\sigma \boldsymbol{e}_\sigma(\boldsymbol{q})\|^2 + \alpha \|\dot{\boldsymbol{q}}\|^2 \tag{3.18a}$$

s.t.

$$\text{joint limits constraint} \tag{3.18b}$$

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, $\boldsymbol{J}_\sigma(\boldsymbol{q}) = [\boldsymbol{J}^T_{\sigma,1}(\boldsymbol{q}), \ldots, \boldsymbol{J}^T_{\sigma,m}(\boldsymbol{q})]^T$ and $\boldsymbol{e}_\sigma(\boldsymbol{q}) = [\boldsymbol{e}^T_{\sigma,1}(\boldsymbol{q}), \ldots, \boldsymbol{e}^T_{\sigma,m}(\boldsymbol{q})]^T$, where $m = |\sigma|$ and $\boldsymbol{J}_{\sigma,i}(\boldsymbol{q})$, $\boldsymbol{e}_{\sigma,i}(\boldsymbol{q})$ 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, Sect. 3.3), for a configuration $\boldsymbol{q} \in \mathscr{C}_\sigma$ to be collision-free it is necessary (but not sufficient) that the z-axis of each contact frame $\mathscr{F}_i$ 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, the cost function of the first QP has been augmented of 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, the formulation of such a term has been omitted, whose structure is similar to that of the first.

The second QP problem is

$$\min_{\dot{\boldsymbol{q}}} \left\| \dot{\boldsymbol{q}} - \boldsymbol{K}_q \boldsymbol{e}_q(\boldsymbol{q}) \right\|^2 \tag{3.19a}$$

s.t.

$$\boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{J}_\sigma(\boldsymbol{q})\dot{\boldsymbol{q}}_1^* \tag{3.19b}$$

$$\text{joint limits constraint} \tag{3.19c}$$

and attempts to bring the current configuration $\boldsymbol{q}$ towards 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 (3.3) 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 $l_{\text{IK}}^{\max}$ 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.

3.4.3 Contact wrenches computation

Given a generic stance σ and a configuration $\boldsymbol{q} \in \mathscr{C}_\sigma$, 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 Sect. 3.1.2. $\boldsymbol{W}_c$ is computed solving the following QP problem

$$\min_{\boldsymbol{W}_c} \left\| \boldsymbol{g}_u(\boldsymbol{q}) - \boldsymbol{J}_{c,u}^T(\boldsymbol{q})\boldsymbol{W}_c \right\|^2 + \beta \|\boldsymbol{W}_c\|^2 \tag{3.20a}$$

s.t.

$$\text{torque limits constraint (3.21)} \tag{3.20b}$$

$$\text{friction cone constraints (3.11)} \tag{3.20c}$$

$$\text{CoP constraints (3.12)} \tag{3.20d}$$

$$\text{yaw moment constraints (3.14)} \tag{3.20e}$$

Here, the four kinds of constraints emerge from the four conditions C1-C4, in the same order.

In particular, the system accounts 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 (3.7)), 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

$$\boldsymbol{\tau}^{\min} \leq \boldsymbol{g}_a(\boldsymbol{q}) - \boldsymbol{J}_{c,a}^T(\boldsymbol{q})\boldsymbol{W}_c \leq \boldsymbol{\tau}^{\max} \tag{3.21}$$

that is obtained by rewriting (3.9) 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 (i.e., the constraint is not enforced for point contacts).

Recall that the computation of contact wrenches is required within the stance planner both to check the static balance at a candidate transition $\boldsymbol{q}^{\text{cand}}$ (line 9 of Procedure 3), 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 2). 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 not the case, 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. Only in this case, static balance at $\boldsymbol{q}^{\text{cand}}$ is considered satisfied.

A remark is here about the choice of formulating the centroidal statics condition 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.

3.5 Whole-body planning

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

The pseudocode of the whole-body planner is given in Algorithm 4. For each pair of consecutive configurations $\boldsymbol{q}_j$ and $\boldsymbol{q}_{j+1}$ ($j = 0,\dots,N-1$) in $\mathscr{S}_q$, a configuration space trajectory that connects the two is planned. To this end, the two-stage approach presented in Ruscelli et al. [2020] is adopted. First, a path consisting of a sequence $\mathscr{S}_{q,j}$ of via points (configurations) in $\mathscr{D}_{\sigma,j}$ joining $\boldsymbol{q}_j$ to $\boldsymbol{q}_{j+1}$ is found (line 3); then, a continuous (with continuous first time derivative) trajectory $\boldsymbol{s}_j(t)$, $t \in [t_j, t_j + \delta_j)$, interpolating the configurations in $\mathscr{S}_{q,j}$ 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 $\mathscr{S}_s$. These, together with the sequences $\mathscr{S}_W$ and $\mathscr{S}_\sigma$, produced by the stance planner, are finally sent to the control layer.

3.5.1 Via points computation

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

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

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 $\mathscr{D}_{\sigma,j}$, is computed using the AtlasRRT* method [Jaillet and Porta, 2012].

