



# Towards Safe and Stable Landing Control for Quadruped Robots

by

**Francesco ROSCIA**

Istituto Italiano di Tecnologia, Italy
and
Università di Genova, Italy

A thesis submitted for the degree of
*Doctor of Philosophy* (XXXVI cycle)

To be publicly defended on

20 February 2024

**Francesco Roscia**
*Towards Safe and Stable Landing Control for Quadruped Robots*
Candidate Student for the Ph.D. Program in *Bioengineering and Robotics*
Curriculum in *Advanced and Humanoid Robotics*
Genoa, Italy. 15 December 2023

Tutors:

**Dr. Michele Focchi**
*Dynamic Legged Systems (DLS)*, Istituto Italiano di Tecnologia, Italy
*Dipartimento di Ingegneria e Scienza dell'Informazione (DISI)*, Università di Trento, Italy

**Dr. Claudio Semini, Principal Investigator**
*Dynamic Legged Systems (DLS)*, Istituto Italiano di Tecnologia, Italy

External Advisor:

**Dr. Andrea Del Prete, Associate Professor**
*Dipartimento di Ingegneria Industriale (DII)*, Università di Trento, Italy

Head of the Ph.D. Program:

**Dr. Paolo Massobrio, Associate Professor**
*Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS)*, Università di Genova, Italy

Reviewers:

**Dr. Paolo Rocco, Full Professor**
*Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)*, Politecnico di Milano, Italy

**Dr. Donghyun Kim, Assistant Professor**
*Manning College of Information and Computer Sciences (CICS)*, University of Massachusetts Amherst, MA, United States

*Abbi il culto sacro (io lo dico per te... e per me) per tutto ciò che può esaltare ed eccitare la tua intelligenza.*

*You must have sacred worship (I say this for you... and for me) for everything that can exalt and excite your intelligence.*

Amedeo Modigliani, *Letter to Oscar Ghiglia* (1903).
Published by Paolo D'Ancona on the magazine *L'Arte* (1930).
Free translation by the thesis' author.

# Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

*Francesco Roscia*
*Genoa, Italy, 15 December 2023*

# Acknowledgements

In opting for an active voice in my thesis, readers will frequently encounter the pronouns *I*, *my*, and *me* and *we*, *our*, and *us* throughout the text. I would like to emphasize that the work presented here is the culmination of collaborative efforts involving several individuals who have supported me over the past three years.

I am profoundly grateful to my supervisors, Dr. Michele Focchi, Dr. Claudio Semini, and Prof. Andrea Del Prete, whose guidance, encouragement, and unwavering support have been instrumental throughout my doctoral journey. Their mentorship, invaluable insights, and continuous belief in my abilities have shaped not only my academic pursuits but also my personal growth.

I am sincerely thankful to Prof. Paolo Rocco and Prof. Donghyun Kim, reviewers of my dissertation, for generously sharing their valuable feedback, insightful suggestions, and extensive expertise to enrich the content and quality of this work.

I am indebted to Istituto Italiano di Tecnologia for providing an exceptional academic environment, resources, and facilities that facilitated my research endeavors.

I wish to express my appreciation to my colleagues and fellow researchers at the Dynamic Legged Systems lab, whose collaboration, discussions, and shared experiences have contributed immensely to the development of this thesis.

I would like to acknowledge Andrea Cumerlotti for taking part, during its Master thesis, in the mechanical development of the Orientation Control System reported in Chapter 3.

My heartfelt thanks go to my family for their unwavering love, understanding, and encouragement throughout this academic endeavor. Their patience, support, and belief in me have been my constant motivation.

Lastly, I am deeply grateful to all those individuals, friends, and mentors whose support, motivation, and encouragement have played a pivotal role in shaping this academic milestone.

# Abstract

In the last decades, legged robots have increasingly matured and demonstrated versatile locomotion capabilities. Recent advances encourage the use of quadrupedal morphology to accomplish highly dynamic motions. Different types of gaits, such as trot or crawl, have been developed. However, sometimes there is no way to get around or over an obstacle with the gaits mentioned above, and jumping motions are required. In contrast to advances in jump planning, relatively little research has been done on landing strategies, that permit safely recovering after unexpected falls or planned jumps. These abilities are beneficial for navigating harsh environments and preventing significant damage to the robot.

The overall research objective of my Ph.D. research aims to fill the gap on the topic of quadrupedal robots' safe and stable landing after high drops. Specifically, I decided to find a solution for the following two problems.

1. Can a quadruped robot have the aerial righting reflexes necessary to reorient itself as it falls?
2. Using only proprioceptive measurements, can a quadruped robot detect a high fall and react in order to safely land?

To answer the first question, I observed that most quadrupeds are designed with lightweight legs, causing the limbs to have minimal impact on the overall angular momentum. Thus, I designed an Orientation Control System (OCS) composed of two rotating masses to gain control authority over the trunk orientation when the robot has no contact with the environment. The axes of rotation of the flywheels are set to be incident, enabling continuous controllability in both roll and pitch directions while keeping the device compact. The approach was tested in a simulation environment using the 2.5 kg torque-controlled quadruped robot Solo12.

Not negligible horizontal velocity for the robot's Center of Mass (CoM) makes the second question much more challenging. In some situations, ignoring it is not an option. For instance, when a quadruped trots with a high speed

and must do a leap without first stopping the motion. To tackle the landing problem, I proposed a reactive Landing Controller (LC). Knowing neither the distance to the landing surface nor the flight time, it reacts to make the robot ready to dissipate the kinetic energy once the touch down is identified. Relying on the notions of capturability and Zero Moment Point, the proposed controller fulfills the following requirements: 1) no bounces should be produced after landing, 2) the trunk must not hit the ground, 3) the robot must reach a stable standing state, and 4) once landed, the feet must not slip. This is the first time a quadruped robot can successfully recover from falls with horizontal velocities up to 3 m/s in simulation. Experiments demonstrate that the torque-controlled quadruped Go1 can successfully achieve a stable configuration after falls when subjected to manually induced horizontal velocities and angular perturbations.

When the horizontal velocity of the quadruped when it falls is larger than the maximum one the LC can stabilize, reaching a stable standing state is impossible. The robot falls down and hardware damage can occur. Conscious of the robot's kinematic, actuation, and friction limits, the Extended Landing Controller (ELC) relaxes the above requirement. During the fall, it plans joint references that avoid collisions between the trunk and the ground and allow for one or more additional aerial phases, as the gymnasts do when they lose balance after touching the ground. Preliminary results considering Go1 free falling with a horizontal velocity of 4 m/s show that if reducing horizontal velocity between a touch down and the subsequent lift off is viable, the combined use of ELC and LC eventually leads the robot to a stable standing state.

Locosim is the underlying enabling technology for these projects: it is an open-source, cross-platform robotics framework for working either in a simulation environment or with real hardware. Benefiting from Python and ROS and integrating features for computation of robots kinematics and dynamics, logging, plotting, and visualization, it considerably helps roboticists who need a starting point for rapid code prototyping.

**Keywords:** Legged Locomotion, Quadruped Robots, Recovery Strategy, Landing Control, Angular Momentum, Robotics Framework

# Contents

# List of Figures

# List of Tables

# Nomenclature

## Notation

| | |
|---|---|
| $a$ | scalar |
| $\boldsymbol{a}$ | vector |
| $\boldsymbol{A}^T$ | transpose matrix |
| $\boldsymbol{A}^{-1}$ | inverse matrix |
| $\boldsymbol{A}^{-T}$ | inverse transpose matrix |
| $\boldsymbol{A}^{\#}$ | pseudoinverse matrix |
| $a_k$ | $k$-th entry of the vector $\boldsymbol{a}$ |
| $\boldsymbol{A}$ | matrix |
| $a_{k,\,j}$ | entry at the $k$-th row, $j$-th column of the matrix $\boldsymbol{A}$ |

## Symbols

| | |
|---|---|
| $\boldsymbol{c}$ | CoM position |
| $\dot{\boldsymbol{c}}$ | CoM velocity |
| $\ddot{\boldsymbol{c}}$ | CoM acceleration |
| $\mathbf{dfk}_k$ | direct kinematic function of the $k$-th body |
| $\mathbf{dfk}_{Rk}$ | rotational direct kinematic function of the $k$-th body |
| $\mathbf{dfk}_{pk}$ | translational direct kinematic function of the $k$-th body |
| $\boldsymbol{\mathcal{F}}$ | generalized forces |
| $\mathcal{FC}_k$ | Friction cone of the $k$-th body |
| $\boldsymbol{f}_k$ | external force acting on the $k$-th body |
| $\boldsymbol{g}$ | gravity vector |

| | |
|---|---|
| $\boldsymbol{g}(\boldsymbol{q})$ | vector of gravitational effects |
| $\boldsymbol{h}$ | vector of nonlinear effects |
| $\boldsymbol{I}_{n\times n}$ | identity matrix of dimension $n \times n$ |
| $\boldsymbol{O}_{n\times n}$ | null matrix of dimension $n \times n$ |
| $\boldsymbol{I}_k$ | inertia matrix of the $k$-th body expressed with respect to the inertial frame |
| ${}_k\boldsymbol{I}_k$ | inertia matrix of the $k$-th body expressed with respect to its CoM |
| $\boldsymbol{J}_k$ | Jacobian matrix of the $k$-th body |
| ${}_{\mathcal{B}}\boldsymbol{J}_{Rk}$ | rotational Jacobian matrix for the kinematic chain from the base to the $k$-th body |
| ${}_{\mathcal{B}}\boldsymbol{J}_{pk}$ | traslational Jacobian matrix for the kinematic chain from the base to the $k$-th body |
| $\boldsymbol{J}_{Rk}$ | rotational Jacobian matrix of the $k$-th body |
| $\boldsymbol{J}_{Rb}$ | matrix mapping orientation rates to angular velocity |
| $\boldsymbol{J}_{pk}$ | traslational Jacobian matrix of the $k$-th body |
| $\boldsymbol{L}$ | angular momentum |
| $m$ | total mass of the robot |
| $\boldsymbol{M}$ | mass matrix |
| $m_k$ | mass of the $k$-th body |
| $\mu$ | Coulomb friction coefficient |
| $\boldsymbol{\mu}_k$ | external moment acting on the $k$-th body |
| $\boldsymbol{N}$ | matrix of Coriolis and centrifugal forces |
| $n_c$ | number of contacts |
| $n_j$ | number of joints |
| $\boldsymbol{\omega}_k$ | angular velocity of the $k$-th body w.r.t. the inertial frame |
| $\dot{\boldsymbol{\omega}}_k$ | angular acceleration of the $k$-th body w.r.t. the inertial frame |
| $\boldsymbol{\phi}_b$ | base orientation |
| $\dot{\boldsymbol{\phi}}_b$ | base orientation rate |
| $\boldsymbol{p}_k$ | position of the $k$-th body w.r.t. the inertial frame |
| $\dot{\boldsymbol{p}}_k$ | linear velocity of the $k$-th body w.r.t. the inertial frame |
| $\ddot{\boldsymbol{p}}_k$ | linear acceleration of the $k$-th body w.r.t. the inertial frame |
| $\mathcal{Q}$ | configuration space |
| $\boldsymbol{q}$ | generalized coordinates |
| $\dot{\boldsymbol{q}}$ | generalized velocities |
| $\ddot{\boldsymbol{q}}$ | generalized accelerations |
| $\hat{\boldsymbol{q}}$ | joints position |
| $\hat{\boldsymbol{q}}_{lb}$ | joints position lower bound |

| | |
|---|---|
| $\hat{\boldsymbol{q}}_{ub}$ | joints position upper bound |
| $\dot{\hat{\boldsymbol{q}}}$ | joints velocity |
| $\ddot{\hat{\boldsymbol{q}}}$ | joints acceleration |
| $\mathbb{R}$ | field of real numbers |
| $\mathbb{R}^+$ | set of real positive numbers |
| $\mathbb{R}^n$ | $n$-dimensional vector space of real numbers |
| $\mathbb{R}^{n\times m}$ | set of all $n$-by-$m$ matrices over the field of real numbers |
| $\boldsymbol{R}_k$ | rotation matrix of the $k$-th body w.r.t. the inertial frame |
| $\dot{\boldsymbol{R}}_k$ | time derivative of the rotation matrix of the $k$-th body w.r.t. the inertial frame |
| $\boldsymbol{S}$ | selection matrix for the Lagrangian dynamics |
| $\boldsymbol{S}_i$ | matrix selecting quantities associated to the $i$-th leg |
| $t$ | time |
| $\hat{\boldsymbol{q}}$ | joints torque |
| $\boldsymbol{\tau}_{lb}$ | joints torque lower bound |
| $\boldsymbol{\tau}_{ub}$ | joints torque upper bound |
| $\boldsymbol{u}$ | center of pressure/zero moment point |
| $\boldsymbol{w}$ | wrench |
| $\boldsymbol{\xi}$ | capture point |

# Acronyms

| | |
|---|---|
| **CD** | Centroidal Dynamics |
| **CMG** | Control Moment Gyroscope |
| **CoM** | Center of Mass |
| **CoP** | Center of Pressure |
| **CP** | Capture Point |
| **CSP** | Constraint Satisfaction Problem |
| **CWC** | Contact Wrench Cone |
| **DDP** | Differential Dynamic Program |
| **ELC** | Extended Landing Controller |
| **FWP** | Feasible Wrench Polytope |
| **GRF** | Degree of Freedom |
| **GRF** | Ground Reaction Force |

| | |
|---|---|
| **HAA** | Hip Abduction/Adduction |
| **HFE** | Hip Flexion/Extension |
| **IK** | Inverse Kinematic |
| **IMU** | Inertial Measurement Unit |
| **IP** | Iterative Projection |
| **KFE** | Knee Flexion/Extension |
| **LC** | Landing Controller |
| **LIP** | Linear Inverted Pendulum |
| **LO** | Lift Off |
| **LP** | Linear Program |
| **LTI** | Linear Time Invariant |
| **LTV** | Linear Time Varying |
| **MICP** | Mixed-Integer Convex Program |
| **MPC** | Model Predictive Control |
| **MSD** | Mass-Spring-Damper |
| **NLP** | Nonlinear Program |
| **NMPC** | Nonlinear Model Predictive Control |
| **NN** | Neural Network |
| **OCP** | Optimal Control Problem |
| **OCS** | Orientation Control System |
| **ODE** | Ordinary Differential Equation |
| **OS** | Operating System |
| **PD** | Proportional Derivative |
| **pWBC** | projection-based Whole Body Control |
| **QP** | Quadratic Program |
| **RL** | Reinforcement Learning |
| **SLIP** | Spring-Loaded Inverted Pendulum |
| **SP** | Support Polygon |
| **SRBD** | Single Rigid Body Dynamics |
| **TD** | Touch Down |
| **TO** | Trajectory Optimization |
| **UDP** | User Datagram Protocol |
| **VHSIP** | Variable Height Springy Inverted Pendulum |
| **WBC** | Whole Body Control |
| **ZMP** | Zero Moment Point |

# 1

# Introduction

*The attempt to create and bring to life artificial beings that employ legged locomotion is characterized by noteworthy advancements and innovations throughout humankind's history. Numerous ancient myths feature the presence of crafted creatures, e.g., the mechanical servants of the Greek god Hephaestus, the clay golems in Jewish folklore, and the legendary statue of Pygmalion that came to life. Leonardo da Vinci (late XV century) engaged in the design and conceptualization of mechanical walking machines* automata*, illuminating the enduring allure of legged robotics. Nevertheless, vehicles with four limbs start appearing only in the 1960s, with the pioneering* General Electric quadruped *and* Phoney Poney*. Progress in the domains of mechanics, electronics, control, and computing has assisted the refinement of quadruped robots to the extent that it has been possible to bring them out of research laboratories and make them commercially available. Nowadays, the field of legged robotics is a promising realm of research with the potential to reshape the world by developing systems capable of performing dangerous, monotonous, or undesirable tasks currently carried out by humans. Safe and stable aerial motion for four-legged robots is still an open research topic.*

## 1.1 Motivations

Robotic system designed for operation in unstructured environments must possess versatile locomotion skills to navigate through obstacles, gaps, rough terrains, inclined surfaces, and staircases. A significant number of mobile robots employ wheeled or tracked locomotion due to low system complexity and efficient motion. Nevertheless, their adaptability to uneven terrains is limited by the size of wheels or tracks. To overcome the challenges presented by uneven terrain, researchers started to develop legged robots. These robots exhibit enhanced mobility over rough terrain allowing legs to temporarily leave their contact with the ground. As a matter of fact, discrete footholds are sufficient to *natural* legged systems to navigate through uneven terrains in a myriad of different ways. Regardless of their inherent differences, a few illustrative examples include humans ascending and descending the stairs of a building, rescue dogs adeptly navigating through debris in earthquake-affected areas, cats skillfully leaping on and off furniture, and ibex climbing steep mountain walls. All extremely difficult, if not impossible, activities for wheeled and tracked vehicles.

The number of legs plays an important role in balance stability and locomotion versatility. Inspired by human capabilities, the field of bipedal robotics has evolved, leading to robots endowed with a high degree of flexibility and adaptability, particularly when dealing with limited support, such as when climbing ladders. However, the control of locomotion on uneven terrains continues to pose a challenge in this context. On the other hand, multi-legged robots demonstrate superior balancing stability and resilience against external disturbances, primarily due to the larger number of supports.

Quadrupedal robots have garnered attention due to their potential for highly dynamic behaviors. These robots have a combination of stability and maneuverability that sets them apart from their counterparts. Central to the success of quadrupedal robots is the development of various gaits, or walking patterns, enabling navigation in different environments. Gaits like trotting and crawling have proven to be efficient and effective for traversing a range of surfaces, from smooth floors to uneven terrains. However, there are limitations. There are situations where the above-mentioned locomotion strategies are insufficient, particularly when it comes to overcoming obstacles or large gaps. Executing an aerial motion simplifies tremendously the task of traversing challenging scenarios. In the context of legged systems, aerial motions encompass a series of

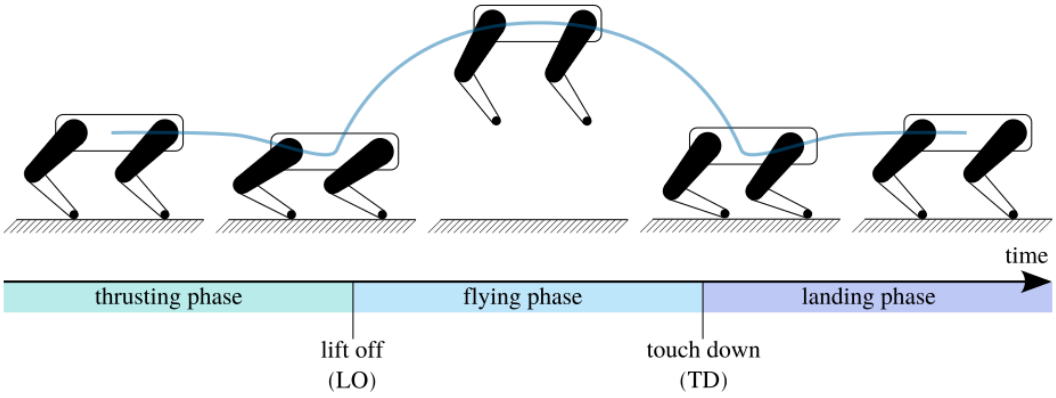



**Figure 1.1:** The three phases of aerial motions, including the two transition events between the phases.

dynamic movements for which the system is voluntarily requested to detach all the contacts with the surrounding environment, allowing it to bypass or surmount obstacles that would be inaccessible with traditional gaits. These movements include leaps, somersaults, spins, flips, and more. In a broad sense, all forms of aerial motions share a common structure, consisting of a sequence of three well-defined phases, see Figure 1.1. During the *thrusting phase* the robot propels itself by exerting the Ground Reaction Forces (GRFs) necessary to break all contacts with the surrounding environment. The subsequent *flying phase* is characterized by the robot's aerial maneuvers. Influenced solely by gravity force, its CoM moves along the ballistic trajectory dictated by the Lift Off (LO) conditions, while its orientation can be altered by modifying the mass distribution, e.g., stretching its limbs. Ultimately, the Touch Down (TD) phase occurs, and during the *landing phase*, new contacts become active. Possibly, the robot stabilizes its posture, actively preventing undesired motions like feet slippage, bounces, and collisions. This sequence of phases, from thrusting to flying and landing, presents a multifaceted array of research topics and engineering challenges. Addressing them involves the development of advanced control algorithms and innovative hardware solutions. Research and development in this domain not only pushes the current boundaries of legged robotics but also has implications for various applications, such as disaster response, exploration, and industrial automation, where navigating complex and unstructured terrains is required.

Aerial motions open up new possibilities for legged robots and offer a range of compelling benefits and applications, expanding the capabilities of these

systems. Here, I present some of the advantages of aerial motions and highlight examples and use cases that illustrate their potential.

### Overcoming Insurmountable Obstacles

One of the primary advantages of aerial motions is the capacity to help legged robots conquer obstacles that are otherwise insurmountable, see Figure 1.2. For instance, in search and rescue scenarios, where rubble and debris can create impassable barriers, aerial motions allow robots to leap over obstructions, reach difficult-to-access areas, and locate survivors.

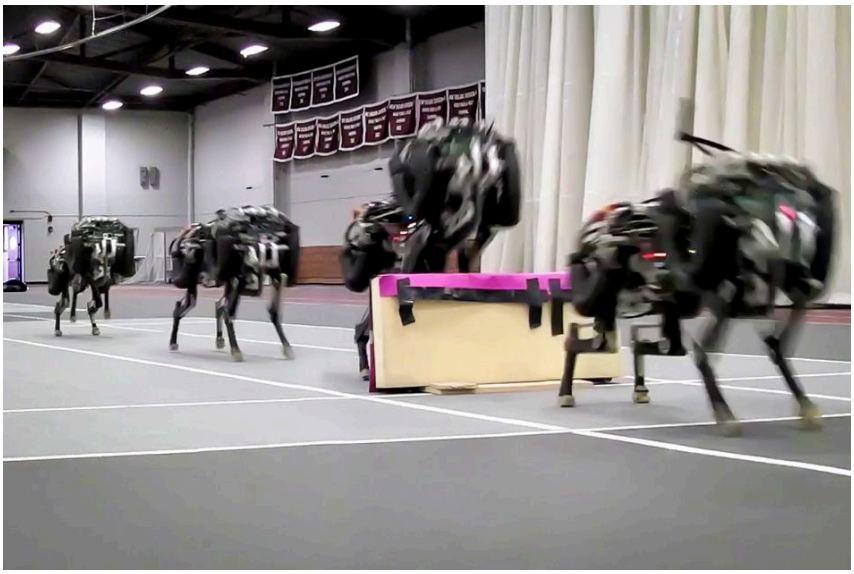


**Figure 1.2:** MIT Cheetah 2 robot demonstrated the ability to leap over obstacles ©MIT [Chu, 2015].

### Agile Maneuverability

Aerial motion provides legged robots with agile maneuverability in complex environments. Robots can execute rapid spins or flips to quickly change their orientation or evade obstacles. This is particularly valuable in environments with tight or complex spaces, where conventional gaits may be impractical. In robotics labs and competitions, the development of agile legged robots continues to explore their capabilities for dynamic maneuvering, see Figure 1.3. For example, in urban search and rescue missions, robots with aerial capabilities will be able to navigate confined spaces.

**Figure 1.3:** The Berkley's Salto (SAltorial Locomotion Terrain Obstacles) robot is capable of executing several jumps across home's furniture. ©2020 IEEE [Yim et al., 2020].

### Enhanced Exploration Capability

In scientific exploration and environmental monitoring, aerial motion enables legged robots to access challenging or remote locations. Space agencies have expressed interest in developing robotic systems that can use aerial motion for exploring challenging terrains on other planets, such as lunar caves or Martian cliffs, Figure 1.4. Jumps represent an energetically efficient way to move in low-gravity environments, enabling to overcome obstacles and cover vast distances over rough terrains. This attribute is invaluable in planetary exploration scenarios, where a robot capable of leaping or somersaulting across the rocky, cratered, and sandy surfaces of celestial bodies can be deployed to navigate terrains that are challenging to traverse using traditional walking gaits.

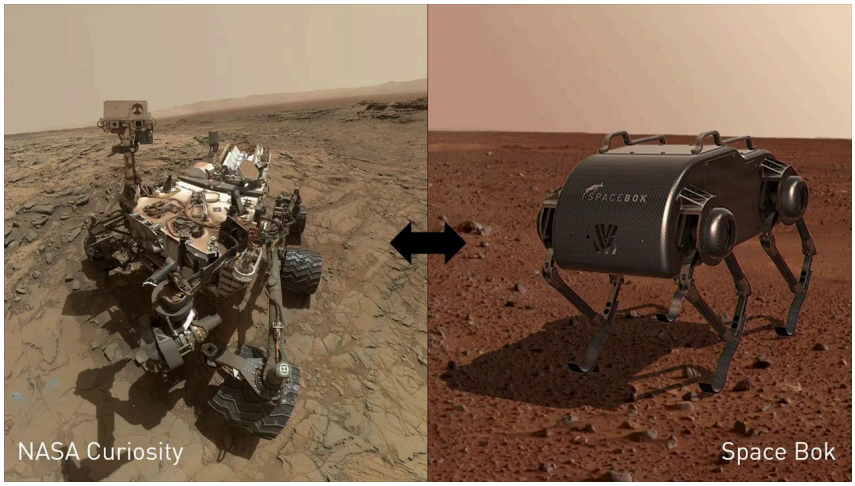



**Figure 1.4:** ESA has been involved in the development of SpaceBok, a quadruped robot from ETH Zurich and ZHAW Zurich. The robot is being used to investigate the potential of dynamic walking and aerial motion to get around in low gravity environments [Kolvenbach et al., 2019].

### Entertainment and Sports

Aerial motions open up new possibilities in entertainment and sports. Imagine a robot capable of executing acrobatic feats in circus performances, or a robot athlete performing flips and spins in gymnastics or parkour competitions. The entertainment industry has explored the use of dynamic legged robots in theme park attractions, immersive experiences, and live performances, see Figure 1.5. These robots not only serve as engaging entertainers but also provide insights into how machines can emulate the agility of human performers.

## 1.2 Objective

While substantial strides have been made in advancing legged robots' thrusting phase, i.e., the initial part of an aerial motion, relatively little research has been done on controllers for the flying and landing phases. Improving them can have a beneficial impact on navigating harsh environments and preventing significant damage after unexpected falls or planned jumps. Throughout the course of this thesis, the reader must keep the following statement in mind:

*It's not the fall that kills you. It's the sudden stop at the end.*[1]

[1]The sentence is attributed to British writer and comedian Douglas Adams (1952 - 2001).

**Figure 1.5:** The Walt Disney Imagineering tested a small swarm of BD-1 droid units roaming all over *Star Wars: Galaxys Edge* in Disneyland to see how they work and can interact with the environment, as well as guests. The robots were part of what Disney calls a "robotic character pipeline". ©2023 Disney.

My overarching objective is to fill the gap on the topic of quadrupedal robots safe and stable landing after high drops. The aim of my Ph.D. research can be encapsulated by two fundamental issues. First, I investigate whether a quadruped robot can be equipped with the aerial righting reflexes to dorsoventrally reorient its main body during the flying phase so as to prepare itself for the landing by pointing limbs toward the ground. Second, I explore whether a quadruped, equipped solely with proprioceptive measurements, can detect a high fall and adapt its limb configuration to prevent the robot from tumbling on the ground once landed. These questions serve as guiding principles of research on aerial locomotion, reflecting a quest for enhancing the capabilities of legged robots and their adaptability to complex and dynamic environments. Indeed, it is imperative to recognize that if a quadruped robot does not have its limbs correctly oriented toward the ground at the end of the flying phase, it jeopardizes its ability to swiftly adjust its joint configurations upon landing, which may result in the robot crashing onto the ground.

## 1.3 Contributions

The original contributions included in this thesis comprise content from both previously published and unpublished papers. These contributions can be categorized into three distinct topics.

### Orientation Control System and its Control Algorithm

I propose a hardware modification in conjunction with a control algorithm, named Orientation Control System (OCS) [Roscia et al., 2023a]. This modification incorporates two rotating and actuated masses, referred to as *flywheels* or *reaction wheels*. They enable the robot to gain control authority over its orientation. Because of the conservation of angular momentum, the control algorithm adjusts the flywheels' speed to steer the trunk orientation, even in the absence of contacts with the ground. Notably, the flywheels are designed with intersecting axes of rotation, resulting in a compact mechanism that can proficiently control both roll and pitch angles. Extensive simulations involving the quadruped Solo12 across various scenarios illustrate the effectiveness of the proposed approach in rejecting disturbances, tracking angular references, and damping base oscillations after the TD.

### Safe Landing Controller Handling Horizontal Velocities

I address the challenges for quadruped robots associated with fall detection and safely landing on a flat horizontal surface. I consider scenarios where information about the distance to the landing surface and flight time are unknown. Specifically, I propose a real-time reactive Landing Controller (LC) [Roscia et al., 2023b] capable of handling substantial horizontal velocity, insensitive to touch down timing uncertainties, and independent from the need to estimate the robot's absolute pose and velocity. The controller relies on the notions of capturability and Zero Moment Point (ZMP) to adeptly adjust the limbs' posture, guiding the robot toward a stable standing configuration during the landing phase. This adjustment avoids undesirable effects like bouncing, feet slippage, and trunk collisions with the ground. This research represents the first time a quadruped robot successfully recovered from falls with a horizontal velocity up to 3 m/s, as validated through simulation. Experiments conclusively demonstrate that the 12 kg torque-controlled quadruped robot Go1 can consistently achieve a standing still posture from omnidirectional drops charac-

terized by diverse manually induced horizontal velocities and varying angular perturbations.

### Extended Landing Controller with Multiple Bounces

In scenarios where the horizontal velocity of a falling quadruped exceeds the stabilization capacity of the LC, achieving a stable standing state becomes unattainable. Under these circumstances, the robot experiences a tip-over, which may result in hardware damage. Executing additional aerial motions could be a solution. Aware of the robot's kinematic limits, actuation capabilities, and friction constraints, I have designed the Extended Landing Controller (ELC) to strategically relax the strict requirement of immediately achieving a stable standing state upon TD detection. It generates references with the purpose of dissipating the kinetic energy over multiple landing phases rather than in only one. Preliminary results, conducted using the Go1 quadruped robot, indicate that reducing the horizontal velocity between the TD and subsequent LO will eventually guide the robot towards achieving a stable standing state.

### The Robotics Framework Locosim

In order to realize the projects mentioned above, a robotics framework became a requisite. In its most general definition, a robotics framework encompasses a collection of programs and data conforming to well-defined principles, e.g., syntax, data structure, and folders and files management, governing the operation of robots. I joined the Locosim development team. Locosim [Focchi et al., 2024] stands as an open-source, cross-platform robotics framework, meticulously designed to seamlessly accommodate both simulation environments and real hardware implementations. In the pursuit of its development, I have set pre-requisites that a robotics framework should embody, including generality, modularity, reusability, extensibility, the delicate balance between rapid prototyping and performance, a feature-rich architecture, and support for end-users development. Locosim, relying on Python and ROS, offers a comprehensive set of features encompassing robots' kinematics and dynamics computation, logging, plotting, and visualization. This framework proves to be a valuable asset for roboticists seeking a robust foundation for fast code prototyping.

## 1.4 Outline

The following chapters delve into controllers aiming to stabilize the flying and landing phases of areal motions of quadruped robots. The remainder of the thesis is structured as follows.

Chapter 2 offers a comprehensive examination of the state of the art on aerial locomotion.

Chapter 3 dives into simultaneously controlling the quadruped robot's roll and pitch orientation during its flying phase. It describes the OCS, which relies on the implementation of two flywheels.

In Chapter 4 the discussion shifts to the linear dynamics of the system. I proposed the LC framework seeking an answer to the question: during the flying phase, how should a quadruped adjust its limbs to prevent undesired collisions with the ground after the TD?

Chapter 5 investigates about the algorithm designed to extend the capabilities of the LC, namely the ELC, that enables landing on multiple bounces by taking into account kinematic, actuation, and frictional constraints that limit the robot's dynamics.

Chapter 6 provides a comprehensive discussion of the results obtained in this dissertation, along with a summary of the key concepts presented. Additionally, it offers insights into potential future research developments.

Some appendices contribute to essential topics for this thesis. Appendix A presents Locosim, an open-source robotics framework whose development team I was part of. Appendix B recalls some notions about the dynamic modeling of legged systems. Appendix C describes the intrinsic differences between holonomic and nonholonomic mechanical systems. Finally, Appendix D summarizes the concept of first integrals of the dynamics and Appendix E describes the concept of orientation error.

# 2

# Related Works and Background

*The recent decades have witnessed significant advancements in the field of robotics, yielding legged robots that are characterized by lightweight structures and enhanced mechanical strength. The progress has empowered legged robots to execute dynamic motions, and researchers have drawn inspiration from the natural world to expand locomotion capabilities. This pursuit has led to investigations into a wide range of behaviors, including jumps, flips, barrel rolls, and various other forms of aerial motion.*

In this chapter, our objective is to comprehensively explore the significant advancements on aerial motions in legged robotics, particularly focusing on quadrupeds. Drawing upon the Newton-Euler equations embedded within the full dynamical model, we aim to investigate the admissible motions of a falling robot or any other articulated system. Finally, we will introduce some essential concepts crucial for subsequent chapters.

## 2.1 State-of-the-Art on Robotic Aerial Motions

A multitude of legged animals, such as red kangaroos [Bennett and Taylor, 1995], spiders [Nabawy et al., 2018] and squirrels [Hunt et al., 2021], demonstrate the ability to modulate mechanical output at individual joints, allowing them to execute agile jumps and traverse their environments rapidly. Additionally, several animals, like cats [Kane and Scher, 1969] and lizards [Jusufi et al., 2011], exhibit remarkable aerial righting reflexes, reorienting themselves dorso-ventrally when falling upside down to position their limbs toward the ground. The translation of these natural capabilities to articulated systems is a longstanding and multifaceted research topic, encompassing mathematical formulations, hardware design, control strategies, and computational efficiency. While extensive literature exists regarding single-leg hoppers, encompassing both control algorithms [Ding et al., 2021b; Shen et al., 2020; Yim et al., 2020] and leg and actuator designs [Arikawa and Mita, 2002; Ding and Park, 2017; Zhao et al., 2013], the implementation of these results on multi-leg robots capable of general locomotion behaviors is much less prevalent.

### 2.1.1 Planning Aerial Motions

The exploration of aerial motions for legged robots has its roots in the 1990s, marked by notable works like Raibert's pioneering research [Hodgins and Raibert, 1990], which showcased a biped performing somersaults using heuristic strategies. Other endeavors, such as [Semini et al., 2015], adopted heuristic approaches to accomplish dynamic tasks involving squat jumps and impactful maneuvers with the hydraulically actuated quadruped robot HyQ. Their strategies involved adjusting the parameters of an active impedance controller. Further advancements were made with automatic parameter fine-tuning in

a model-based, state-feedback controller, as illustrated in [Gehring et al., 2016]. These developments optimized various gait patterns, including squat jumps, running trot, pronking, and bounding gaits for the torque-controllable quadruped robot StarlETH, using sampling-based search. With the increased availability of higher computational power, approaches based on numerical optimization started to appear. In [Park et al., 2021], a set of algorithms was introduced enabling obstacle jumping for the MIT quadruped robot Cheetah 2. These algorithms employ an approach based on Model Predictive Control (MPC) for regulating the bounding velocity and positioning the robot for successful leaps. GRFs profiles parameterized using Bézier splines are used by the bounding controller introduced in [Park et al., 2014] and further investigated in [Park et al., 2017].

Numerous Trajectory Optimization (TO) methods have been devised to generate jumping motions based on *full-body dynamics*. Notably, works like [Nguyen and Nguyen, 2022; Nguyen et al., 2022, 2019] successfully produced such motions while considering the complete dynamics. However, these approaches require the pre-computation of the base pose motion trajectories. On the other hand, the off-line framework presented in [Song et al., 2022] does not need prior knowledge of either reference motion or contact schedule, but it relies on a time-consuming evolutionary search. For enhancing computational efficiency, and thus for being able to plan on-line reference trajectories, lower dimensional models can be used. A common approach relies on the Single Rigid Body Dynamics (SRBD) [Chignoli and Kim, 2021; Ding et al., 2019, 2021a] or on Centroidal Dynamics (CD) [Chignoli et al., 2022; Ding et al., 2020]. Remarkably, a rich set of reference trajectories for aerial motions, encompassing jumping onto and off platforms, spins, flips, barrel rolls, and running jumps over obstacles, was successfully executed by the MIT Mini Cheetah in [Chignoli and Kim, 2021; Chignoli et al., 2022; García et al., 2021]. Motion planning was achieved through the optimization of the centroidal momentum, and the tracking has been made possible by the variational-based optimal controller as presented in [Chignoli and Wensing, 2020]. Moreover, this controller demonstrated the capability to manage underactuated contact configurations effectively. [Ding et al., 2020] uses the parameterization of contact forces with Bézier splines for kino-dynamic optimization motion planning of aggressive jumps while considering joint torque constraints. The concept of the Feasible Wrench Polytope (FWP) [Orsolino et al., 2018] was adopted to address the complexity of torque constraints. Another common choice for capturing the

dominant dynamics is the so-called Spring-Loaded Inverted Pendulum (SLIP) model [Blickhan, 1989]. In its original version, it consists of a point mass attached to a massless spring whose force is governed by Hooke's law. The model has been profusely used to describe bipedal running and hopping. However, the assumptions of constant spring stiffness and fixed rest length limit its application to highly dynamic locomotion, such as high-speed running and jumping [Xiong and Ames, 2018]. To overcome this issue, researchers introduced some variations, e.g., variable rest length [Rezazadeh et al., 2015; Schmitt and Clark, 2009], variable spring constant [Liu et al., 2015], or different spring force laws [Poulakakis and Grizzle, 2009]

We cannot conclude this roundup about planning techniques for aerial motions without mentioning blossoming alternatives to model-based controllers. In particular, Reinforcement Learning (RL) is drawing the attention of the robotic community. While normal gaits like trotting can be learned from scratch, the challenges become more pronounced when dealing with acrobatic maneuvers where tedious reward shaping is necessary [Hwangbo et al., 2019; Rudin et al., 2022]. This is primarily due to the highly nonlinear dynamics and reward sparsity. Recent RL-based frameworks often assume the presence of prior knowledge, manifesting either as a reference motion or trajectory [Bussola et al., 2023; Fuchioka et al., 2023; Li et al., 2023], or pre-existing controllers [Yang et al., 2023]. In [Bellegarda and Nguyen, 2020], deep RL improves the robustness for leaps on significantly uneven terrain with variable robot dynamical parameters. The resulting controller is able to track many different jumping trajectories and works with different joint gains, which can save human trial-and-error associated with gain tuning. Finally, there are some data-driven algorithms from the graphics community, such as DeepLoco [Peng et al., 2017], DeepMimic [Peng et al., 2018], and Allstep [Xie et al., 2020], which have demonstrated impressive locomotion capabilities either with humanoid [Radosavovic et al., 2023] and quadruped robots [Peng et al., 2020]. Although many of these are in simulated environments, data-driven algorithms are increasingly being applied in actual robot control.

### 2.1.2 Exploring Inertial Effects

When an articulated system is in mid-air, its CoM is constrained to move on a ballistic trajectory, defined by the lift-off position and velocity. On the other hand, the base orientation can be changed by exploiting the conservation

of angular momentum law. Differently saying, it is possible to control the orientation of a robot when it falls by changing its inertia, e.g., changing the relative position of the bodies composing it.