While constructing a tree of configurations belonging to $\mathscr{D}_{\sigma,j}$ using an RRT*-based strategy, the algorithm incrementally builds an atlas of the stance submanifold $\mathscr{C}_{\sigma,j}$ defined by σ_j, i.e., a collection of charts, each one represented by a tangent space that locally approximates $\mathscr{C}_{\sigma,j}$ 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 $\mathscr{C}_{\sigma,j}$ using its atlas.

Every time a new configuration is generated, the algorithm verifies that it belongs to the feasible subspace $\mathscr{D}_{\sigma,j}$ 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 is performed using the method discussed in Sect. 3.4.3, 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 $\mathscr{S}_{q,j}$ is retrieved from the constructed tree and passed to the trajectory optimization procedure.

Algorithm 4: Whole-Body Planner

$\mathscr{S}_s \leftarrow \{\}$;
for $j \leftarrow 0$ **to** $N-1$ **do**
  $\mathscr{S}_{q,j} \leftarrow$ ComputeViaPoints($\boldsymbol{q}_j, \boldsymbol{q}_{j+1}, \sigma_j$);
  $\boldsymbol{s}_j(t), t \in [t_j, t_j+\delta_j) \leftarrow$ OptimizeTrajectory($\mathscr{S}_{q,j}$);
  $\mathscr{S}_s \leftarrow \mathscr{S}_s \cup \{\boldsymbol{s}_j\}$;
end
return $\mathscr{S}_s$;

3.5.2 Trajectory optimization

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

$$\boldsymbol{s}_j(t) = \boldsymbol{\rho}_{j,k}(t), \quad t \in [t_{j,k}, t_{j,k+1} = t_{j,k} + \delta_{j,k}), \tag{3.22}$$

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

$$\boldsymbol{\rho}_{j,k}(t) = \sum_{l=0}^{3} \boldsymbol{\lambda}_{j,k,l} (t - t_{j,k})^l, \tag{3.23}$$

with $\boldsymbol{\lambda}_{j,k,l}$ ($l = 0, \ldots, 3$) its coefficients. Moreover, according to (3.22), the overall duration of trajectory $\boldsymbol{s}_j(t)$ will be

$$\delta_j = \sum_{k=0}^{M_j-1} \delta_{j,k}. \tag{3.24}$$

Collect in vectors

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

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

$$\min_{\boldsymbol{\zeta}_j, \boldsymbol{\xi}_j} \left\| \left\| \boldsymbol{\zeta}_j \right\| \right\|^2 + \gamma \| \boldsymbol{\xi}_j \|^2 \tag{3.25a}$$

s.t.

$$\boldsymbol{\zeta}_j \geq \mathbf{0}, \tag{3.25b}$$

$$\boldsymbol{\rho}_{j,k}(t_{j,k}) = \boldsymbol{q}_{j,k}, k = 0, \ldots, M_j - 1, \tag{3.25c}$$

$$\boldsymbol{\rho}_{j,k}(t_{j,k+1}) = \boldsymbol{q}_{j,k+1}, k = 0, \ldots, M_j - 1, \tag{3.25d}$$

$$\dot{\boldsymbol{\rho}}_{j,k}(t_{j,k+1}) = \dot{\boldsymbol{\rho}}_{j,k+1}(t_{j,k+1}), k = 0, \ldots, M_j - 2, \tag{3.25e}$$

$$\dot{\boldsymbol{\rho}}_{j,0}(t_{j,0}) = \mathbf{0}, \tag{3.25f}$$

$$\dot{\boldsymbol{\rho}}_{j,M_j-1}(t_{j,M_j}) = \mathbf{0}, \tag{3.25g}$$

$$\dot{\boldsymbol{q}}_{\text{jnt}}^{\min} \leq \boldsymbol{S}^T \dot{\boldsymbol{\rho}}_{j,k}(t_{j,k}) \leq \dot{\boldsymbol{q}}_{\text{jnt}}^{\max}, k = 1, \ldots, M_j - 1, \tag{3.25h}$$

$$\ddot{\boldsymbol{q}}_{\text{jnt}}^{\min} \leq \boldsymbol{S}^T \ddot{\boldsymbol{\rho}}_{j,k}(t_{j,k}) \leq \ddot{\boldsymbol{q}}_{\text{jnt}}^{\max}, k = 1, \ldots, M_j - 1. \tag{3.25i}$$

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 (3.25b), passage through the via points (3.25c-3.25d), continuity of the velocity at the internal via points (3.25e), null velocity at the initial and final points (3.25f-3.25g), limits on the resulting joint velocities and accelerations (3.25h-3.25i), with matrix $\boldsymbol{S}$ defined in (3.6).

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 $\boldsymbol{\rho}_{j,k}(t)$ is completely contained in $\mathscr{D}_{\sigma,j}$ 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 $\mathscr{S}_{q,j}$ will be close to each other.

3.6 Control layer

The control layer adopted in the presented framework is mainly based on Laurenzi et al. [2018]; Polverini et al. [2020]; Ruscelli et al. [2020]. The next section discusses it as a possible option for executing the references produced through the offline planning phase. However, other implementations, e.g. based on MPC or Inverse Dynamics, of the control layer – which is not the main focus of this chapter – can potentially be involved.