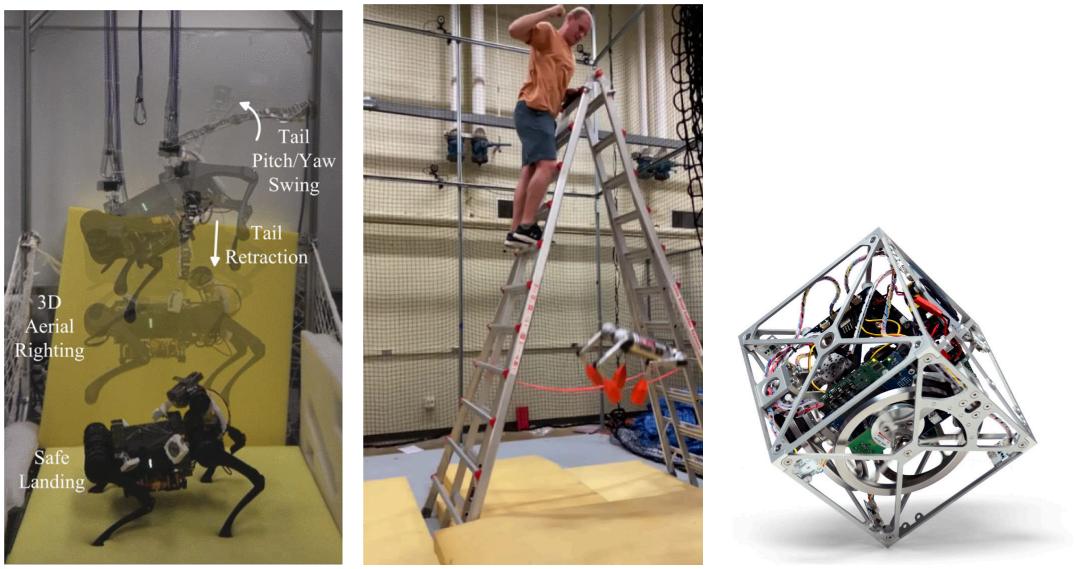
A quadrupedal animal, like a cat, can correct its orientations during a fall by changing the relative position of limbs and tail with respect to the trunk and also by rotating its spine. However, it is worth noting that most quadruped robots are typically equipped with lightweight limbs. This design choice simplifies locomotion planning and control by making use of what is known as the *massless leg assumption* [Raibert, 1986]. Consequently, the resulting limbs have a limited impact on the overall angular momentum of the system.

### Tail

Much research has explored the use of an additional link designed to mimic the behavior of a tail, as exemplified in [Chu et al., 2019; Wenger et al., 2016]. Such a link rotates around an axis that does not pass through the robot's CoM. The distances between the axis of rotation of the tail and the CoMs of both the trunk and the tail have a large effect on the total inertia, even with a small tail mass. Inspired by cheetahs, [Patel and Braae, 2013] designed and proved that a tail-like mechanism enhances the stability of a wheeled platform during high-speed turns. Nevertheless, the placement of the additional link makes the resulting robot asymmetric. To enhance agile locomotion and safety in landing, a morphable inertial tail with 3-Degree of Freedoms (DoFs) (pitch, yaw, and telescoping) was introduced and attached on a monopod [An et al., 2022], a biped [An et al., 2020], and a quadruped [Tang et al., 2023] (see Figure 2.1a). The limitations of utilizing tails stem from their restricted range of motion, which confines their application to a single jump rather than enabling a repeated sequence [Johnson et al., 2012].

### Limbs

Given that limbs have a not negligible inertia if compared with the trunk's one, it is possible to obtain a similar result by creating repetitive circular motions with the feet, like in [Mingo Hoffman and Paolillo, 2021]. By extending the feet outward, the robot's overall inertia increases. Therefore, if one leg is extended during one phase of the motion and retracted during another, it gives rise to a net angular momentum on the trunk, leading to a rotational effect. In [Kurtz et al., 2022] Mini Cheetah was provided with heavy boots to improve




(a) (b) (c)

**Figure 2.1:** Examples of hardware modifications for orientation control.
(a) A combined motion snapshot of the Unitree A1 performing aerial righting and landing by utilizing a tail in a fall from 1 m height onto a soft mattress. ©2023 IEEE [Tang et al., 2023].
(b) The 9 kg Mini Cheetah [Katz et al., 2019] performing ventral reorientation and landing on a soft mat pile by leveraging four heavy boots (0.5 kg each). ©2022 IEEE [Kurtz et al., 2022].
(c) Cubli is a $15 \times 15 \times 15$ cm cube that can jump up and balance on an edge or a corner. Flywheels mounted on three faces of the cube rotate at high angular velocities and then brake suddenly, causing the Cubli to jump up. ©2012 IEEE [Gajamohan et al., 2012].

the influence on torso rotation, see Figure 2.1b. Purely vertical falls with large rotations on the sagittal plane are handled by a combination of off-line optimization and supervised machine learning. For many initial conditions, they solve TO problems and use a Neural Network (NN) to *memorize* the results. At run time, the network outputs a trajectory that lands the robot on its feet. The drawback of increasing the limbs' inertia is that the robot becomes tailored for the specific task of landing since planning problems can no longer rely on the assumption of massless legs.

### Gyroscope

An alternative approach involves the utilization of a Control Moment Gyroscope (CMG). This device comprises a spinning wheel with a constant angular velocity, encased within two or three controllable gimbals. By adjusting the orientation of the wheel's axis of rotation, gyroscopic torque is generated. Although the CMG system is commonly employed for spacecraft reorientation purposes [Yoon and Tsiotras, 2002], its application in robot locomotion, whether in wheeled [Brown and Xu, 1996] or legged systems [Mikhalkov et al., 2021], has been relatively rare. While it offers intriguing capabilities, due to its mechanical complexity, the inclusion of a pan-tilt unit to drive the gyroscope renders it impractical for installation on small, lightweight robots.

### Flywheels

Flywheels represent an alternative means to change the robot's orientation when no contact is active. Applying torque to a rotating mass connected to the robot's body results in accelerating the robot's body rotation. Due to the conservation of angular momentum, this acceleration causes the base to accelerate in the opposite direction, akin to the effect of torque applied directly to the base. The use of flywheels for orientation control initially found application in spacecraft maneuvering [Oland and Schlanbusch, 2009]. In the realm of legged locomotion, it has been sporadically explored, primarily concerning pitch orientation, on biped [Brown and Schmiedeler, 2016; Xiong and Ames, 2020] and quadruped [Vasilopoulos et al., 2016] locomotion. For instance, the quadruped robot SpaceBok developed by ETH was equipped with a reaction wheel [Kolvenbach et al., 2019], allowing effective pitch angle control. This mechanism serves to drive pitch errors to zero during the flight phases. Achieving a rapid response necessitates abrupt changes in the flywheel's angular velocity (angular acceleration). Employing a brake mechanism eliminates the requirement for a motor capable of delivering high torques, maintaining a compact system [Gajamohan et al., 2012]. The motor gradually accelerates the wheel to a specific speed to store angular momentum. When reorientation is needed, the brake halts the wheel's rotation. However, it is important to note that this braking effect is unidirectional, restricting the approach to generating base rotations in the direction opposite to the flywheel's angular velocity. Consequently, this method may not be suitable for applications requiring continuous controllability. On the other hand, direct-drive controlled flywheels

are capable of producing accelerations in both directions and accommodate continuous control strategies. The disadvantage of the flywheels lies in their ability to generate a torque acting on the robot base for a limited time, i.e., until their actuators reach the maximum spin rate.

### 2.1.3 Safe and Stable Landing

Stable landing for quadruped robots has been achieved employing low gain feedback control [Nguyen and Nguyen, 2022; Nguyen et al., 2019; Song et al., 2022], MPC [Di Carlo et al., 2018; Neunert et al., 2018; Nguyen et al., 2022], or impulse modulation [Park et al., 2021]. However, tracking control alone may not provide enough robustness when dealing with significant dynamic uncertainties, such as deviations caused by imperfect LO states, uncertainties related to landing surfaces, or high speed. If the controller fails to precisely track the reference trajectory for the thrusting phase, it can lead to discrepancies between the actual and planned flying phase, e.g., different duration or displacement, potentially resulting in an unfeasible landing trajectory. Additionally, these approaches necessitate the pre-computation of entire reference trajectories, which are contingent on the specific LO conditions. For scenarios requiring real-time optimization, obtaining such data may not be feasible if the robot operates continuously and must execute jumps without stopping the motion. Addressing these challenges necessitates the implementation of real-time controllers for both in-air motion strategy and reaction upon TD detection. In the previous section, we explored various works focusing on posture adjustment using inertial shaping, ignoring the importance of online leg adjustment in reaching desired landing positions. [Bingham et al., 2014] investigated falls from heights, considering a combination of rolling and impact absorption through a landing technique called *soft roll*. In this context, a soft roll refers to a posture that optimizes rolling while enabling the system to act as a damped spring-mass system. This approach shares similarities with [Ha et al., 2012]'s study on simulating the physical aspects of falling and landing motions for animated characters. Using a deep RL algorithm and performing sim-to-real transfer via domain randomization, Rudin et al. show Spacebok performing consecutive 2D jumps in a micro-gravity environment [Rudin et al., 2021]. Even if the method is tailored to uneven terrains, tests are conducted on flat and rigid ground. The motion capture system Vicon provides measures for the robot's absolute position and orientation. How to extend the results

to real-world 3D conditions with more dramatic impacts and inertial effects is unclear. [Qi et al., 2022] addresses the problem of landing on celestial bodies too. It proposes a model-free RL controller with an auto-tuning reward function to address attitude and landing control on Near-Earth asteroids for which gravity cannot be supposed to be a central force. In [Bledt, 2020], MIT Mini Cheetah is dropped with a CoM height of 1 m above the ground. The heuristics-guided MPC allows the robot to push onto the ground enough to arrest its downward momentum and return to a stable standing state within two steps. A recent work [Zhang et al., 2023] implements heuristics to shift the feet posture for landing based on physical intuitions. However, the approach lacks feasibility guarantees grounded on a model of the system. In [Jeon et al., 2022], a complete control framework can select optimal contact locations and timings in real-time. The controller consists of a NN providing warm-start trajectories to a kino-dynamic TO problem, solved on-line in an MPC fashion. The robot safely recovers from drops with different orientations, but involving only *vertical* falls. The bottleneck of this framework is the NN: if a solution is not found within 300 ms, a new request is made and the optimization restarts. However, such a time span represents an extensive flying phase: it corresponds to a fall from a height of about 0.5 m, starting with zero velocity. Therefore, falls from reduced heights may not be addressed with this approach.

Alternatively, other impact-aware controllers could be applied to the landing task by modeling contact dynamics. [Lynch et al., 2020] captured the plastic nature of the terrain deformation upon contacts and established a linear dependence between GRFs and feet penetration depth. For this terrain model, it derived properties of optimal force control for depth minimization, subject to force and stroke limits, developing force and impedance control solutions to soft landing problem. [Li and Wensing, 2020] proposed a variation of Differential Dynamic Program (DDP) to address the discontinuity at impacts of legged locomotion by incorporating an impact-aware value function update in the backward sweep. Nonetheless, obtaining a precise contact model poses a formidable challenge, particularly in the context of quadrupedal landings. This challenge is amplified when one accounts for multi-contact scenarios, involving ground surfaces of unknown height, inclination, and varying degrees of unevenness.

## 2.2 Background on Allowed Motions for an Articulated System

In this section, we are going to understand the effects of joint torques, gravity, and contact forces on the motions of an articulated system by relying on the Newton-Euler equations embedded in the full dynamical model. The discussion continues by recalling the concepts of Center of Pressure (CoP), viability, and capturability.

A quadruped robot is a free-floating system having $n_j$ articulated joints connecting a set of rigid bodies. Specifically, we consider robots having three joints on each leg, namely Hip Abduction/Adduction (HAA), Hip Flexion/Extension (HFE), Knee Flexion/Extension (KFE). Without considering undesired cases, e.g., involuntary crashes, we assume that GRFs are exerted by the environment only onto limb appendages, here supposed to be point feet. In an inertial reference frame, the full dynamical model of an articulated system having $n_c = 0, 1, \ldots, 4$ contact points is the following set of $n_j + 6$ second order differential equations:

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{S}^T\boldsymbol{\tau} + \sum_{k=1}^{n_c} \boldsymbol{J}_{pk}^T(\boldsymbol{q})\,\boldsymbol{f}_k, \tag{2.1}$$

where $\boldsymbol{q} \in \mathbb{R}^{n_j} \times \mathrm{SE}(3)$ is the vector of generalized coordinates that encompass the joint configuration $\hat{\boldsymbol{q}} \in \mathbb{R}^{n_j}$, the base (or trunk) position $\boldsymbol{p}_b \in \mathbb{R}^3$ and orientation $\boldsymbol{\phi}_b \in \mathbb{R}^3$, representing an element of SO(3). Moreover, we use $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{(n_j+6)\times(n_j+6)}$ to denote the mass matrix, $\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} \in \mathbb{R}^{n_j+6}$ contains the dynamical effects due to Coriolis and centrifugal forces, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^{n_j+6}$ contains the dynamical effects due to the gravity, $\boldsymbol{S} \in \mathbb{R}^{(n_j+6)\times n_j}$ is the selection matrix, $\boldsymbol{\tau} \in \mathbb{R}^{n_j}$ is the vector of joint torque, $\boldsymbol{J}_{pk}(\boldsymbol{q}) \in \mathbb{R}^{3\times(n_j+6)}$ is the translational Jacobian matrix of the $k$-th contact point and $\boldsymbol{f}_k \in \mathbb{R}^3$ is the GRF exerted on it.

The full body dynamics (2.1) embeds the Newton-Euler equations describing the linear and angular dynamics of the whole system (for the derivation of the Newton-Euler equations from the full body dynamics, consult Appendix B). The Newton equation describes the dynamics of the system's CoM and can be written as

$$m(\ddot{\boldsymbol{c}} + \boldsymbol{g}) = \sum_{k=1}^{n_c} \boldsymbol{f}_k \tag{2.2}$$

where $m \in \mathbb{R}^+$ is the total mass of the robot, $\boldsymbol{c} \in \mathbb{R}^3$ is the position of its CoM and $-\boldsymbol{g} \in \mathbb{R}^3$ is the gravity acceleration vector. The Euler equation, defining the dynamics of the main body orientation, can be expressed with respect to the robot's CoM as

$$\dot{\boldsymbol{L}} = \sum_{k=1}^{n_c} [\boldsymbol{p}_k - \boldsymbol{c}]_\times \boldsymbol{f}_k \tag{2.3}$$

being $\boldsymbol{p}_k \in \mathbb{R}^3$ the point of application of $\boldsymbol{f}_k$, i.e., the foot location, and $[\cdot]_\times$ is the skew-symmetric operator associated to the vector product (cf. (B.7)). $\boldsymbol{L} \in \mathbb{R}^3$ is the angular momentum of the whole system with respect to $\boldsymbol{c}$:

$$\boldsymbol{L} = \sum_j [\boldsymbol{p}_j - \boldsymbol{c}]_\times m_j \dot{\boldsymbol{p}}_j + \boldsymbol{I}_j \boldsymbol{\omega}_j \tag{2.4}$$

where the sum is over all the rigid bodies composing the robot. Above, $\boldsymbol{p}_j$ is the position of the link frame of the $j$-th body, $\dot{\boldsymbol{p}}_j$ and $\boldsymbol{\omega}_j$ are its linear and angular velocities, $m_j$ and $\boldsymbol{I}_j$ are its mass and inertia matrix (expressed in inertial frame coordinates). We are going to explore the dynamical constraints that restrict the pose of an articulated system. Initially, we focus on the impact of joint torques alone. Subsequently, we incorporate the influence of gravitational forces and introduce the effects of contact forces.

### Only Joint Torques

When an articulated system is solely influenced by the action of the torque inputs $\boldsymbol{\tau}$, both linear and angular momenta remain conserved:

$$m\dot{\boldsymbol{c}} = \text{constant}, \tag{2.5}$$

$$\boldsymbol{L} = \text{constant}. \tag{2.6}$$

If the system has zero velocity, both the constants are zero, and from (2.5) follows that regardless of the joint torques, the system's CoM remains stationary. Equation (2.6) imposes a nonholonomic constraint, i.e., a relationship among the velocities of the bodies that does not reflect itself on a restriction of the positions the system can reach (see Appendix C for further details). We can state that the system's joint configuration $\hat{\boldsymbol{q}}$ and its orientation $\boldsymbol{\phi}_b$ can be cohesively controlled to reach any desired values solely by applying the joint torques $\boldsymbol{\tau}$. As a matter of fact, the joints torque $\boldsymbol{\tau}$ impose changes on the joints position $\hat{\boldsymbol{q}}$, which in turn change the body position $\boldsymbol{p}_j$ and the latter imposes modification on the body angular velocity $\boldsymbol{\omega}_j$. Because of the con-

servation of the angular momentum, also the base angular velocity changes, which ultimately modifies the base orientation $\boldsymbol{\phi}_b$. In other words, the control of the system's articulations extends its influence to regulate the orientation of the system as a whole.

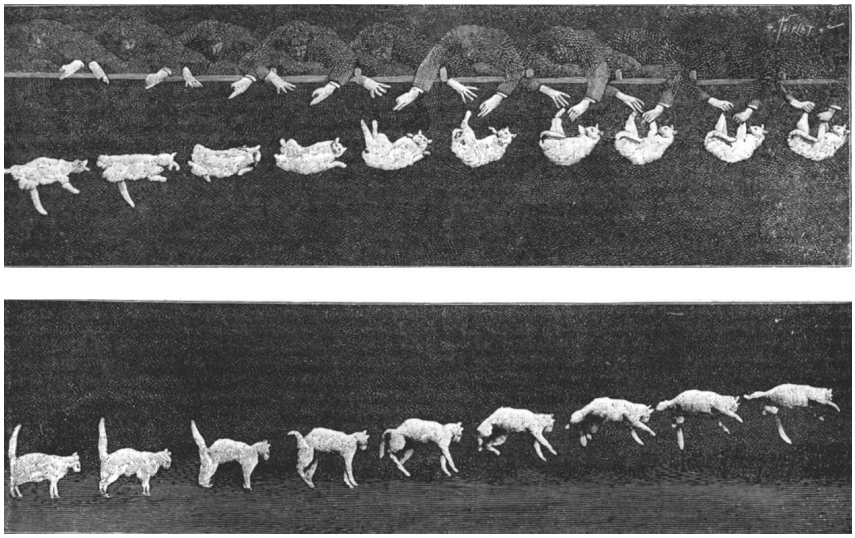
### Including Gravity

We have just observed the behavior of an articulated system when under the action of joint torques only, as though it were floating without gravity. After a quick inspection of the Newton-Euler equation, we can state that when the system is also subject to the gravity, the unique alteration in its motion occurs at its CoM: it undergoes a linear acceleration along the direction of the gravity vector instead of remaining stationary,

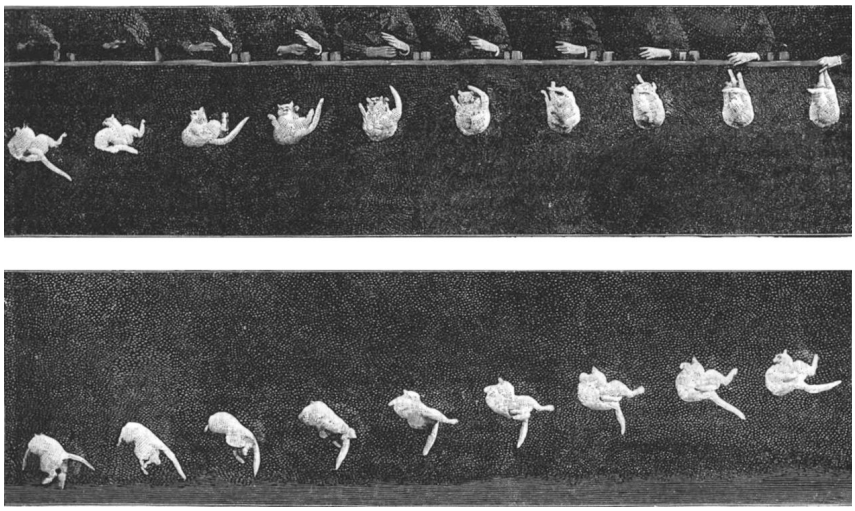
$$\ddot{\boldsymbol{c}} = -\boldsymbol{g}.$$

The conservation of angular momentum is not affected by the action of gravity, nor is the nonholonomic constraint it imposes. Therefore, when an articulated system does not have any contact with the surrounding environment (during a flying phase), its CoM moves along a ballistic trajectory, i.e., with constant vertical acceleration. Meanwhile, its base orientation can be changed by modifying the distribution of the masses, e.g., stretching the limbs, moving a tail, or activating a reaction wheel.