As anticipated in Sect. 3.3, a naive feedforward execution of the planned motions $\mathscr{S}_s$ 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:

$$\bar{\boldsymbol{\tau}} = \boldsymbol{\tau}_{\text{bal}} + \boldsymbol{\tau}_{\text{jnt}}. \tag{3.26}$$

The first term $\boldsymbol{\tau}_{\text{bal}}$ in (3.26) 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

$$\min_{\boldsymbol{W}_c} \left\| \boldsymbol{W}_c - \boldsymbol{W}_{c,j} \right\|^2 \tag{3.27a}$$

s.t.

centroidal statics constraint (3.7) (3.27b)

torque limits constraint (3.21) (3.27c)

friction cone constraints (3.11) (3.27d)

CoP constraints (3.12) (3.27e)

yaw moment constraints (3.14) (3.27f)

Such QP is obtained by minor modifications of that used to compute the contact wrenches during the planning phase (see Sect. 3.4.3). 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:

$$\boldsymbol{\tau}_{\text{bal}} = \boldsymbol{g}_a(\hat{\boldsymbol{q}}) - \boldsymbol{J}_{c,a}^T(\hat{\boldsymbol{q}})\overline{\boldsymbol{W}}_c, \tag{3.28}$$

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 (3.28), the term $\hat{\boldsymbol{q}} = [\boldsymbol{0}_{3\times1}^T \ \hat{\boldsymbol{o}}_{\text{fb}}^T \ \boldsymbol{q}_{\text{jnt}}^T]^T$ 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 (3.26) consists instead in a feedback term for tracking the planned motion $\boldsymbol{s}_j$ and is produced by the stance switching module as

$$\boldsymbol{\tau}_{\text{jnt}} = \boldsymbol{K}_P(\bar{\boldsymbol{q}}_{\text{jnt}} - \hat{\boldsymbol{q}}_{\text{jnt}}) + \boldsymbol{K}_D(\dot{\bar{\boldsymbol{q}}}_{\text{jnt}} - \dot{\hat{\boldsymbol{q}}}_{\text{jnt}}), \tag{3.29}$$

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 $c^{\text{diff}} = \langle \mathfrak{t}^{\text{diff}}, \mathscr{F}^{\text{diff}}, \boldsymbol{r}_c^{\text{diff}} \rangle$, 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 (3.29) are generated via kinematic control using a scheme similar to that discussed in Sect. (19) with the idea of imposing a linear velocity $\dot{\boldsymbol{p}}^{\text{ref}}$ to end-effector $\mathscr{F}^{\text{diff}}$ to make it advance/retract towards/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.

3.7 Implementation details

The 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 (Sects. 19, 3.6, and 3.4.3) are managed through the *CartesI/O* framework [Laurenzi et al., 2019] which relies on the *OpenSoT* library [Mingo Hoffman et al., 2017] 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* [Mingo Hoffman and Tsagarakis, 2021] and solved using efficient QP solvers such as *qpOASES* [Ferreau et al., 2014] or *OSQP* [Stellato et al., 2020].

In the whole-body planner, the first stage (Sect. 3.5.1) is realized using the implementation of AtlasRRT* provided by the *OMPL* library [Sucan et al., 2012], which is appropriately customized in order to include the described feasibility checks. The NLP constituting the second stage (Sect. 3.5.2) is formulated using *CasADi* [Andersson et al., 2019] and solved via *IPOPT* [Wächter and Biegler, 2006].

Collision checks, in both the stance and whole-body planners, are performed using the *Planning Scene* component of the *MoveIT!* framework [Görner et al., 2019], which exploits the *Flexible Collision Library* (*FCL*) [Pan et al., 2012]. To this purpose, the point cloud $\mathscr{P}$ is converted into an *OctoMap* [Hornung et al., 2013], encoding free and occupied space, which is compared with the link meshes from the robot *URDF*.

Finally, a UI has been developed, 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 Sect. 3.8.1), stances describing the tasks assigned to the humanoid, and the inspection of solutions found by the single modules of the planning layer.

3.8 Validation

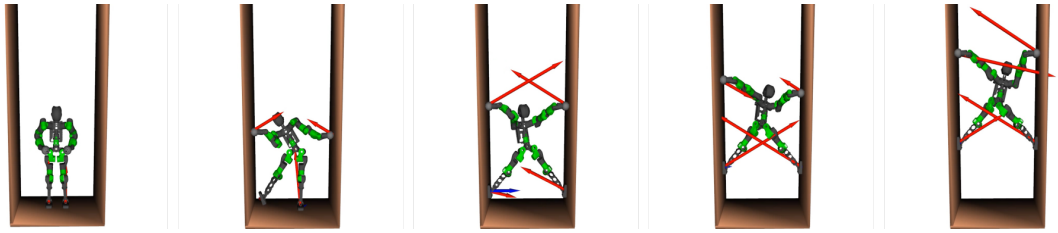
The validation of the proposed MCPC framework has been performed using COMAN+, a torque-controlled humanoid robot designed at Istituto Italiano di Tecnologia. COMAN+ is 1.70 m tall, weights 70 kg and has 28 DoFs: 7 DoFs for each arm, 6 DoFs for each leg, with the two related to the ankle provided by a particular four-bar actuation mechanism [Ruscelli et al., 2018], and 2 DoFs for the torso, allowing roll and yaw rotations. For the purpose validating the proposed method, the anthropomorphic hands of COMAN+ have been replaced with spherical end-effectors.