The phenomenon becomes evident in a classical example involving an articulated system influenced by both gravity and joint torques: the *falling cat problem* [Kane and Scher, 1969; Marey, 1894]. Remarkably, cats (but also other legged animals) in free fall consistently exhibit the ability to reorient themselves dorso-ventrally and land on their feet. The series of stop-motion sequences captured in Figure 2.2a and 2.2b depict the animal experiencing a free fall and tumbling through the air. Despite maintaining a constant angular momentum, it engages in a rotation as it swings its legs and tail back and forth. These appendages exhibit varying lengths and, consequently, differing inertial properties, which contribute to turning upright the cat that skillfully lands on its feet.



(a) Side view.



(b) Back view.

**Figure 2.2:** Étienne-Jules Marey's Analysis of a Falling Cat [Marey, 1894] (the series run from right to left, and the lower is the continuation of the upper). These images, captured at 12 frames per second using Marey's chronophotographic gun, document a falling cat's descent and self-righting maneuvers. Marey's observations led him to rule out hypotheses involving either the animal using the interface between its paws and the hands that let it fall as a fulcrum or the resistance of air for rotational motion. Instead, he highlighted the cat's strategic use of mass inertia and limb positioning to pivot to an upright position during the fall. Published in 1894, the images elicited a humorous response from the editor "The expression of offended dignity shown by the cat at the end of the first series indicates a want of interest in scientific investigation." (Reproduced with permission from Springer Nature. Licence number: 5655991358663. ©Nature)

### Including Contact Forces

The preceding analyses shed light on the limitations of joint torques and gravity in moving the system's CoM. Their influence is restricted to a motion along the gravity vector field. The locomotion of articulated systems is thus contingent on the presence of GRFs, which are limited by unilateral and frictional constraints.

## 2.2.1 Center of Pressure

Both full dynamical model in (2.1) and the Newton-Euler equations in (2.2)-(2.3) are quite general and can be specialized to the case of interest under some assumptions.

**Assumption 2.1.** *The ground surface is flat.*

Without loss of generality, suppose that the fixed inertial frame has its $z$-axis orthogonal to the ground and the $xy$-plane lies on it. Pre-multiplying (2.2) by $[\boldsymbol{c}]_\times$ and summing with (2.3), we can condense the pose dynamics into:

$$m\,[\boldsymbol{c}]_\times\,(\ddot{\boldsymbol{c}}+\boldsymbol{g})+\dot{\boldsymbol{L}}=\sum_k [\boldsymbol{p}_k]_\times\,\boldsymbol{f}_k. \tag{2.7}$$

Let us divide the result by the $z$-coordinate of (2.2) (that is expressed in the inertial frame):

$$\frac{m\,[\boldsymbol{c}]_\times\,(\ddot{\boldsymbol{c}}+\boldsymbol{g})+\dot{\boldsymbol{L}}}{m\,(\ddot{c}^z+g^z)}=\frac{\sum_k [\boldsymbol{p}_k]_\times\,\boldsymbol{f}_k}{\sum_k f_k^z,}. \tag{2.8}$$

Because of the choice of the inertial frame, we have $p_k^z = 0$ for the feet in contact with the ground. Therefore, the $x$ and $y$ coordinates of (2.8) simplify as is expressed in the inertial frame):

$$\boldsymbol{c}^{x,y}-\frac{c^z}{\ddot{c}^z+g^z}\,(\ddot{\boldsymbol{c}}^{x,y}+\boldsymbol{g}^{x,y})+\frac{1}{\ddot{c}^z+g^z}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\dot{\boldsymbol{L}}^{x,y}=\frac{\sum_k f_k^z\,\boldsymbol{p}_k^{x,y}}{\sum_k f_k^z}. \tag{2.9}$$

The Center of Pressure (CoP), denoted as $\boldsymbol{u}\in\mathbb{R}^3$, is a point lying on the ground at which the net GRF acts

$$\boldsymbol{u}^{x,y}\triangleq\frac{\sum_k f_k^z\,\boldsymbol{p}_k^{x,y}}{\sum_k f_k^z}. \tag{2.10}$$

As previously introduced, GRFs are subject to unilateral constraints. Consequently, in order to adhere to these constraints, CoP is required to remain within the Support Polygon (SP), that is the convex hull of the contact points:

$$\boldsymbol{u}^{x,y} \in \text{convHull}\left(\boldsymbol{p}_k^{x,y},\, k = 1,\, \ldots,\, n_c\right). \tag{2.11}$$

Being $\boldsymbol{u}^{x,y}$ outside the Support Polygon (SP) would mean that some $f_k^z < 0$, i.e., one of the GRFs is violating the unilaterality restriction.

Notice that, reorganizing (2.10), another interpretation of the point $\boldsymbol{u}$ can be expressed. The horizontal momenta of the external forces with respect to the CoP equal zero

$$\left(\sum_k \left[\boldsymbol{p}_k - \boldsymbol{u}\right]_\times \boldsymbol{f}_k\right)^{x,y} = \sum_k \left(\boldsymbol{p}_k^{x,y} - \boldsymbol{u}^{x,y}\right) f_k^z = \boldsymbol{0}.$$

Therefore, the CoP is also commonly known as the Zero Moment Point (ZMP). When the CoP is within the SP, it ensures that the angular moments (computed about this point) applied by the GRFs about any point within the SP are balanced and sum to zero. This prevents the system from rotating or tipping over. Summarizing, an articulated system described by the Newton-Euler equations (2.2)–(2.3) is dynamically stable if its CoP is confined within its SP. Notice that the inclusion of the CoP in the SP is only a sufficient condition for stability.

### 2.2.2 Viability and Capturability

We conclude this chapter by reporting the theoretical concepts that will be essential for ensuring the stability of robots during their landing phase: *viability* and *capturability*.

If the primary objective of a legged system is to avoid losing balance, especially during the landing phase, we can categorize all the postures where the robot is going to fall down into one subset of postures, and simply avoid them. We call such a subset *failed states*. The remaining set of postures is known as viable states, as defined in [Aubin, 2009]. Viable states represent a subset of the state space in which the legged system must remain to avoid losing balance. In theory, viable states define the safe operational region for the robot. In practice, explicit computation of these states is often exceedingly complex.

Rather than attempting to analyze the entire set of viable states, we can narrow our focus to a more specific collection of states capable of achieving a stable stop within a predetermined number of steps. These states are referred to as capturable states, following the terminology in [Koolen et al., 2012], and naturally, they constitute an interesting subset of the viable states. To put it differently, a capturable state is a posture in which the kinetic energy of the legged robot is zero and can be maintained at zero with the application of suitable joint torques.

A state evolution through the system's state space is defined as a safe feasible trajectory if it adheres to the robot's dynamics, is attainable by the robot's actuators, and exclusively consists of viable states. In this context, researchers introduced the concept of Capture Point (CP), which is a specific point on the ground. If the robot covers this Capture Point (CP), meaning it makes its SP encompass this point, and then maintains its CoP on it, there exists a safe feasible trajectory leading to a capturable state. In the general case, several points exist on the ground satisfying this condition. Therefore, there exists a Capture Region such that if the CoP is placed inside this region, then the articulated system can stop for some state space trajectory.

While it is difficult to compute CP using the full dynamical model, we can easily compute them in closed form for some simplified descriptions. In the next section, we report the exact closed-form solutions of the Capture Region for the Linear Inverted Pendulum (LIP) model.

### Capture Points for the Linear Inverted Pendulum

The Linear Inverted Pendulum (LIP) model is a simplified dynamic model used to describe the motion of a legged robot during the stance phase of walking, running, or other forms of locomotion. This model assumes that during each step, the robot's CoM moves as if it were a point mass atop a massless rod (the inverted pendulum). The CoM moves along a straight line trajectory similar to the motion of a pogo stick as it hops up and down, maintaining a fixed height, see Figure 2.3. LIP model has undergone extensive examination concerning both motion generation and stability analysis. It relies on Assumption 2.1 and on the following statements:

1. the ground surface is orthogonal to the gravity;
2. the variation of angular momentum is negligible;
3. the CoM height is constant.

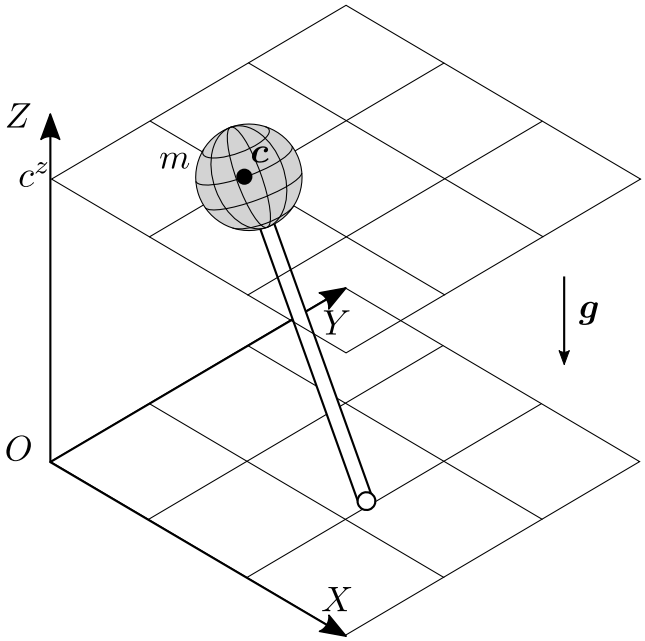



**Figure 2.3:** The 3D LIP model. This figure is inspired by figure 3 of [Koolen et al., 2012].

One can derive the LIP equations of motion by including the above assumptions into (2.9):

$$\ddot{\boldsymbol{c}}^{x,y} = \omega^2 \left(\boldsymbol{c}^{x,y} - \boldsymbol{u}^{x,y}\right) \tag{2.12}$$

having set

$$\omega^2 = \frac{g^z}{c^z}. \tag{2.13}$$

The above second order system is linear, and the two horizontal directions are decoupled. The assumptions for deriving the LIP model are extremely restrictive, and thus, it is not possible to embed it into a landing framework. Nevertheless, the LIP model is useful for understanding the concept of capturability. The set of capturable states can be identified analytically with the compound variable

$$\boldsymbol{\xi}^{x,y} = \boldsymbol{c}^{x,y} + \frac{1}{\omega}\dot{\boldsymbol{c}}^{x,y}. \tag{2.14}$$

This has been introduced with different names that highlight its different properties.

- eXtrapolated Center of Mass (XCoM) [Hof et al., 2005]. Rearranging

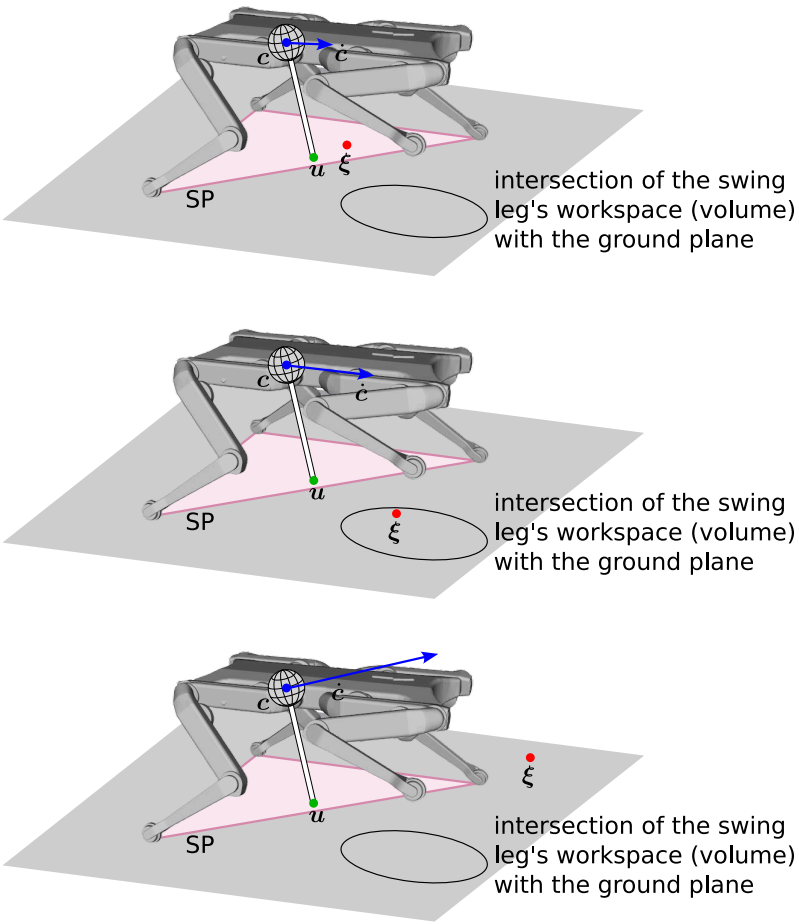



**Figure 2.4:** Top: when the CP $\boldsymbol{\xi}$ is included in the SP, a legged system can modulate its CoP $\boldsymbol{u}$ to balance and does not need to take a step. Center: when the CP does not fall inside the SP, the robot must take a step to come to a stop. Bottom: if the CP is outside the workspace of the swing foot, there is no possibility of stabilizing the system in one step. This figure is inspired by figure 2 of [Pratt et al., 2006].

(2.14), we identify a stable first order linear system

$$\dot{\boldsymbol{c}}^{x,y} = \omega \left( -\boldsymbol{c}^{x,y} + \boldsymbol{\xi}^{x,y} \right) \tag{2.15}$$

which shows that the CoM is converging to $\boldsymbol{\xi}$.

- Capture Point (CP) [Pratt et al., 2006]. Differentiating (2.14) with respect to time and making use of (2.12) yields

$$\dot{\boldsymbol{\xi}}^{x,y} = \omega \left( \boldsymbol{\xi}^{x,y} - \boldsymbol{u}^{x,y} \right), \tag{2.16}$$

i.e., the point $\boldsymbol{\xi}$ diverges away from the CoP. If a legged system is assumed to behave like a LIP, the resultant of the GRFs is applied on the CoP. Therefore, we can prevent the divergence of the CP by overlapping it with the CoP, i.e.,

$$\boldsymbol{\xi}^{x,y} \in \text{convHull}\{\boldsymbol{p}_k^{x,y}\} \Rightarrow \boldsymbol{\xi}^{x,y} = \boldsymbol{u}^{x,y} \text{ and } \dot{\boldsymbol{\xi}}^{x,y} = \mathbf{0}.$$

It follows that the CoM will converge above it and eventually come to a stop, see Figure 2.4. From this fact, the word *capture*.

The relationship among CoM, CP and CoP is shown in Figure 2.5.

Another relevant description of capturable states for the LIP model comes as a byproduct of the first integral of the model dynamics (2.12). First integrals are functions that remain constant throughout the system's dynamic evolution, see Appendix D. It is named orbital energy and is a conserved quantity [Kajita and Tani, 1991]. Since in (2.12) the two directions are decoupled, let us focus only on the $x$–coordinate. The orbital energy is:

$$E_{LIP} = \frac{1}{2}(\dot{c}^x)^2 - \frac{g^z}{2c^z}(c^x)^2 \tag{2.17}$$

If the CoM is moving toward the CoP with $E_{LIP} > 0$, then orbital energy is sufficient to let the CoM pass over $\boldsymbol{u}$ and continue on its way. It can be seen physically that the CoM velocity around the pivot is large enough that

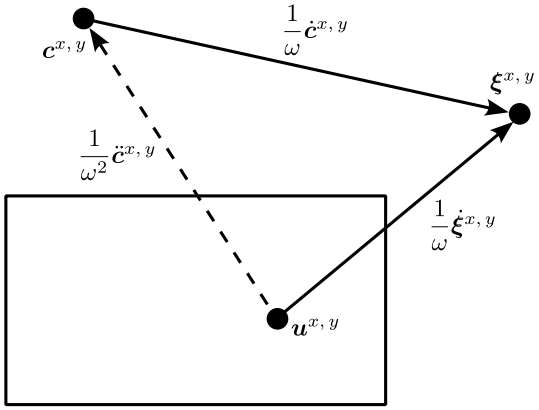



**Figure 2.5:** Relationship among CoM, CP and CoP. The velocity of the CoM $\boldsymbol{c}$ is directed toward the CP $\boldsymbol{\xi}$, whose velocity is directed away from the CoP $\boldsymbol{u}$. Combining the two first order dynamics yields the CoM acceleration, moving the CoM away from the CoP.

the deceleration induced by the moment of gravity (about the pivot foot) is not able to nullify it. If $E_{LIP} < 0$, then the CoM will stop and reverse the direction of motion before getting over the virtual foot. If $E_{LIP} = 0$, then the CoM converges to rest above $\boldsymbol{u}$, [Pratt et al., 2006]. The states such that $E_{LIP} = 0$ form the Capture Region.

The final message of this section is reported here. If the robot's state allows the CP to lie inside the SP, then the robot can stop without making any step, and it is capturable and viable. This point will play a crucial role in Chapters 4 and 5.

# 3

# Orientation Control System: Enhancing Aerial Maneuvers for Quadruped Robots

*For legged robots, aerial motions are the only option to overpass obstacles that cannot be circumvented with standard gaits. The chapter focuses on the limited controllability over the orientation during the flying phase. To gain control authority, I propose an Orientation Control System (OCS), consisting of two rotating masses (flywheels). Due to the conservation of angular momentum, the rotational velocity of the flywheels can be adjusted to steer the robot's orientation, even when the robot has no contact with the ground. The resulting device can control both roll and pitch angles, considering the robot's moments of inertia in the two directions. The concept was tested in simulations with the robot Solo12. A video with all the simulations is available at* https://youtu.be/Zad20mtFJm0?si=AbO5DZ-vqsLYxbMP.

*This work has been carried out supervising the master student Andrea Cumerlotti, who took care of the mechanical design of the OCS.*

Published. Journal Paper: Francesco Roscia, Andrea Cumerlotti, Andrea Del Prete, Claudio Semini, and Michele Focchi, "Orientation Control System: Enhancing Aerial Maneuvers for Quadruped Robots," in *MDPI Sensors*, vol. 23, no. 3, 1234, 2023, https://doi.org/10.3390/s23031234.

Conservation of angular momentum introduces a nonholonomic constraint in mechanical systems devoid of external forces, except for gravity's influence. This unique constraint serves as a pivotal element for effectively controlling the angular orientation of free-falling systems, such as a quadruped robot during its aerial maneuvers. In this chapter, we present a novel design for an innovative device referred to as the Orientation Control System (OCS). It is composed of two rotating masses strategically positioned to facilitate precise control over the orientation of legged robots. Notably, the rotational axes of the flywheels are configured to intersect, providing enhanced controllability on both roll and pitch axes, all while maintaining a compact and straightforward electronics and mechanical design. We employ direct-drive controlled flywheels, allowing for the generation of accelerations in both spinning directions. This system facilitates the implementation of continuous control laws.

We apply direct-drive controlled flywheels, which enable the generation of accelerations in both the spinning directions and the implementation of continuous control laws.

The advantages of employing our OCS are manifold:

1. Flywheels are capable of effectively correcting orientation errors encoun-

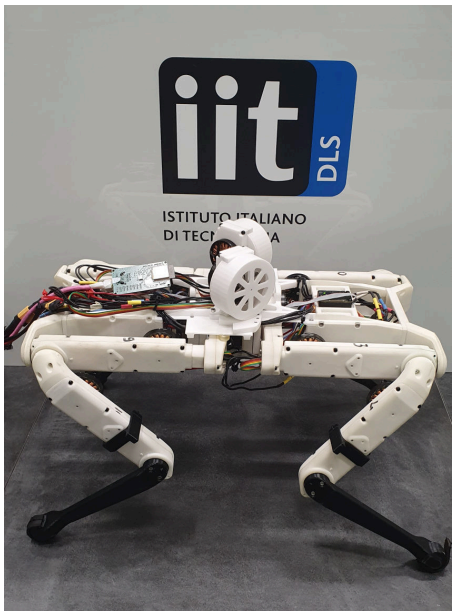



**Figure 3.1:** The proposed OCS for the 2.5 kg quadruped robot Solo12 consists of two 0.1 kg flywheels with incident rotation axes. Each wheel is located in a 3D-printed shell and mounted on the trunk of the body.

tered during the flying phase of aerial motions. These errors might arise from disturbances or tracking discrepancies in the angular momentum at the LO.

2. They can dynamically follow a time-varying reference, allowing the robot to land with a desired angular velocity and orientation, possibly aligning the trunk with the terrain.
3. The landing phase can be significantly improved by reducing trunk oscillations.
4. These additional actuators, employed solely for orientation control, have the potential to alleviate the workload on the legs. In complex scenarios, e.g., a somersault, limbs and OCS can operate in parallel to achieve a rotational angle larger than the one achievable only with legs due to torque and kinematic limitations.

The proposed OCS has been mounted on the trunk of the lightweight, torque-controlled quadruped Solo12 [Grimminger et al., 2020], as shown in Figure 3.1.

The remainder of this chapter is organized as follows. In Section 3.1, we revisit the law of conservation of the total angular momentum and conduct an exploratory inquiry about Solo12's inertia shaping. In Section 3.2, the designing principles of our OCS are presented, together with the strategy for simultaneous control of the robot's roll and pitch angles. Section Section 3.3 presents simulations in different scenarios that demonstrate the capability of rejecting disturbances and tracking angular references when there is no contact with the ground, and of dampening base oscillation after TD. Conclusions and possible evolution of the work are reported in Section 3.4.

## 3.1 Preliminaries

The starting point for any OCS is Euler's equation of motion. For any mechanical system, the time derivative of the angular momentum $\boldsymbol{L}$ computed with respect to a reference point $O$, fixed in an inertial frame, equals the sum of all the moments acting on the system, as previously shown in (2.3). When the result of the external moments about the robot CoM is zero, the Euler's equation simplifies to:

$$\dot{\boldsymbol{L}} = 0 \quad \Rightarrow \quad \boldsymbol{L}(t) = const, \tag{3.1}$$

which is known as the conservation of angular momentum. Referring to legged robots, this condition occurs when the only external force acting on the system is the gravity, i.e., it generates no moment about the CoM. For instance, when the robot is not in contact with the ground or with other objects, as it happens during a fall or the flying phase of a jump. In this case, it is possible to change the angular velocity of the base link, thereby changing the joint positions and velocities, as a result of the *non-holonomy* of the total angular momentum. If the angular momentum of some body changes, then one of the others must change to maintain the total sum constant.

The robot we use for validating our approach is Solo12. It was designed in such a way that the largest amount of the mass is located in the main body. Even if each limb accounts for about 13.4% of the total mass, changes in the total inertia of the robot come with the motion of only the upper leg (thigh) and the lower leg (calf) since the displacement of distribution of the hip mass with the system's CoM is approximately constant. In view of this, each limb is responsible for only 7.5% of the total mass in varying the total inertia of the robot. The feature of concentrating the majority of mass in the trunk is not unique to Solo12: it is a common trait shared by most quadruped robots. As an additional example, in the case of the Mini Cheetah, approximately 90% of its mass is situated within the trunk [Chignoli and Wensing, 2020]. This design aims to minimize limb inertia while maximizing the range of motion across all leg joints. As a drawback, the contribution of angular moments due to the acceleration of limbs is moderate. Flywheels can be used to alleviate the lack of control authority on the system's orientation.

## 3.2 Design the Orientation Control System

In this section, the procedure for selecting the inertia of the flywheels is presented, together with a strategy to exploit them so as to control the robot's roll and pitch orientations simultaneously.

### 3.2.1 Bounds on the Inertia

Investigation for the selection of the flywheels' inertia can be performed with Elroy's Beanie model, depicted in Figure 3.2. This is a 2D model consisting of two rigid bodies connected through their CoM with a revolute joint. One of the bodies represents the robot in its nominal configuration, and the other

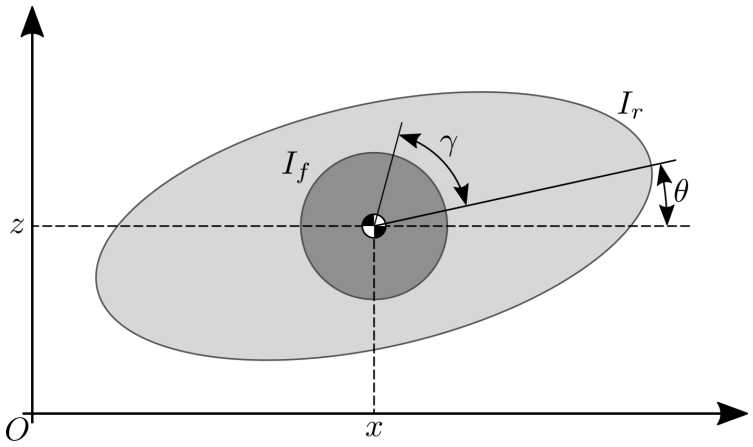



**Figure 3.2:** Schematic representation of the Elroy's Beanie model used for analyzing the effect of the flywheels on the robot orientation.

models the OCS, here represented by a single wheel for the sake of simplicity. The aim was to examine the rotational dynamics of the system as a whole. To simplify the analysis, in the following, only the effects on the pitch angle of the robot were considered, keeping in mind that the same arguments also apply to the roll. Let us identify the moment of inertia of the robot in the nominal configuration as $I_r$, and the moment of inertia of the two flywheels as $I_f$, both referred to the system's CoM. The angular momentum $L$ of this system can be written as:

$$L(t) = (I_r + I_f)\,\dot{\theta}(t) + I_f\dot{\gamma}(t) \tag{3.2}$$

where $\dot{\theta}$ and $\dot{\gamma}$ are, respectively, the robot's pitch rate for the trunk and the wheel's angular speed. The robot is driven to reach a desired pitch rate $\dot{\theta}_{des}$ by acting on the wheel speed. Without loss of generality, one can assume the flywheel is stationary with the robot at the instant at which the reorientation maneuver starts ($\dot{\gamma}_0 = 0$), having an initial system angular momentum of

$$L_0 = (I_r + I_f)\,\dot{\theta}_0.$$

Under the condition of conservation of the angular momentum, this quantity is constant over time, and it is possible to evaluate the lower bound for $I_f$, given a desired pitch rate and the maximum velocity of the flywheels $\dot{\gamma}_{max}$:

$$I_f \geq I_r \frac{\Delta\dot{\theta}}{\Delta\dot{\theta} + \dot{\gamma}_{max}}. \tag{3.3}$$

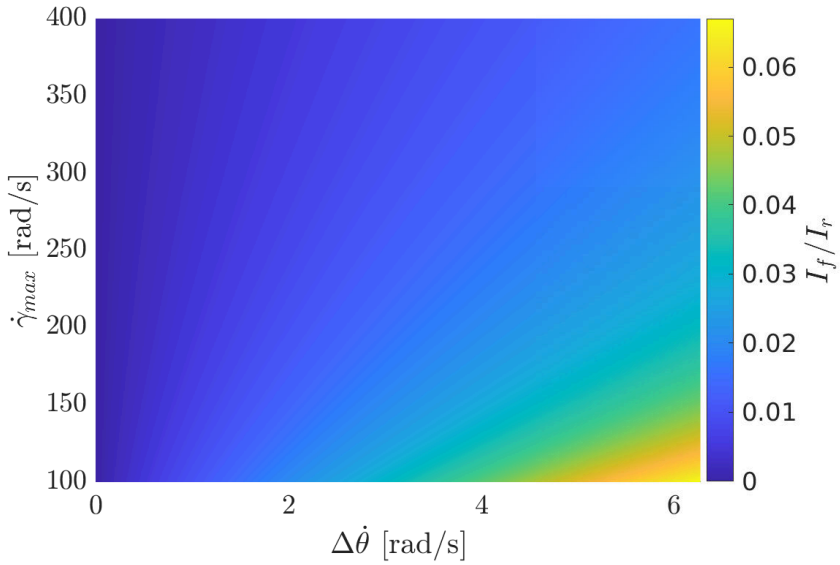



**Figure 3.3:** Lower bound for the flywheels' inertia (normalized by the robot's inertia) necessary to achieve a base velocity variation $\Delta\dot{\theta}$, given the speed bound $\dot{\gamma}_{max}$.

where $\Delta\dot{\theta} = \left|\dot{\theta}_0 - \dot{\theta}_{des}\right|$ is the base velocity variation. Figure 3.3 reports the lower bound of $I_f$ given the desired base velocity variation and the maximum velocity of the actuator.

### 3.2.2 Flywheels' Axes of Rotation

The direction of the axis of rotation for each flywheel carries significant implications for their respective contributions to the overall angular momentum. Notice that ensuring control over both the roll and pitch necessitates setting the axes of rotation for the left and right wheels not aligned. The axes of rotation, identified in the base reference frame with the unit vectors $\hat{\boldsymbol{a}}_l$ and $\hat{\boldsymbol{a}}_r$, are set to be incident, laying on a plane parallel to the $xy-$plane of the base reference frame, as visually depicted in Figure 3.4.

We have created two identical modules containing the flywheels to simplify the OCS design and avoid unwarranted complications. These modules are intended for mounting on Solo12's trunk in a symmetrical fashion, specifically with respect to the robot's sagittal plane. Let us denote $\alpha \leq \pi/2$ with the non-negative incident angle between the wheels' axes of rotation and the robot's

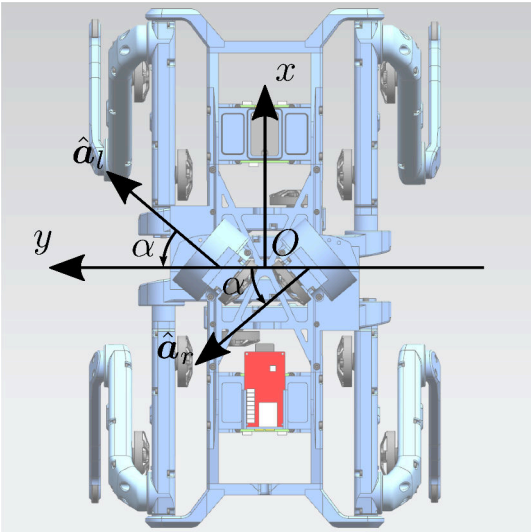



**Figure 3.4:** Top view of Solo12 with the proposed OCS. The $x-$ and $y-$axes embody the forward and left directions of the robot base, respectively. The figure also illustrates the unit vectors $\hat{\boldsymbol{a}}_l$ and $\hat{\boldsymbol{a}}_r$ that represent the axes of rotation of the left flywheel and of the right one and $\alpha$ is the angle of incidence, defined in Section 3.2.2.

lateral direction. The matrix:

$$\boldsymbol{C} = \begin{bmatrix} \hat{\boldsymbol{a}}_l & \hat{\boldsymbol{a}}_r \end{bmatrix} = \begin{bmatrix} \sin(\alpha) & -\sin(\alpha) \\ \cos(\alpha) & \cos(\alpha) \\ 0 & 0 \end{bmatrix} \tag{3.4}$$

maps the flywheel torque into moments $\boldsymbol{u} = \begin{bmatrix} \tau_{fl} & \tau_{fr} \end{bmatrix}^T$ about the flywheel axes $\hat{\boldsymbol{a}}_l$ and $\hat{\boldsymbol{a}}_r$. As long as $\boldsymbol{C}$ is in full column rank, it is possible to control both roll and pitch angles. If $\alpha = 0$, the roll becomes uncontrollable through the OCS; otherwise, if $\alpha = \pi/2$, the pitch becomes uncontrollable. The optimal angle is selected considering the ratio of the eigenvalues along the $x$ and $y$ directions of the ellipsoid of the robot's inertia (see Figure 3.5):

$$\alpha^* = \tan^{-1}\left(\frac{I_{r,xx}}{I_{r,yy}}\right). \tag{3.5}$$

Choosing the optimal angle allows us to jointly maximize the flywheels' contributions to the robot's roll and pitch. In the case of Solo12, this angle is about 40°. With these considerations in mind, the angular momentum produced by

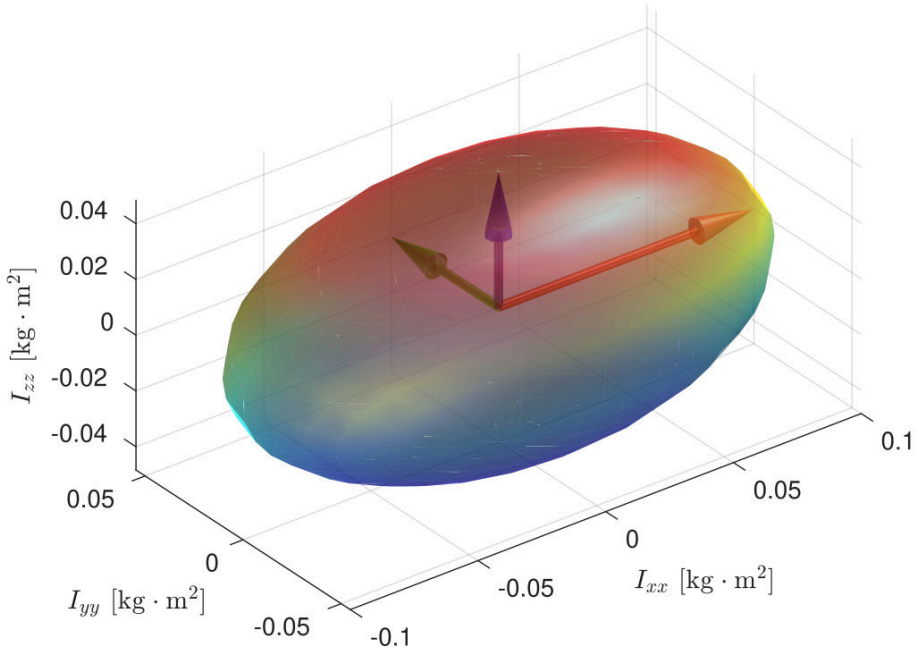



**Figure 3.5:** The inertia tensor of Solo12 seen as an ellipsoid. The directions of its axes are dictated by the eigenvectors of the tensor and the lengths depend on the eigenvalues.

the wheels, expressed in the robot base frame, is:

$$\begin{aligned} {}_b\boldsymbol{L}_f &= {}_b\boldsymbol{L}_{fl} + {}_b\boldsymbol{L}_{fr} \\ &= I_{f,zz}\,\boldsymbol{\omega}_{fl} + I_{f,zz}\,\boldsymbol{\omega}_{fr} \\ &= I_{f,zz} \begin{bmatrix} (\omega_{fl} - \omega_{fr})\sin(\alpha) \\ (\omega_{fl} + \omega_{fr})\cos(\alpha) \\ 0 \end{bmatrix} \end{aligned} \tag{3.6}$$

in which $\boldsymbol{\omega}_{fl} = \omega_{fl}\,\hat{\boldsymbol{a}}_l$ and $\boldsymbol{\omega}_{fr} = \omega_{fr}\,\hat{\boldsymbol{a}}_r$ are the angular velocity vectors of the two wheels, and $\omega_{fl}$ and $\omega_{fr}$ are the angular speeds provided to each flywheel by its actuation system. Above $I_{f,zz}$ represents the flywheel inertia component about its rotation axis. The latter equation shows that the difference of the two angular speeds impacts roll rotations, while their sum can be used to steer

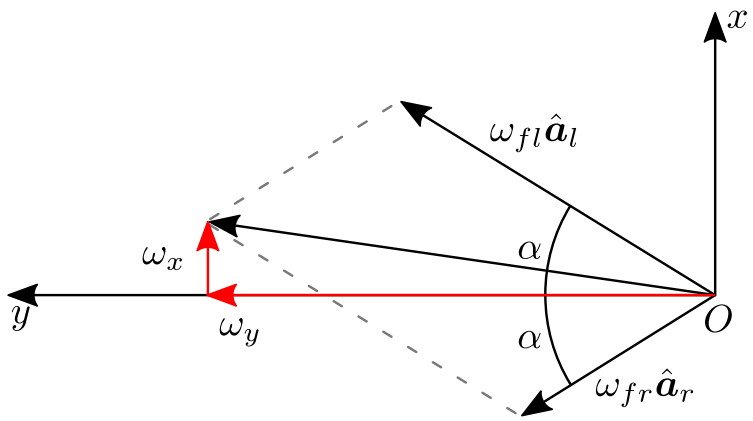



**Figure 3.6:** Influence of the right and left flywheels' speed on the robot angular velocity. Having incident rotation axes, the OCS allows control of both the robot's roll and pitch. Notice that the roll angle is influenced by the difference of the angular speeds of the flywheels, $\omega_x = (\omega_{fl} - \omega_{fr})\sin(\alpha)$. On the contrary, the pitch angle depends on the sum of the angular speeds $\omega_y = (\omega_{fl} + \omega_{fr})\cos(\alpha)$.

the pitch, see Figure 3.6. Using the definition of $\boldsymbol{C}$, (3.6) is rewritten as:

$$\begin{aligned} {}_b\boldsymbol{L}_f &= I_{f,zz} \begin{bmatrix} \sin(\alpha) & -\sin(\alpha) \\ \cos(\alpha) & \cos(\alpha) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_{fl} \\ \omega_{fr} \end{bmatrix} \\ &= I_{f,zz}\boldsymbol{C}\boldsymbol{\omega}_f. \end{aligned} \tag{3.7}$$

### 3.2.3 Flywheels' Inertia

Once the desired inertia $I_f$ is selected according to the 2D Elroy's Beanie model, it is used to realize the 3D OCS (Figure 3.7). Two identical flywheels were designed to have the shape of hollow cylinders. This shape increases the inertia by locating the mass far away from the rotation axis. The inertia tensor expressed in its principal axes, $\boldsymbol{I}_f = \text{diag}\{I_{f,xx}, I_{f,yy}, I_{f,zz}\}$, depends on the cylinders' inner and outer radii, $r$ and $R$, its height $h$ and material density $\rho$:

$$\begin{aligned} I_{f,xx} = I_{f,yy} &= \frac{1}{12}\pi\rho h\left(3\left(R^4 - r^4\right) + h^2\left(R^2 - r^2\right)\right) \\ I_{f,zz} &= \frac{1}{2}\pi\rho h\left(R^4 - r^4\right) \end{aligned} \tag{3.8}$$

Notice that in Elroy's Beanie model, a single body models the complete OCS. The inertia introduced in Elroy's Beanie model has to be split between the two

flywheels: $I_{f,zz} = I_f / (2\cos(\alpha))$. The parameters $R$ and $h$ can be set to have a compact OCS and $\rho$ depends on the chosen material, which was stainless steel in our case. The inner radius $r$ could be adjusted to obtain the desired inertia. Spokes with negligible mass connected the wheel to the motor shaft. All the parameters are reported in Table 3.1, together with the selected inertia and mass of a single flywheel.