To provide a comprehensive validation, the next two subsections 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.

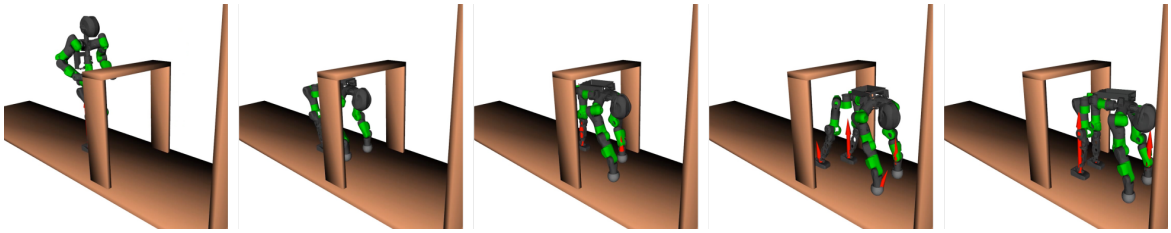
3.8.1 Planning results

To illustrate the performance of the planning layer, this subsection presents numerical results obtained on an Intel Core i7-7500U CPU running at 2.70 GHz. The four multi-contact loco-manipulation tasks shown in Fig. 3.5 have been considered, and each provides some

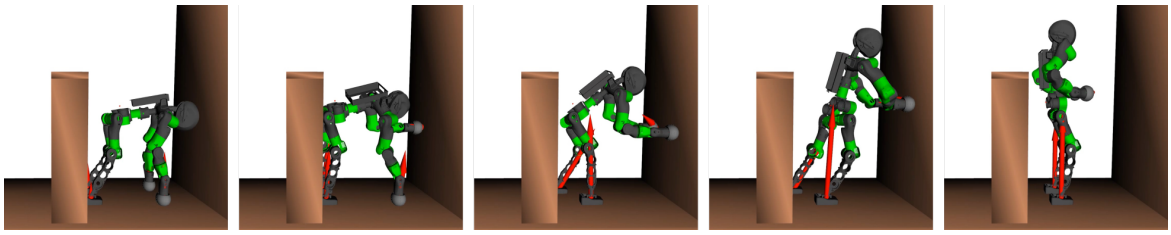
(a) Ladder climbing.



(b) Parallel walls climbing.



(c) Quadrupedal walking.



(d) Standing up.

Figure 3.5 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.

Table 3.1 Averaged performance data of the stance planner.

Task	Planning time (s)	Transition generation time (s)	Number of iterations	Number of vertices in $\mathscr{T}$	Number of stances in $\mathscr{S}_\sigma$
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

snapshots of a typical solution produced by the planning layer. Moreover, the accompanying video 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 can not 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., $U = \{\mathscr{F}^{\text{lf}}, \mathscr{F}^{\text{rf}}, \mathscr{F}^{\text{lh}}, \mathscr{F}^{\text{rh}}\}$, which can establish, respectively, surface and point contacts. The placement of potential contact frames is that shown in Fig. 3.2, except for the ladder climbing task that uses a slightly different placement for the hands, shown in Fig. 3.6, which allows establishing contacts with the rungs in a more natural fashion.

For the second and third tasks, the stance planner has been provided with an intermediate stance σ^{int} (illustrated, respectively, in snapshot 3 of Fig. 3.5-(b) and in snapshot 2 of Fig. 3.5-(c)) to bias the growth of the tree first towards such intermediate stance, and then towards the desired final stance, speeding up the whole process. In particular, when the stance planner chooses to perform a tree expansion attempt via exploitation (see Sect. 3.4.1), contact c^{fin} is randomly picked among those in σ^{int} (σ^{fin}) if a vertex containing σ^{int} does not exist (exists) in $\mathscr{T}$.

The values used for the parameters involved in the planning layer for the four tasks are reported above. For ease of reading, Table 3.2 reports 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 (Sect. 3.4.1), 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 (Sect. 3.4.2) works with $l_{\text{tran}}^{\max} = 100$ and $\Delta T_{\text{tran}} = 1$ s.

- In the IK solver (Sect. 19), $\boldsymbol{K}_{\sigma}$ 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 (Sect. 3.4.3), 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 forth, 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 that uses $\Delta T_{\text{via}} = 2$ s.

Since the planning layer is randomized (as both the stance and whole-body planners rely on probabilistic strategies), it was tested performing 100 runs for each of the four tasks. Thus,

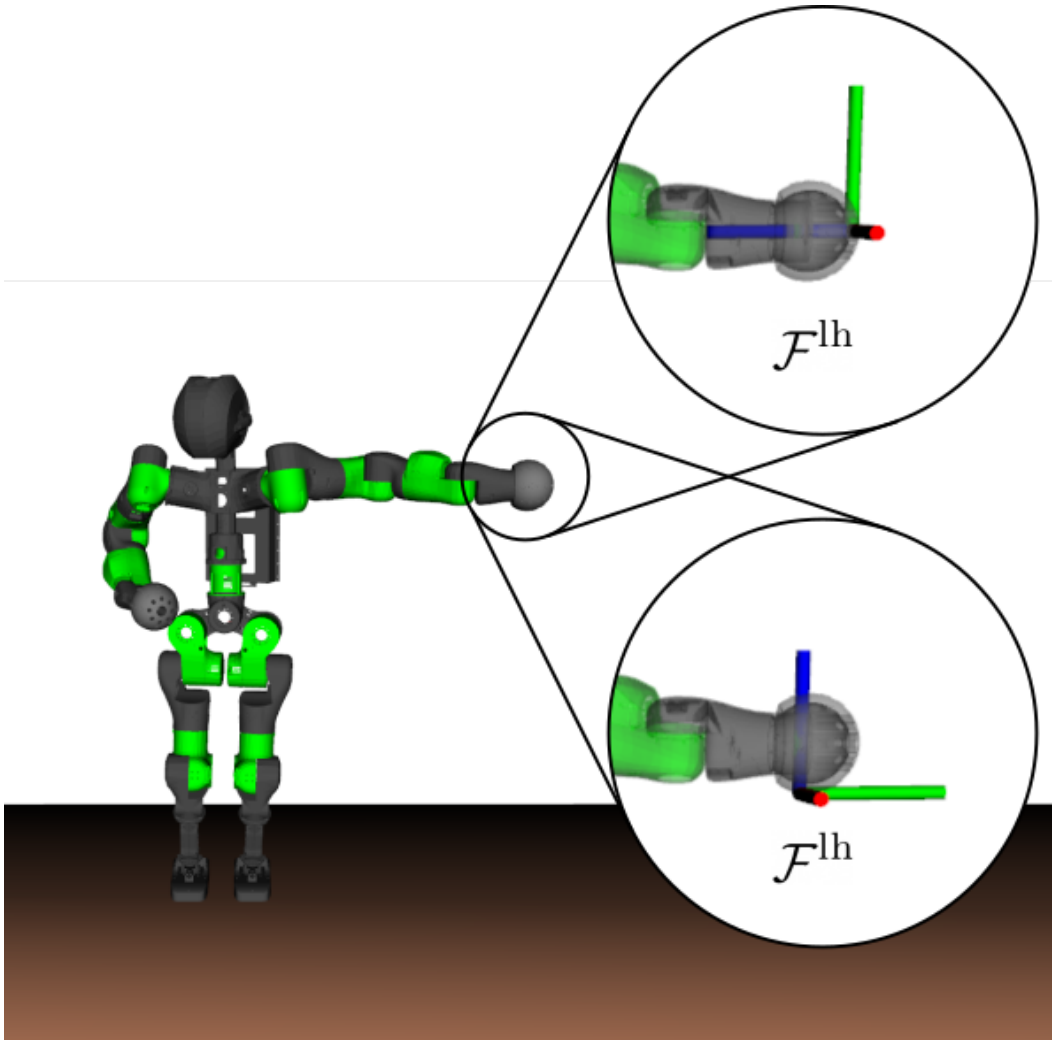


Figure 3.6 The different placement of the potential contact frame on the left hand between the ladder climbing task (bottom) and the other tasks (top). The same applies to the right hand.

Table 3.2 Recap of the most relevant parameters involved in the planning layer. For each parameter, the adopted symbol, the procedure in which it is used, and a synthetic description of its role are reported.

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

in each run, first the sequences $\mathscr{S}_\sigma$, $\mathscr{S}_q$, $\mathscr{S}_W$ of stances, transitions, and contact wrenches were produced using the stance planner; then, the whole-body planner computed the sequence $\mathscr{S}_s$ of trajectories using $\mathscr{S}_\sigma$ and $\mathscr{S}_q$.

Table 3.1 collects the most significant performance data, averaged over the 100 runs, of the proposed stance planner. For each task, 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 $\mathscr{T}$, and the number of stances in the solution sequence $\mathscr{S}_\sigma$ are reported.

It is worth mentioning that the time needed by the whole-body planner to find a solution, provided in input a sequence $\mathscr{S}_q$ containing $N+1$ transitions, is the sum of the N times needed to plan the N trajectories between consecutive configurations in $\mathscr{S}_q$. Each of these times is constituted by those employed by the two stages which are, respectively, fixed at ΔT_{via} and not significant (as it has been observed that the NLP described in Sect. 3.5.2 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.

Figure 3.7 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.

It is worth emphasizing that the 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.