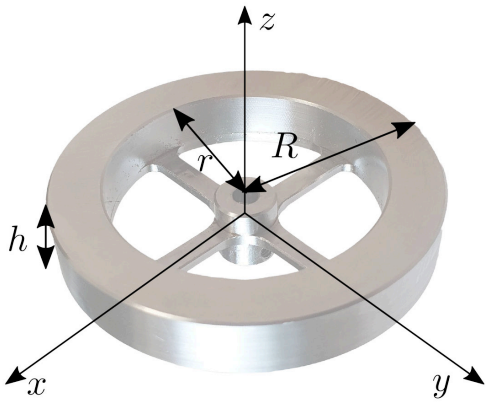



**Figure 3.7:** Picture of the final design of the flywheel, shown together with its principal axes of inertia. Here, $h$ is the wheel thickness, $r$ is the inner radius of the wheel, and $R$ is the radius of the outer one.

**Table 3.1:** Parameters of a single flywheel.

| Parameter | Value | Unit |
|---|---|---|
| $\dot{\gamma}_{MAX}$ | 5000 | RPM |
| $\Delta\dot{\theta}$ | 30 | °/s |
| $r$ | 0.0220 | m |
| $R$ | 0.0300 | m |
| $h$ | 0.0102 | m |
| $\rho$ | 7860 | kg/m$^3$ |
| $m$ | 0.102 | kg |
| $I_{xx}, I_{yy}$ | $3.64 \times 10^{-5}$ | kg · m$^2$ |
| $I_{zz}$ | $7.11 \times 10^{-5}$ | kg · m$^2$ |

### 3.2.4 Control Law

To derive a control law based on the robot's base orientation, we made use of (3.1), expressing all the contributions to the time derivative of total angular

momentum with respect to the base reference frame:

$$ {}_b\boldsymbol{I}_f \, {}_b\dot{\boldsymbol{\omega}}_f + {}_b\boldsymbol{I}_r \, {}_b\dot{\boldsymbol{\omega}}_r + {}_b\boldsymbol{\omega}_r \times ({}_b\boldsymbol{I}_r \, {}_b\boldsymbol{\omega}_r) = \mathbf{0} \tag{3.9} $$

From this expression, the moment on the base caused by the acceleration of the flywheels is defined and can be used as a feedback torque $\boldsymbol{\tau}_{fb}$:

$$ \begin{aligned} \boldsymbol{\tau}_{fb} &= {}_b\boldsymbol{I}_f \, {}_b\dot{\boldsymbol{\omega}}_f \\ &= -{}_b\boldsymbol{I}_r \dot{\boldsymbol{\omega}}_b - {}_b\boldsymbol{\omega}_r \times ({}_b\boldsymbol{I}_r \, {}_b\boldsymbol{\omega}_r) \end{aligned} \tag{3.10} $$

We use a proportional and derivative control law based on the orientation error $\boldsymbol{e} \in SO(3)$:

$$ \boldsymbol{K}_p \boldsymbol{e} + \boldsymbol{K}_d \dot{\boldsymbol{e}} \tag{3.11} $$

where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are diagonal and positive definite matrices of gains. The orientation error $\boldsymbol{e}$ requires that the algebra of the special rotational group is computed. Further details are reported in Appendix E. The derivative error can be computed using $\dot{\boldsymbol{e}} = {}_b\boldsymbol{\omega}_r{}^{des} - {}_b\boldsymbol{\omega}_r$, in which ${}_b\boldsymbol{\omega}_r{}^{des}$ and ${}_b\boldsymbol{\omega}_r$ are, respectively, the desired and actual angular velocities of the base:

$$ \boldsymbol{\tau}_{fb} = -{}_b\boldsymbol{I}_r \left( \boldsymbol{K}_p \boldsymbol{e} + \boldsymbol{K}_d \dot{\boldsymbol{e}} \right) - {}_b\boldsymbol{\omega}_r \times ({}_b\boldsymbol{I}_r \, {}_b\boldsymbol{\omega}_r) \, . \tag{3.12} $$

Projecting the moment $\boldsymbol{\tau}_{fb}$ onto the flywheel axes with $\boldsymbol{C}^T$, the control action $\boldsymbol{u}$ yields

$$ \boldsymbol{u} = \boldsymbol{C}^T \boldsymbol{\tau}_{fb}. \tag{3.13} $$

## 3.3 Results of Simulations

Three simulations on different scenarios are reported to validate our OCS. We wanted to test the capability of the proposed approach in terms of the following:

1. to reject a disturbance when the robot is in the flying phase of a jump,
2. to dampen trunk oscillations after TD,
3. and to work in parallel with the joints of the legs to achieve a highly dynamic motion.

All the simulations were performed within the robotics framework Locosim (see Appendix A), using Gazebo [Koenig and Howard, 2004] as simulation environment. References for the joints of the legs, relative to a jump motion,

were computed off-line using Crocoddyl [Mastalli et al., 2020], an optimal control library for robots based on DDP algorithms. It uses Pinocchio [Carpentier et al., 2019] for fast computation of robot dynamics and of analytical derivatives. References $\hat{\boldsymbol{q}}_{ref}$, $\dot{\hat{\boldsymbol{q}}}_{ref}$ and $\boldsymbol{\tau}_{ref}$ for joint positions, velocities, and torques were then executed on-line with a proportional-derivative joint controller:

$$\boldsymbol{\tau}_j = \boldsymbol{K}_{p,j}(\hat{\boldsymbol{q}}_{ref} - \hat{\boldsymbol{q}}) + \boldsymbol{K}_{d,j}(\dot{\hat{\boldsymbol{q}}}_{ref} - \dot{\hat{\boldsymbol{q}}}) + \boldsymbol{\tau}_{ref} \tag{3.14}$$

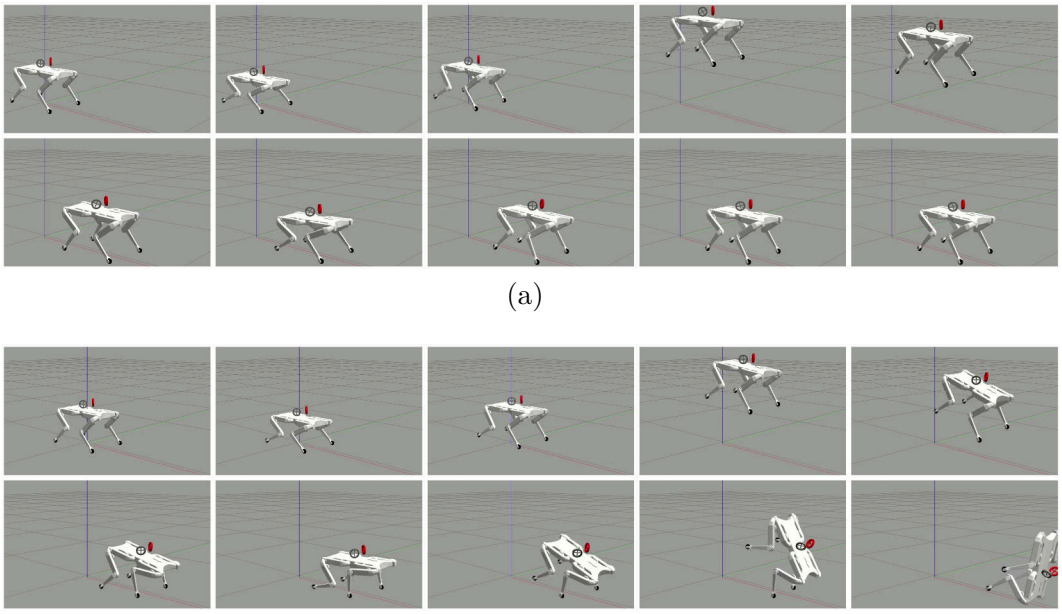
The OCS was commanded to track base orientation references using the control law introduced in Section 3.2.4 coded in Python.

### 3.3.1 Disturbance Rejection

With the first test, we want to prove the ability of disturbance rejection of the OCS. During the flying phase of a forward jump, 0.1s after the lift-off, an external disturbance moment was applied to the robot's base, deflecting the robot's orientation. This disturbance, expressed in the world reference frame, was set to

$$\boldsymbol{\tau}_{dist} = \begin{bmatrix} 0.2 \\ 0.8 \\ 0.0 \end{bmatrix} \text{N} \cdot \text{m}$$

and applied for 0.05 s. If the flywheels were not used, Solo12 fell down after TD. Instead, when using the flywheels it was possible to drive the robot to a safe configuration after landing without the need to implement a specific landing strategy. Respectively, Figure 3.8 and Figure 3.9 report snapshots and plots of the described task, with the OCS enabled or disabled.



(a)

(b)

**Figure 3.8:** Simulation: disturbance rejection. Snapshots of a forward jump with disturbance acting on the robot's trunk. Top: Using the proposed OCS, Solo12 successfully completes the aerial motion planned using DDP. Bottom: if the OCS is disabled, Solo12 crushes on the ground after the TD because of poor tracking to the planned joints reference.

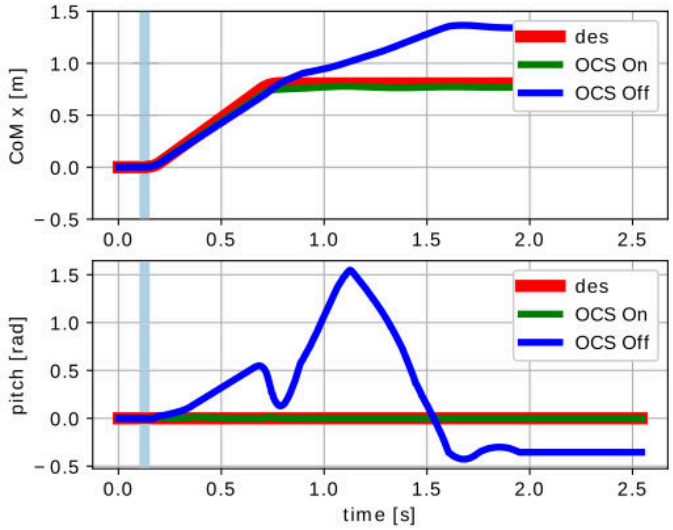



**Figure 3.9:** Simulation: disturbance rejection. Results showing the forward CoM trajectory and the pitch one vs. time plots. A disturbance moment $\boldsymbol{\tau}_{dist}$ on the trunk is compensated enabling the OCS. If it was disabled, the robot was unable to restore a safe configuration post-TD and, eventually, fell down. The light blue area represents the interval of time in which the disturbance was applied.

### 3.3.2 Trunk Realignment

The second test demonstrated the ability of the OCS to reorient the base. The robot robot was dropped from a height of 1.0 m with an initial pitch orientation of 30° and zero base angular velocity. If the flywheels were not actuated, the robot touched the ground with the same initial orientation. Nevertheless, the trunk oscillated in the forward and pitch directions during the landing phase. If the OCS was enabled, it drove the robot's base to be horizontal when it was still in the air, and the oscillations after TD were drastically reduced, both in the pitch angle and forward direction (Figure 3.10).

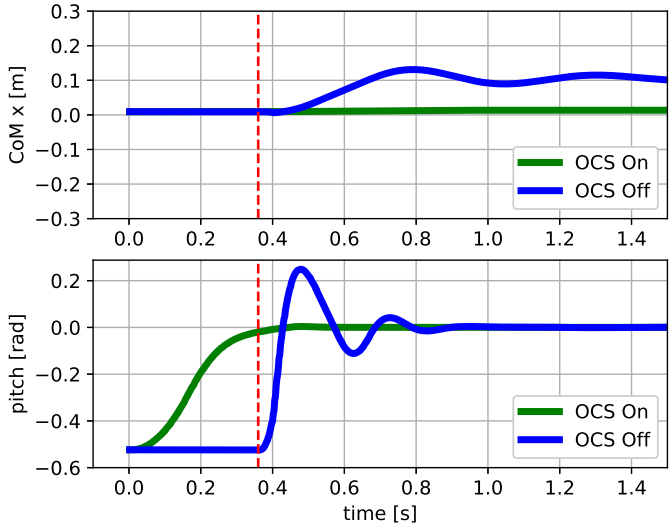



**Figure 3.10:** Simulation: trunk realignment. Results showing the $\text{CoM}_x$ and pitch trajectory vs. time plots. The OCS drove the robot's orientation during a fall so as to be horizontal. This allowed dampening of the base oscillations after TD (vertical dashed line), even without implementing a landing strategy.

### 3.3.3 Somersault

Finally, the ability of the flywheels to relieve the efforts of the leg joints to achieve a highly dynamic movement was shown with a backflip (Figure 3.11). For this, we targeted a space application, carrying out a simulation with lunar gravity acceleration (1.62 $\text{m/s}^2$). In this way, it was possible to obtain high

jumps with a long flying phase without having to select more powerful actuators. The leg joint trajectory computed off-line described a purely vertical jump of 1 m, having a flying phase that lasted 2 s. Right after the lift-off, the flywheels started the trunk reorientation task of performing a back-flip, by means of a spin of 360° on the pitch. For this maneuver, the value of the incident angle $\alpha$ was set to 0°, since no roll rotation was required[1]. This simulation demonstrated that the OCS alleviated the effort applied to the legs. Indeed, it would not have been possible to execute a backflip without the flywheels since the joint torques were planned for executing a pure vertical jump of 1 m, as can be seen in the attached video.

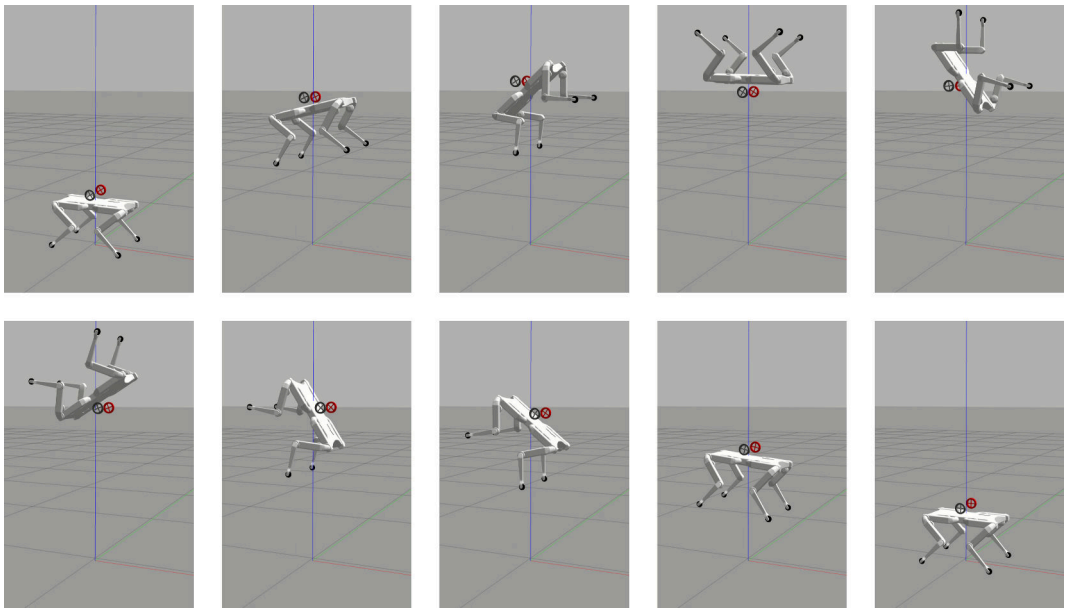


**Figure 3.11:** Simulation: somersault. The sequence of snapshots of Solo12 performing a back-flip in the Gazebo environment. The leg actuators account for the vertical motion, while the flywheels' task is to execute a complete pitch rotation. The red, green, and blue lines represent the axes of the inertial (world) reference frame.

## 3.4 Summary

This chapter presented the design of a novel OCS, composed of two flywheels, that enable the control of the trunk of a legged robot, increasing the accuracy

[1]Our hardware design allowed manual change of this value before performing the task.

of aerial maneuvers when the system is subject to the nonholonomic constraint of conservation of the angular momentum, as well as enhancing stability when in contact after TD by damping the oscillations. The novelty of the design, which involves flywheels attached with incident rotational axes on the trunk, is that it allows control of the orientations in both the roll and pitch directions while keeping the device compact. The effect is optimized considering the inertial property of the mechanical structure. Simulations performed with the quadruped Solo12 validated the effectiveness of the proposed approach in the following: to reject disturbances during the flying phase, to stabilize the platform after TD (even in the absence of a specific landing strategy), and to achieve a fast reorientation maneuver in a reduced gravity environment. At the current stage, the OCS can control roll and pitch angles, leaving the yaw and its rate in an uncontrolled state. We suppose that strategies based on Brockets theorem could exploit the nonholonomic constraint imposed by the angular momentum conservation and impose a control action also on the yaw.

# 4

# Reactive Landing Controller for Quadruped Robots

*Safely recovering from unexpected falls or planned drops is still an open problem. It is further made more difficult when high horizontal velocities are involved. In this chapter, we discuss about an optimization-based reactive Landing Controller (LC) that uses only proprioceptive measures for quadruped robots that free-fall on a flat horizontal ground, knowing neither the distance to the landing surface nor the flight time. Utilizing an estimate of the horizontal velocity of the CoM, this method employs the Variable Height Springy Inverted Pendulum (VHSIP) model to continuously adjust the positions of the feet as the robot is in the flying phase. To prevent damaging impacts, these adjustments are chosen to ensure that the GRFs following the Touch Down (TD) do not induce a net angular moment and to eliminate the possibility of multiple bounces. In this way, the quadruped is ready to attain a successful landing in all directions, even in the presence of significant horizontal velocities. A number of tests are reported in the attached video, available at* https://youtu.be/wiuedeHfSEY.

Published. Journal paper: Francesco Roscia, Michele Focchi, Andrea Del Prete, Darwin G. Caldwell, and Claudio Semini, "Reactive Landing Controller for Quadruped Robots," in *IEEE Robotics and Automation Letters*, vol. 8, issue 11, 2023, https://doi.org/10.1109/LRA.2023.3313919.

High horizontal velocity makes the landing problem for a quadruped robot challenging. There are many situations where this aspect is not negligible, e.g., when a quadruped trots at a high speed and must do a leap without first stopping the motion. So, for a robot to be able to attain a stable standing posture after a fall, the control framework should estimate and deal with both the vertical and horizontal velocity components. This chapter addresses the problem of landing on a flat horizontal surface for a quadruped robot. Specifically, the objective is to define a control strategy that satisfies the following requirements once the robot touches the ground:

**(R1)** no bounces after landing;

**(R2)** the trunk must not hit the ground;

**(R3)** the robot must reach a stable standing state;

**(R4)** once landed, the feet must not slip.

To meet these requirements, we model the dynamics of the quadruped after the TD as a Variable Height Springy Inverted Pendulum (VHSIP). The employment of models similar to the SLIP is not a new concept in the literature (Section 2.1), and they have also been utilized in the context of aerial motion


**Figure 4.1:** Based only on proprioceptive measures, during a fall, the proposed Landing Controller (LC) adjusts Go1's limbs posture in order to drive the robot to a stable standing configuration after Touch Down (TD), avoiding bounces, feet slippage, and undesirable contacts with the ground.

[Blickhan, 1989; Raibert, 1986; Xiong and Ames, 2018]. Despite its low complexity, the template model captures the linear dynamics of the robot with sufficient accuracy and is inherently suitable for damping impacts with the terrain. It allows us to design a control architecture with a short computation time, keeping it adequately reactive to change the control strategy at TD promptly. Our approach relies only on proprioceptive measurements, so a motion capture or visual perception system is not required. The contributions of this chapter are listed in the following.

- We introduce a real-time *omni-directional* LC framework (Figure 4.2), which can handle horizontal velocity, is tolerant to TD timing uncertainties, and does not need an estimate of the robot absolute position.
- We demonstrate successful landings in both a simulation environment and with real hardware from various heights and significant horizontal velocity, up to 3 m/s dropping the robot from a height of 1 m. To our best knowledge, this is the first time a quadruped robot can recover from a fall with such horizontal velocity relying only on proprioceptive measures.
- We present a detailed analysis showing the advantages and limitations

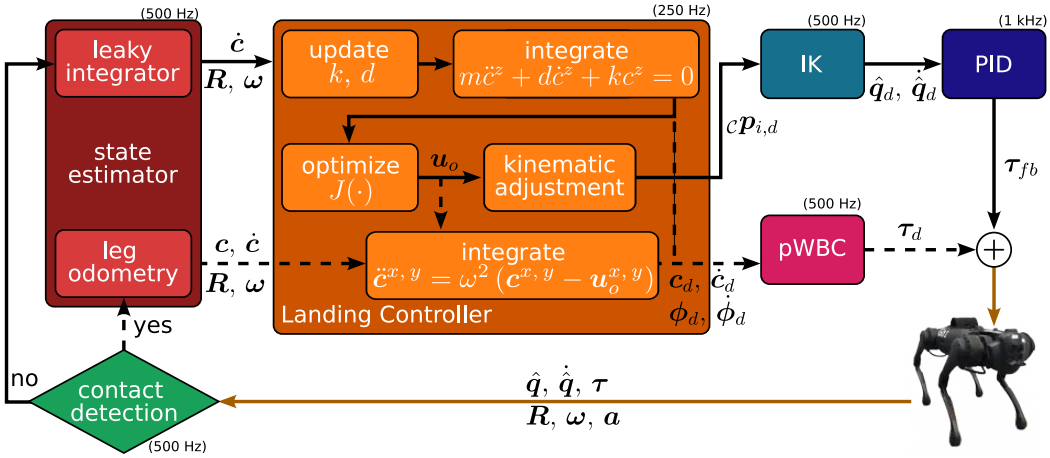



**Figure 4.2:** Overview of the LC framework. When the robot is in the flying phase (solid black branches), the virtual foot location is continuously re-optimized using the template model Variable Height Springy Inverted Pendulum (VHSIP), and the feet are shifted accordingly. Once landing is detected, the last computed trajectory for the CoM and trunk orientation is tracked. The dashed branches are active only after TD is detected. Sensing of proprioceptive measurements and actuation of control efforts are always active (brown branches).

of our LC. We verify that it outperforms a naive approach that does not move the feet when the robot is falling. Although the template model neglects angular dynamics, our approach can tolerate drops starting with non-negligible orientations, e.g., with roll between $-40°$ and $30°$, or pitch rate from $-440°/s$ to $210°/s$ .

Even though the approach was developed and tested for four-legged robots, it is generic enough to be easily extended to robots with any number of legs.

The remaining content of this chapter is organized as follows. In Section 4.1, we discuss the VHSIP template model and formally state the landing problem. The structure of the landing framework is detailed among Section 4.2, Section 4.3, and Section 4.4, devoted to motion generation, TD detection and state estimation, and motion control, respectively. Finally, we present implementation details, simulation and experimental results in Section 4.5.

## 4.1 Modeling and Problem Formulation

The fall of a legged robot can be decomposed in two phases: a flying phase, in which the robot is subjected only to gravity, and a landing phase, which begins when limbs (typically the feet) come into contact with the environment and generate GRFs. The switching instant between the two phases is named Touch Down (TD). In this section, we derive the VHSIP model that we use as a template to formalize and address the landing problem.

### 4.1.1 Derivation of the Variable Height Springy Inverted Pendulum Model

Let us introduce an inertial coordinate frame $\mathcal{W}$: the $Z$–axis is orthogonal to a flat ground with the $XY-$plane lying on it. In this frame, the robot can be seen as a single rigid body of mass $m \in \mathbb{R}^+$, lumped at its CoM $\boldsymbol{c} \in \mathbb{R}^3$, having inertia ${}_{\mathcal{C}}\boldsymbol{I}_{\mathcal{C}} \in \mathbb{R}^{3\times3}$. When the robot is falling, the balance of moments constrains the linear dynamics to follow the ballistic trajectory while conserving the angular momentum $\boldsymbol{L} \in \mathbb{R}^3$. Whenever the robot enters in contact with the ground, with $n_c$ contact points, the balance of linear and angular moments can be written as in (2.2)–(2.3) Under the assumptions of horizontal ground ($p_i^z = 0,\ \forall i = 1, \ldots, n_c$) and negligible variation of the angular momentum

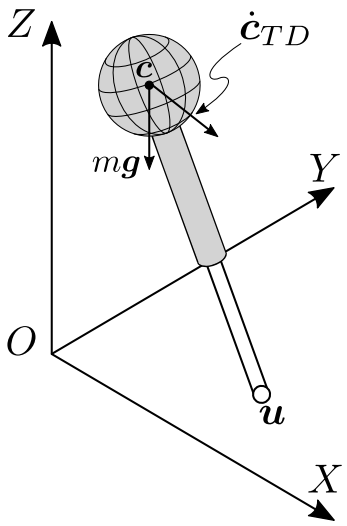



**Figure 4.3:** The 3D VHSIP model used to describe the robot dynamics after TD. This figure is inspired by figure 3 of [Koolen et al., 2012].

($\dot{\boldsymbol{L}} \approx 0$), we obtain the relationship

$$\ddot{\boldsymbol{c}}^{x,y} = \omega^2(t)\left(\boldsymbol{c}^{x,y} - \boldsymbol{u}\right), \tag{4.1}$$

where $\boldsymbol{u} \in \mathbb{R}^3$ is the CoP, lying on the ground, defined as (2.10). The dynamics in the horizontal directions of the CoM are decoupled, but they both depend on the vertical motion through

$$\omega^2 \triangleq \frac{(g^z + \ddot{c}^z)}{c^z}, \tag{4.2}$$

which is not constant over time, unlike the case of the LIP model. The time-dependent parameter $\omega\,(t)$ allows us to describe the robot motion as a variable height linear inverted pendulum. Let us assume the CoP to be still on the ground, as a *virtual foot*. The vector $\overrightarrow{\boldsymbol{cu}}$ can change its length and rotate about $\boldsymbol{u}$ due only to the gravity force $m\boldsymbol{g}$ and the initial CoM velocity $\dot{\boldsymbol{c}}_0^{x,y}$ (for our case it equals the TD velocity $\dot{\boldsymbol{c}}_{TD}^{x,y}$, see Figure 4.3). If the CoM height is constant, (4.1) simplifies to the LIP model [Kajita and Tani, 1991]. Nevertheless, such an assumption is too restrictive. Indeed, to achieve a smooth landing and dissipate the impact energy effectively, it is convenient to enforce the vertical CoM dynamics after TD to behave as a Mass-Spring-Damper (MSD) system:

$$m\ddot{c}^z + d\dot{c}^z + k(c^z - l_0) = F^z, \tag{4.3}$$

in which $l_0$ is the spring rest position, and $k$, $d \in \mathbb{R}^+$ are the *virtual stiffness* and *virtual damping* coefficients, respectively. If the joint controller already implements a gravity compensation strategy, one may assume $F^z = 0$. In this way, the vertical CoM dynamics is an autonomous system, depending only on the state at TD: $c^z_{TD}$ and $\dot{c}^z_{TD}$. We will refer to (4.1)–(4.3) as the VHSIP model. Let us denote with $\boldsymbol{\xi}$ the system CP.

Since the CoM dynamics of the VHSIP in the two horizontal directions are equivalent and decoupled, we analyze only the motion along the $X-$ axis, keeping in mind that the arguments are valid also along the $Y-$ axis. After TD, the VHSIP is associated with a conserved quantity, the so-called Orbital Energy $E(c^x - \xi^x, \dot{c}^x)$ [Pratt and Drakunov, 2007]:

$$E = \frac{1}{2}\dot{r}^{x^2} h^2(r^x) + g^z r^{x^2} f(r^x) - 3g^z \int_0^{r^x} f(\zeta)\zeta d\zeta$$

with $r^x = c^x - \xi^x$, $f$ is a twice-differentiable function[1] for which it holds $c^z = f(r^x)$ and $f'$ is its derivative. Additionally, we define $h(r^x) = f(r^x) - f'(r^x)r^x$. If the CoM is moving toward the virtual foot with $E > 0$, then orbital energy is sufficient to let the CoM pass over $\boldsymbol{\xi}$ and continue on its way. If $E < 0$, then the CoM will stop and reverse the direction of motion before getting over the virtual foot. If $E = 0$, then the CoM converges to rest above $\boldsymbol{\xi}$, [Pratt et al., 2006]. Therefore, robot configurations in equilibrium can be reached by selecting $\boldsymbol{\xi}$ so that at TD the following condition holds

$$E(c^x_{TD} - \xi^x, \dot{c}^x_{TD}) = 0. \tag{4.4}$$

From Section 2.2.2, we know that if $\boldsymbol{\xi}$ is confined in the feet polygon, then it is possible to sex $\boldsymbol{u} = \boldsymbol{\xi}$ with the CoM dynamics converging above them. Whereas for the LIP it is possible to explicitly compute the CP, for our VHSIP model, this is not the case. In the following, we show how we overcome this complexity.

[1] The scalar function $f$ maps horizontal displacements to heights. It exists as long as $c^x(t)$ is a bijective function of time. As a matter of fact, in this case, $c^x(t)$ admits an inverse that would allow us to write $c^z(t)$ as a function of $c^x(t)$. The trajectory $c^x(t)$ is bijective if the virtual foot $\xi^x$ is constant and $\omega^2(t) > 0$. The former clause is already taken as an assumption. The latter occurs whenever the CoM does not penetrate the ground ($c^z > 0$) and there is no pulling force from the ground ($\ddot{c}^z > -g^z$): this clause is always verified for the problem we are considering.

### 4.1.2 Landing Problem

Now, we can state the problem more formally.

**Problem Statement.** *Consider a robot that is free falling onto a flat horizontal ground with negligible angular velocity. Without knowing the distance between the robot and the ground, find the parameters for the template VHSIP model (i.e., $k$, $d$ and $\boldsymbol{u}$) that fulfill the requirements **(R1)**, **(R2)**, **(R3)**, and **(R4)**. Then, compute the joint torques that realize the CoM motion obtained with the selected template model.*

## 4.2 Motion Reference Generation

This section introduces and discusses the structure of our LC. At any sampling instant, we suppose that the TD is about to occur and use the template VHSIP model by continuously updating the CoM trajectory. Knowing the position of the CoM relative to the robot's feet, we can avoid estimating the robot's absolute position when no contact is active. During the flying phase, the system has to prepare to dissipate the kinetic energy throughout the landing phase. The robot should adjust its limbs such that, after the TD, the virtual foot, i.e., the CoP, will be on the CP. Notice that (4.1)–(4.3) are defined in an inertial frame, but our LC is designed to be independent of absolute position estimates. We circumvent this limitation by introducing the terrain frame $\mathcal{T}$. First, we denote the robot CoM frame with $\mathcal{C}$ having the origin on the CoM, the $X_{\mathcal{C}}$–axis along the forward direction of the main body of the robot and the $Y_{\mathcal{C}}$–axis orthogonal to it, pointing towards the left side (see Figure 4.4). The terrain frame is a horizontal frame [Barasuol et al., 2013], hence the $X_{\mathcal{T}}Y_{\mathcal{T}}$-plane is orthogonal to gravity and $X_{\mathcal{T}}$ and $X_{\mathcal{C}}$ lie on the same plane. We will refer to the distance between $O_{\mathcal{T}}$ and $O_{\mathcal{C}}$ during the flying phase as $l_0$, which is a user-modifiable parameter. For a given control interval, the frame $\mathcal{T}$ is fixed and can be employed to compute the new CoM reference trajectory. At the subsequent control interval, one of the following two alternative conditions will arise:

- TD is detected. Thus, the CoM reference will be tracked, stabilizing the system.
- TD is not detected. A *kinematic adjustment* (detailed in Section 4.2.4) is performed in order to keep the feet aligned with the ground despite

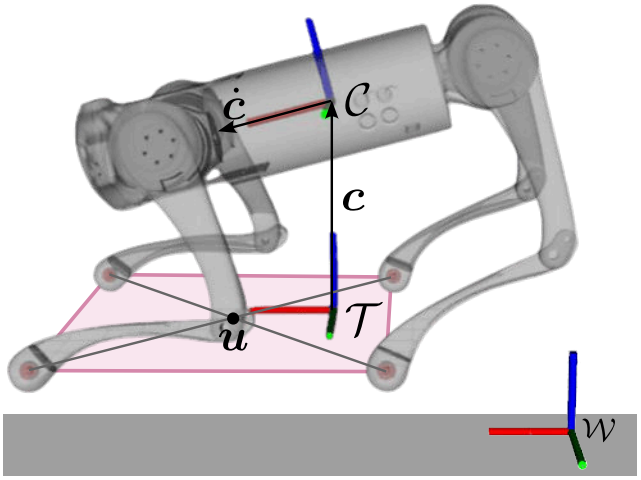



**Figure 4.4:** The CoM frame $\mathcal{C}$ has the origin at the robot CoM and is aligned with the trunk. The terrain frame $\mathcal{T}$ is the horizontal reference frame for which the CoM has coordinates $\boldsymbol{c} = \begin{bmatrix} 0 & 0 & l_0 \end{bmatrix}^T$ when the robot is in air. After TD, $\mathcal{T}$ is kept fix. Respectively, red, green, and blue segments denote the $X$, $Y$, and $Z$ axes. The robot is performing the kinematic adjustment since its orientation is not horizontal and $\dot{\boldsymbol{c}}$ has a non-null horizontal component; hence the legs are moved forward.

the base orientation, i.e., on the $X_{\mathcal{T}}Y_{\mathcal{T}}$–plane. Then, a new terrain frame is set, and the process repeats.

The benefits of continuous re-computation are two-fold: to make the robot reactive and to avoid the need for state estimation[2]. Since all the quantities in the remainder of this section are expressed in the terrain frame, henceforth, we will omit the frame for the sake of readability.

### 4.2.1 Vertical Motion Reference

In the context of the VHSIP model, the requirement **(R1)** is equivalent to having a critically-damped oscillator in (4.3). This can be achieved by setting the damping equal to $d = 2\sqrt{km}$. Thus, the two system poles are real and coincident in $\lambda = -\sqrt{k/m}$. The kinematic adjustment module guarantees that when the TD occurs, the CoM is always at a distance $l_0$ from the floor. Then, the reference trajectory for the CoM in the vertical direction depends only on the TD velocity $\dot{c}^z_{TD} < 0$. The evolution of a critically-damped system can be

[2]We still have to estimate the linear velocity, but the implementation described in Section 4.3.2 mitigates the influence of accelerometer biases.

computed in closed form:

$$\dot{c}^z(t) = e^{\lambda(t-t_{TD})} \ (\lambda\,(t - t_{TD}) + 1) \ \dot{c}^z_{TD} \tag{4.5a}$$

$$c^z(t) = e^{\lambda(t-t_{TD})} \ (t - t_{TD}) \ \dot{c}^z_{TD} + l_0 \tag{4.5b}$$

The stiffness $k$ must be selected to ensure the minimum value for $c^z(t)$ is above the ground, preventing undesired collisions between trunk and ground (requirement **(R2)**). A critically-damped system makes this minimum unique for $t > t_{TD}$. Therefore, denoting with $\Delta^z$ the minimum terrain clearance, the requirement $c^z(t) \geq \Delta^z$ for $t \geq t_{TD}$ can be translated in a lower-bound for the stiffness coefficient:

$$k \geq k_{lb} = \frac{m}{\Big(e\,(\Delta^z - l_0)\Big)^2}\,(\dot{c}^z_{TD})^2\,.$$

We set a maximum for the convergence time to prevent long dynamic evolution when TD velocity is low. Being (4.3) a stable second-order system with coincident eigenvalues, it is at steady-state for $t \geq -7/\lambda$. Choosing the maximum time of convergence $t_c$, another limitation for the stiffness comes out:

$$k \leq k_{ub} = m\left(\frac{7}{t_c}\right)^2.$$

At every control instant, we fix the virtual stiffness to $k = \max\{k_{lb},\, k_{ub}\}$ and update the virtual damping accordingly. Then, a new CoM reference trajectory is computed.

### 4.2.2 Horizontal Motion Reference

To prevent the robot from tipping over during landing and to stabilize it into a standing configuration **(R3)**, we seek for the (constant) virtual foot location $\boldsymbol{u} = \begin{bmatrix} u^x & u^y \end{bmatrix}^T$ that drives the CoM above it with zero linear velocity when the system reaches steady-state, i.e., that attains zero orbital energy (4.4). Equation (4.1) can be rewritten in state-space form:

$$\underbrace{\begin{bmatrix} \dot{c}^x(t) \\ \ddot{c}^x(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} 0 & 1 \\ \omega^2(t) & 0 \end{bmatrix}}_{\boldsymbol{A}(t)} \underbrace{\begin{bmatrix} c^x(t) \\ \dot{c}^x(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} 0 \\ -\omega^2(t) \end{bmatrix}}_{\boldsymbol{B}(t)} u^x \tag{4.6}$$

showing its nature of Linear Time Varying (LTV) system, with $\omega(t)$ evolving in time. Even if the dynamics are linear, its integration does not admit a closed-form solution because the matrix $\boldsymbol{A}(t)$ does not commute with its integral $\int_{t_{TD}}^{t} \boldsymbol{A}(\sigma)d\sigma$, [Chen, 1999]. Then, to get a numerical solution, we set up an Optimal Control Problem (OCP) to find the CP position $u_o^x$ that makes the CoM converge above it with null velocity over a finite horizon. Hereafter, when a time-varying variable appears with a subscript, e.g., $k$, it must be interpreted as evaluated at time $kT_s$, for instance, $x_k = x(kT_s)$, being $T_s$ the sampling time. Our OCP relies on forward Euler integration, and it is formulated as

$$\begin{aligned} \min_{\mathbf{x}_k,\, u^x} \;& w_p|\boldsymbol{C}_p\mathbf{x}_N - u^x|^2 + \; w_v|\boldsymbol{C}_v\mathbf{x}_N|^2 + \; w_u|u^x|^2 \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \bar{\boldsymbol{A}}_k\mathbf{x}_k + \bar{\boldsymbol{B}}_k u^x \qquad k = 0,\, 1,\, \ldots,\, N-1 \end{aligned} \tag{4.7}$$

having set the matrix obtained from the discretization of (4.6) to $\bar{\boldsymbol{A}}_k = \boldsymbol{I}_{2\times 2} + T_s\boldsymbol{A}_k$ and $\bar{\boldsymbol{B}}_k = T_s\boldsymbol{B}_k$. $\boldsymbol{I}_{2\times 2}$ is the $2\times 2$ identity matrix, and $\boldsymbol{C}_p = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and $\boldsymbol{C}_v = \begin{bmatrix} 0 & 1 \end{bmatrix}$ are selection matrices that pick out position and velocity from the state $\mathbf{x}$, respectively. Moreover, $w_p,\, w_v,\, w_u \in \mathbb{R}^+$ are weights that penalize the final CoM deviation from the virtual foot, the final CoM velocity, and the virtual foot distance from $O_{\mathcal{T}}$. The horizon $N$ should be large enough to guarantee that the system is at steady-state at time $NT_s$. We solve (4.7) with a single shooting approach. For doing so, the knowledge of $\omega^2(t)$, $0 \leq t \leq NT_s$, is needed. Thus, (4.3) must be forward integrated up to $NT_s$. Then, by recursively applying (4.1), we can express the state evolution as a function of the initial condition $\mathbf{x}_0$ and of the virtual foot location $u^x$ as

$$\mathbf{x}_k = \boldsymbol{P}_k^x\mathbf{x}_0 + \boldsymbol{P}_k^{xu}u^x. \tag{4.8}$$