3.8.2 Experiments

The experimental validation was carried out using the *XBot* real-time software framework [Muratore et al., 2020] as control middleware for the COMAN+ humanoid. 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 (3.17) the contact wrenches at the spherical hands, which are sensorless, while the contact wrenches at the feet are directly

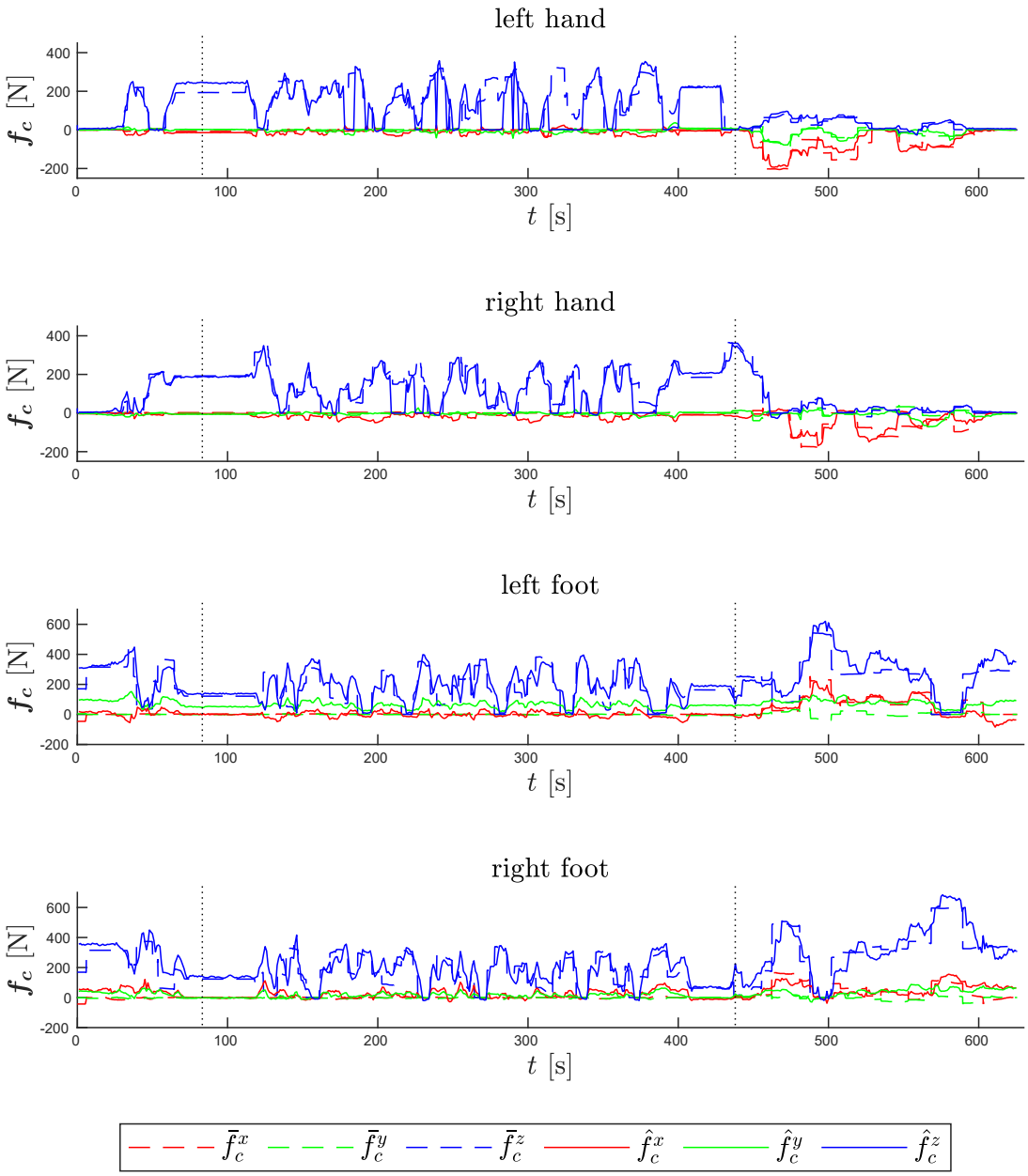


Figure 3.8 Evolution of the contact forces at each end-effector during the whole experiment: reference (dashed lines) vs estimated/measured (solid lines) values.

provided by their force/torque sensors. Both hands and feet soles are covered by a thin layer of rubber.

In the experiment, the robot torque commands are computed via (3.26). It is worth noticing how the choice of the gain matrices $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ in (3.29) 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 (3.26) 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, for the execution of the experiments was 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 switching module to check the achievement of a certain stance, set to 0 N and 30 N, respectively, for each end-effector in U.

The 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, the third scenario (see Fig. 3.5-(c) and 3.5-(d)) described in the previous section was reconstructed; 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 ≈ 1.65 m far from the initial location. A passage is placed at ≈ 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 ≈ 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 the planning layer. Then, COMAN+ was requested to sequentially perform the quadrupedal walking and standing-up motions resulting from the planning stage.

Fig. 3.7 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. 3.8 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 establish contacts with its arm to support the transition to 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 [6] 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.

3.9 Discussion

This section provides some additional comments about the proposed MCPC framework.

3.9.1 Considerations w.r.t. previous works

The two tasks successfully addressed in the experimental validation have been rarely considered in previous works.

Quadrupedal walking with a real humanoid is achieved in Yoshiike et al. [2019] 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. Instead, the proposed methodology 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 the control layer in executing the planned motions.

Standing up exploiting a wall as support is considered in Tonneau et al. [2018]. That work focuses exclusively on the planning aspects and thus the result is shown solely on a simulated humanoid. In contrast, the proposed approach was able to tackle such challenging tasks both at the planning and control level, ultimately achieving experimental results on a real humanoid, which can be considered a particularly relevant result.

[6] https://youtu.be/zS44CegGqow

The stance planner proposed in Tonneau et al. [2018] is, to the best of my 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 the 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 Tonneau et al. [2018] 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., Chung and Khatib [2015]) 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 [Hauser et al., 2008]. The stance planner has the advantage of avoiding the need for any of these operations.

3.9.2 Limitations and possible adaptations

The presented MCPC framework proved to be capable of generating sensible humanoid motions for different multi-contact loco-manipulation tasks proving to be usable in practical applications. However, for the sake of completeness, below are listed three main limitations of the proposed framework that will be tackled in future work.

1. The framework only considers static balance, still representing a valid template for multi-contact planning. In the direction of a dynamic 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., Caron and Kheddar [2016]) 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. The 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. The 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 $\mathscr{S}_q$ produced by the stance planner. A simple solution would be to prune the tree $\mathscr{T}$ of the subtree rooted at $\boldsymbol{q}_j$ (in the spirit of lazy planners, see for example Ferrari et al. [2019]) and restart the stance planner from the resulting tree; the investigation of more effective strategies will be part of future work.

3.10 Conclusions

This chapter 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. The 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. The proposed framework has been validated via both numerical and experimental results obtained on the COMAN+ humanoid robot.

In addition to the adaptations mentioned in Sect. 3.9.2, the presented MCPC framework can be further developed along several lines. A first possibility to consider is its extension to the case of hybrid wheeled-legged quadrupedal robots such as CENTAURO [Kashiri et al., 2019], 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 [Scianca et al., 2021], for example in the presence of an imminent risk of falling.

References

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36.

Atkeson, C. G., Benzun, P. B., Banerjee, N., Berenson, D., Bove, C. P., Cui, X., DeDonato, M., Du, R., Feng, S., Franklin, P., et al. (2018). What happened at the darpa robotics challenge finals. In *The DARPA Robotics Challenge Finals: Humanoid Robots to the Rescue*, pages 667–684. Springer.

Bouyarmane, K., Caron, S., Escande, A., and Kheddar, A. (2019). Multi-contact motion planning and control. In *Humanoid Robotics: A Reference*, pages 1763–1804. Springer Netherlands.

Bouyarmane, K. and Kheddar, A. (2011). 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*, pages 4414–4419.

Bouyarmane, K. and Kheddar, A. (2012). Humanoid robot locomotion and manipulation step planning. *Advanced Robotics*, 26(10):1099–1126.

Bretl, T. (2006). Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *The International Journal of Robotics Research*, 25(4):317–342.

Caron, S. and Kheddar, A. (2016). Multi-contact walking pattern generation based on model preview control of 3D COM accelerations. In *2016 IEEE-RAS International Conference on Humanoid Robots*, pages 550–557.

Caron, S., Pham, Q.-C., and Nakamura, Y. (2015). 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*, pages 5107–5112.

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

Chung, S.-Y. and Khatib, O. (2015). Contact-consistent elastic strips for multi-contact locomotion planning of humanoid robots. In *2015 IEEE International Conference on Robotics and Automation*, pages 6289–6294.

Escande, A., Kheddar, A., and Miossec, S. (2013). Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5):428–442.

Ferrari, P., Cognetti, M., and Oriolo, G. (2019). Sensor-based whole-body planning/replanning for humanoid robots. In *2019 IEEE-RAS International Conference on Humanoid Robots*, pages 511–517.

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363.

Görner, M., Haschke, R., Ritter, H., and Zhang, J. (2019). MoveIt! Task constructor for task-level motion planning. In *2019 IEEE International Conference on Robotics and Automation*, pages 190–196.

Guan, Y. and Yokoi, K. (2006). Reachable space generation of a humanoid robot using the monte carlo method. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1984–1989.

Hauser, K., Bretl, T., Latombe, J.-C., Harada, K., and Wilcox, B. (2008). Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research*, 27(11-12):1325–1349.

Henze, B., Roa, M. A., and Ott, C. (2016). Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios. *The International Journal of Robotics Research*, 35(12):1522–1543.

Herzog, A., Rotella, N., Mason, S., Grimminger, F., Schaal, S., and Righetti, L. (2016). Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

Jaillet, L. and Porta, J. M. (2012). Asymptotically-optimal path planning on manifolds. In *Robotics: Science and Systems*, pages 1–8.

Jamone, L., Natale, L., Sandini, G., and Takanishi, A. (2012). Interactive online learning of the kinematic workspace of a humanoid robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2606–2612.

Kashiri, N., Baccelliere, L., Muratore, L., Laurenzi, A., Ren, Z., Mingo Hoffman, E., Kamedula, M., Rigano, G. F., Malzahn, J., Cordasco, S., et al. (2019). Centauro: A hybrid locomotion and high power resilient manipulation platform. *IEEE Robotics and Automation Letters*, 4(2):1595–1602.

Kingston, Z., Moll, M., and Kavraki, L. E. (2018). Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:159–185.

Laurenzi, A., Mingo Hoffman, E., Muratore, L., and Tsagarakis, N. G. (2019). CartesI/O: A ROS based real-time capable cartesian control framework. In *2019 IEEE International Conference on Robotics and Automation*, pages 591–596.

Laurenzi, A., Mingo Hoffman, E., Polverini, M. P., and Tsagarakis, N. G. (2018). Balancing control through post-optimization of contact forces. In *2018 IEEE-RAS International Conference on Humanoid Robots*, pages 320–326.

Lee, S.-H. and Goswami, A. (2012). A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Autonomous Robots*, 33(4):399–414.

Mingo Hoffman, E., Rocchi, A., Laurenzi, A., and Tsagarakis, N. G. (2017). Robot control for dummies: Insights and examples using OpenSoT. In *2017 IEEE-RAS International Conference on Humanoid Robotics*, pages 736–741.

Mingo Hoffman, E. and Tsagarakis, N. G. (2021). The Math of Tasks: a domain specific language for constraint-based task specification. *International Journal of Humanoid Robotics*, 18(3):2150008.

Mordatch, I., Todorov, E., and Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *Transactions on Graphics*, 31(4):1–8.

Muratore, L., Laurenzi, A., Mingo Hoffman, E., and Tsagarakis, N. G. (2020). The XBot real-time software framework for robotics: From the developer to the user perspective. *IEEE Robotics & Automation Magazine*, 27(3):133–143.

Pan, J., Chitta, S., and Manocha, D. (2012). FCL: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866.

Park, J. H. (2019). Compliance/impedance control strategy for humanoids. In *Humanoid Robotics: A Reference*, pages 1009–1028. Springer Netherlands.

Polverini, M. P., Laurenzi, A., Mingo Hoffman, E., Ruscelli, F., and Tsagarakis, N. G. (2020). Multi-contact heavy object pushing with a centaur-type humanoid robot: Planning and control for a real demonstrator. *IEEE Robotics and Automation Letters*, 5(2):859–866.

Polverini, M. P., Mingo Hoffman, E., Laurenzi, A., and Tsagarakis, N. G. (2019). Sparse optimization of contact forces for balancing control of multi-legged humanoids. *IEEE Robotics and Automation Letters*, 4(2):1117–1124.

Rossini, L., Mingo Hoffman, E., Laurenzi, A., and Tsagarakis, N. (2021). NSPG: An efficient posture generator based on null-space alteration and kinetostatics constraints. *Frontiers in Robotics and AI*, 8:715325.

Ruscelli, F., Laurenzi, A., Mingo Hoffman, E., and Tsagarakis, N. G. (2018). A fail-safe semi-centralized impedance controller: Validation on a parallel kinematics ankle. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–9.

Ruscelli, F., Polverini, M. P., Laurenzi, A., Mingo Hoffman, E., and Tsagarakis, N. G. (2020). 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*, pages 3869–3876.

Saab, L., Ramos, O. E., Keith, F., Mansard, N., Soueres, P., and Fourquet, J.-Y. (2013). Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transactions on Robotics*, 29(2):346–362.

Scianca, N., Ferrari, P., De Simone, D., Lanari, L., and Oriolo, G. (2021). A behavior-based framework for safe deployment of humanoid robots. *Autonomous Robots*, pages 1–22.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672.

Stephens, B. J. and Atkeson, C. G. (2010). Dynamic balance force control for compliant humanoid robots. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1248–1255.

Sucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82.

Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., and Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics*, 34(3):586–601.

Trinkle, J. C., Pang, J.-S., Sudarsky, S., and Lo, G. (1997). On dynamic multi-rigid-body contact problems with coulomb friction. *Journal of Applied Mathematics and Mechanics*, 77(4):267–279.

Vaillant, J., Kheddar, A., Audren, H., Keith, F., Brossette, S., Escande, A., Bouyarmane, K., Kaneko, K., Morisawa, M., Gergondet, P., et al. (2016). Multi-contact vertical ladder climbing with an HRP-2 humanoid. *Autonomous Robots*, 40(3):561–580.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57.

Werner, A., Henze, B., Rodriguez, D. A., Gabaret, J., Porges, O., and Roa, M. A. (2016). Multi-contact planning and control for a torque-controlled humanoid robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5708–5715.

Yoshiike, T., Kuroda, M., Ujino, R., Kanemoto, Y., Kaneko, H., Higuchi, H., Komura, S., Iwasaki, S., Asatani, M., and Koshiishi, T. (2019). The experimental humanoid robot E2-DR: A design for inspection and disaster response in industrial environments. *IEEE Robotics & Automation Magazine*, 26(4):46–58.