The matrices $\boldsymbol{P}_k^x$ and $\boldsymbol{P}_k^{xu}$ are defined by iteration:

$$\begin{cases} \boldsymbol{P}_k^x &= \bar{\boldsymbol{A}}_{k-1}^x\,\boldsymbol{P}_{k-1}^x \\ \boldsymbol{P}_0^x &= \boldsymbol{I}_{2\times 2} \end{cases} \qquad \begin{cases} \boldsymbol{P}_k^{xu} &= \bar{\boldsymbol{A}}_{k-1}^x\,\boldsymbol{P}_{k-1}^{xu} + \bar{\boldsymbol{B}}_k^x \\ \boldsymbol{P}_0^{xu} &= \boldsymbol{0}_{2\times 1} \end{cases}$$

for $k = 1, \ldots, N$. With this notation, (4.7) rewrites as an unconstrained Quadratic Program (QP) in the scalar optimization variable $u^x$:

$$\min_{u^x} \; J(\mathbf{x}_0, u^x) \triangleq \mathbf{x}_0^T\boldsymbol{Q}^x\mathbf{x}_0 + u^{xT}\boldsymbol{Q}^u u^x + 2\mathbf{x}_0^T\boldsymbol{Q}^{xu}u^x, \tag{4.9}$$

where $\boldsymbol{Q}^x$, $\boldsymbol{Q}^u$ and $\boldsymbol{Q}^{xu}$ are real matrices of dimensions $2 \times 2$, $1 \times 1$ and $2 \times 1$, respectively. They are obtained with the following computations:

$$\begin{aligned}
\boldsymbol{Q}^x &= w_p \left(\boldsymbol{C}_p \boldsymbol{P}_N^x\right)^T \left(\boldsymbol{C}_p \boldsymbol{P}_N^x\right) + \; w_v \left(\boldsymbol{C}_v \boldsymbol{P}_N^x\right)^T \left(\boldsymbol{C}_v \boldsymbol{P}_N^x\right), \\
\boldsymbol{Q}^u &= w_p \left(\boldsymbol{C}_p \boldsymbol{P}_N^{xu} - 1\right)^T \left(\boldsymbol{C}_p \boldsymbol{P}_N^{xu} - 1\right) + \; w_v \left(\boldsymbol{C}_v \boldsymbol{P}_N^{xu}\right)^T \left(\boldsymbol{C}_v \boldsymbol{P}_N^{xu}\right) + \; w_u, \\
\boldsymbol{Q}^{xu} &= w_p \left(\boldsymbol{C}_p \boldsymbol{P}_N^x\right)^T \left(\boldsymbol{C}_p \boldsymbol{P}_N^{xu} - 1\right) + \; w_v \left(\boldsymbol{C}_v \boldsymbol{P}_N^x\right)^T \left(\boldsymbol{C}_v \boldsymbol{P}_N^{xu}\right).
\end{aligned}$$

Zeroing the partial derivative of the cost $J(\cdot)$, the optimal virtual foot location is found:

$$u_o^x = -\frac{\boldsymbol{Q}^{xu^T}}{\boldsymbol{Q}^u} \mathbf{x}_0.$$

Notice that the minimization of $J(\cdot)$ corresponds to minimizing the Orbital Energy. Repeating the same argument for the lateral component of the CoM, the coordinates of the virtual foot in the terrain frame are obtained. If a TD occurs, the robot must track the CoM reference trajectory. This is obtained by plugging $\boldsymbol{u}_o$ into (4.1) and forward integrating along the horizon to get the horizontal components while keeping the vertical component previously computed as described in (4.3). Conversely, if a TD does not occur, the kinematic adjustment described in Section 4.2.4 is performed, shifting the feet according to the previously computed virtual foot location. Additionally, the terrain frame is updated, and a new reference is calculated.

### 4.2.3 Angular Motion Reference

Even though the VHSIP does not capture the angular dynamics of the robot, it is still valid within a range of orientations and angular rates. Suppose the robot lands with a non-horizontal trunk orientation. In that case, we plan the Euler angles $\boldsymbol{\phi}_d \in \mathbb{R}^3$ to reach a horizontal configuration while the second-order system in (4.3) attains the steady state:

$$\begin{bmatrix} \dot{\boldsymbol{\phi}}_d(t) \\ \ddot{\boldsymbol{\phi}}_d(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_d(t) \\ \dot{\boldsymbol{\phi}}_d(t) \end{bmatrix}, \qquad \begin{aligned} \boldsymbol{\phi}_d(0) &= \boldsymbol{\phi}_{TD}, \\ \dot{\boldsymbol{\phi}}_d(0) &= \dot{\boldsymbol{\phi}}. \end{aligned} \tag{4.10}$$

Euler rates can be used since the robot operates far enough from singular configurations.

### 4.2.4 Kinematic Adjustment

During the flying phase, the robot limbs must prepare to achieve the CoP at TD. We call such motion *kinematic adjustment* [Barasuol et al., 2013]. Figure 4.4 shows the robot while performing the kinematic adjustment. It consists of placing the robot feet $\boldsymbol{p}_i$ on the vertices of a rectangle parallel to the landing surface, even if the robot trunk is not horizontal. The rectangle lies on the $X_{\mathcal{T}}Y_{\mathcal{T}}-$plane, and it is shifted such that its centroid is placed onto the CP $\boldsymbol{u}_o$:

$$\boldsymbol{p}_{i,\,d}(t) = \boldsymbol{p}_{i,\,0} + \alpha(t)\,\boldsymbol{u}_o, \tag{4.11}$$

where $\boldsymbol{p}_{i,\,0}$ is the position of the $i$-th foot when the robot is in the homing configuration. To avoid abrupt changes in the reference that would affect the orientation of the base, we perform a linear interpolation with $\alpha : t \to [0,\,1]$. Denoting with ${}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}} \in SO(3)$ the rotation matrix describing the orientation of the terrain frame with respect to the trunk, we can express feet positions in the $\mathcal{C}-$frame

$${}_{\mathcal{C}}\boldsymbol{p}_{i,\,d} = {}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}}\Big(\boldsymbol{p}_{i,\,d} - \begin{bmatrix}0 & 0 & l_0\end{bmatrix}^T\Big). \tag{4.12}$$

Feet motions can be realized with joint-space control. Desired joint configurations can be computed by solving an Inverse Kinematic (IK) problem for each foot assuming mass-less legs:

$$\hat{\boldsymbol{q}}_{i,\,d} = \mathrm{IK}({}_{\mathcal{C}}\boldsymbol{p}_{i,\,d}) \tag{4.13}$$

where $\hat{\boldsymbol{q}}_{i,\,d}$ contains the desired angles of the joints on the $i-$th leg. We can employ closed form IK at the position level since the mechanical structure of the robot in use has no internal loops, and there is no redundancy. Desired joint velocities are computed considering the Cartesian-space error ${}_{\mathcal{C}}\boldsymbol{e}_i = {}_{\mathcal{C}}\boldsymbol{p}_{i,\,d} - {}_{\mathcal{C}}\boldsymbol{p}_i$:

$$\dot{\hat{\boldsymbol{q}}}_{i,\,d} = \Big(\boldsymbol{J}_{Ti}\left(\hat{\boldsymbol{q}}\right) + \varepsilon \boldsymbol{I}_{3\times 3}\Big)^{-1} k_v\, {}_{\mathcal{C}}\boldsymbol{e}_i$$

being ${}_{\mathcal{B}}\boldsymbol{J}_{pi}$ the translational Jacobian matrix from the base to the associated to the $i$-th foot, $\varepsilon \in \mathbb{R}^+$ a regularization parameter to deal with singularities, and $k_v \in \mathbb{R}^+$ a scaling factor. The joint trajectory is tracked with a joint-space Proportional Derivative (PD) controller:

$$\boldsymbol{\tau}_{fb} = \boldsymbol{K}_P^j\left(\hat{\boldsymbol{q}}_d - \hat{\boldsymbol{q}}\right) + \boldsymbol{K}_D^j\left(\dot{\hat{\boldsymbol{q}}}_d - \dot{\hat{\boldsymbol{q}}}\right) + \boldsymbol{\tau}_g\left(\hat{\boldsymbol{q}}\right) \tag{4.14}$$

with the gravity compensation $\boldsymbol{\tau}_g(\hat{\boldsymbol{q}})$ as feedforward term. We tuned off-line the diagonal matrices of positive gains $\boldsymbol{K}_P^j$ and $\boldsymbol{K}_D^j$, executing tasks with the feet not touching the ground.

## 4.3 Touch Down Detection and State Estimation

In this section, we detail the procedure adopted for determining the instant of time in which a flying phase terminates and the subsequent landing phase starts. Additionally, we present how LC framework computes the linear and angular position of the robot's base expressed with respect to the terrain frame.

### 4.3.1 Touch Down Detection

We define the transition between the flying and the landing phases, the Touch Down (TD), as the first time instant when all feet are in contact with the ground. Mechanical switches or contact sensors could be employed, but these devices are typically expensive and suffer from impacts. Since torque estimates based on the motor current are available on our robot, we identify the contact status of the $i$-th foot from an estimate of the GRFs $\hat{\boldsymbol{f}}_i$ exerted on it:

$$\hat{\boldsymbol{f}}_i = \boldsymbol{J}_{pi}^{-T}(\hat{\boldsymbol{q}})\boldsymbol{S}_i\Big(\boldsymbol{C}(\hat{\boldsymbol{q}})\,\dot{\hat{\boldsymbol{q}}} + \boldsymbol{g}(\hat{\boldsymbol{q}}) - \boldsymbol{\tau}\Big) \tag{4.15}$$

where $\boldsymbol{S}_i$ is the matrix picking out torques associated to the $i$-th leg. Denoting with $\boldsymbol{n} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ the unit vector normal to the ground, a contact is detected if the normal component of the GRF $\hat{f}_i^n \geq f_{th}^n$, with $f_{th}^n$ a (small) robot-dependent threshold.

### 4.3.2 State Estimation

Because of the terrain frame definition and of the kinematic adjustment performed in the flying phase, we do not need to estimate the position of the CoM at TD: we can assume it to be $\boldsymbol{c}_{TD} = \begin{bmatrix} 0 & 0 & l_0 \end{bmatrix}^T$. Our robot is equipped with an Inertial Measurement Unit (IMU) that outputs rotation matrix ${}_{\mathcal{W}}\boldsymbol{R}_{\mathcal{C}}$, angular velocity ${}_{\mathcal{C}}\boldsymbol{\omega}$ and linear acceleration ${}_{\mathcal{C}}\boldsymbol{a}$ (referred to the robot CoM frame). The linear acceleration in the inertial frame is retrieved after estimating the

accelerometer bias ${}_{\mathcal{C}}\boldsymbol{b} \in \mathbb{R}^3$ as

$$\boldsymbol{a} = {}_{\mathcal{W}}\boldsymbol{R}_{\mathcal{C}} \left( {}_{\mathcal{C}}\boldsymbol{a} - {}_{\mathcal{C}}\, \boldsymbol{b} \right) - \boldsymbol{g}$$

Misestimation of the bias could lead to drift. Therefore, we found it beneficial to reconstruct the base linear velocity $\boldsymbol{v} \in \mathbb{R}^3$ making use of a leaky integrator, equally to [Herzog et al., 2014]:

$$\hat{\boldsymbol{v}}_{k+1} = \left( \boldsymbol{I}_{3\times 3} - \boldsymbol{W} T_s \right) \hat{\boldsymbol{v}}_k + \boldsymbol{I}_{3\times 3}\, T_s \boldsymbol{a}_k \tag{4.16}$$

where $\boldsymbol{W} = \text{diag}\left(w^x,\, w^y,\, w^z\right)$ is a diagonal matrix of positive discount factors in the three directions. Since the flying phase has a short time duration, the leaky integrator achieves satisfactory performances in mitigating the drift due to inaccuracies of bias estimation. The estimate $\hat{\boldsymbol{v}}_k$ is plugged into the robot full model for computing the CoM velocity during the flying phase. Conversely, after TD, we rely on leg odometry to estimate the CoM position and velocity, as in [Focchi et al., 2017].

Figure 4.5 compares the performance of leg odometry and Inertial Measurement Unit (IMU) integration in estimating base frame velocity. I request the Go1 robot to execute a composite motion (up and down, forward and backward) to swing its base repetitively while keeping its feet steady by tracking the following velocity reference:

$$\begin{aligned} \boldsymbol{p}_b\left(t\right) &= \boldsymbol{p}_{b,\,0} + \begin{bmatrix} 0.03\sin\left(\pi t\right) \\ 0 \\ 0.04\cos\left(\pi t\right) \end{bmatrix} \\ \boldsymbol{\phi}_b\left(t\right) &= \boldsymbol{\phi}_{b,\,0} \end{aligned} \tag{4.17}$$

being $\boldsymbol{p}_{b,0}$ and $\boldsymbol{\phi}_{b,0}$ the base position and orientation in homing configuration. We employ the projection-based Whole Body Control (pWBC) described below in Section 4.4 to track the base reference. One immediately notices that the result of IMU integration is less affected by noises.

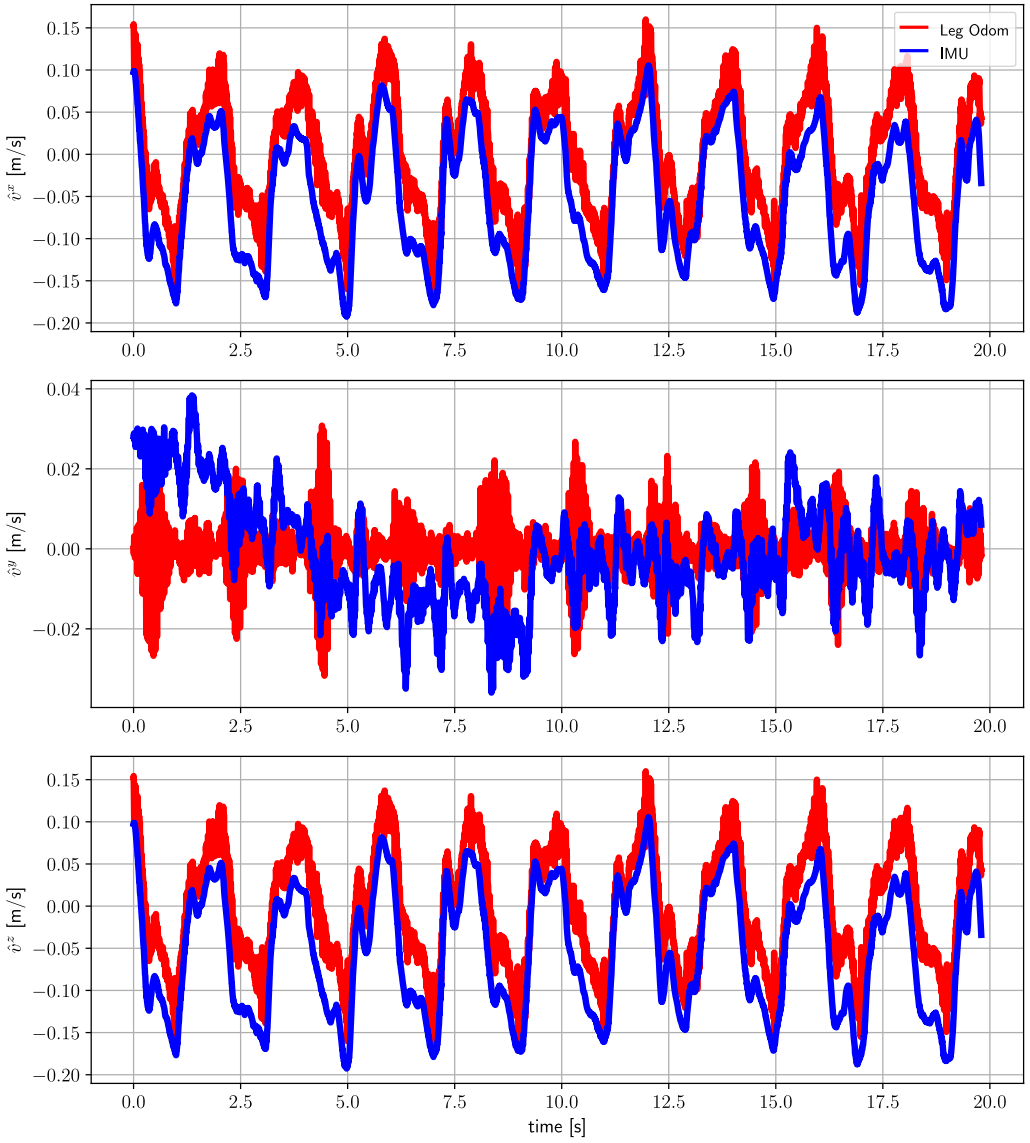



**Figure 4.5:** Comparison between velocity estimates using leg odometry and IMU integration along the forward, lateral and vertical directions of the world frame. Notice that the IMU-based estimate is less affected by noise.

## 4.4 Motion Control During Landing Phase

This section discusses the pWBC we employ to track in the landing phase. First, we design a Cartesian impedance, attached at the CoM, to track the CoM reference after TD, stabilize the orientation, and reject disturbances on both linear and angular directions. The control law will generate a wrench $\boldsymbol{w}_d \in \mathbb{R}^6$ that we map into desired GRFs $\boldsymbol{f}_{i,d} \in \mathbb{R}^3$, at the robot's feet [Focchi et al., 2017].

### Feedback Wrench

A PD feedback term is computed to track the CoM reference and the base orientation:

$$\boldsymbol{w}_{fb} = \begin{bmatrix} \boldsymbol{w}_{fb}^{lin} \\ \boldsymbol{w}_{fb}^{ang} \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_P^{lin}\left(\boldsymbol{c}_d - \boldsymbol{c}\right) + \boldsymbol{K}_D^{lin}\left(\dot{\boldsymbol{c}}_d - \dot{\boldsymbol{c}}\right) \\ \boldsymbol{K}_P^{ang}\,\boldsymbol{e} + \boldsymbol{K}_D^{ang}\left(\boldsymbol{\omega}_d - \boldsymbol{\omega}\right) \end{bmatrix} \tag{4.18}$$

where $\boldsymbol{e}$ is the orientation error (defined as in Appendix E) and $\boldsymbol{\omega}$ is the actual angular velocity. The desired angular velocity is obtained with the appropriate mapping from the desired Euler rates (see (B.16)):

$$\boldsymbol{\omega}_d = \boldsymbol{T}\left(\boldsymbol{\phi}_d\right)\dot{\boldsymbol{\phi}}_d.$$

### Feed-forward Wrench

We annihilate the effects of the gravity force on the CoM through the feed-forward component

$$\boldsymbol{w}_g = \begin{bmatrix} \boldsymbol{w}_g^{lin} \\ \boldsymbol{w}_g^{ang} \end{bmatrix} = \begin{bmatrix} m\boldsymbol{g} \\ \boldsymbol{0}_{1\times 3} \end{bmatrix}.$$

To improve the tracking performances, desired accelerations enter the controller with a feed-forward term

$$\boldsymbol{w}_{ff} = \begin{bmatrix} \boldsymbol{w}_{ff}^{lin} \\ \boldsymbol{w}_{ff}^{ang} \end{bmatrix} = \begin{bmatrix} m\ddot{\boldsymbol{c}}_d \\ {}_{\mathcal{T}}\boldsymbol{R}_{\mathcal{C}}\,{}_{\mathcal{C}}\boldsymbol{I}_{\mathcal{C}}\,{}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}}\dot{\boldsymbol{\omega}}_d \end{bmatrix} \tag{4.19}$$

where $\dot{\boldsymbol{\omega}}_d$ is deduced from the desired Euler angles and rates.

### Whole Body Controller

After TD, the robot has all the feet in contact with the ground, and we map the desired wrench

$$\boldsymbol{w}_d = \boldsymbol{w}_{fb} + \boldsymbol{w}_g + \boldsymbol{w}_{ff}$$

to the stack of desired GRFs $\boldsymbol{f}_d \in \mathbb{R}^{3n_c}$ by solving the QP

$$\begin{aligned} \boldsymbol{f}_d = \underset{\boldsymbol{f}_d}{\operatorname{argmin}} \; & \frac{1}{2}\|\boldsymbol{G}\boldsymbol{f}_d - \boldsymbol{w}_d\|^2 \\ \text{s.t.} \quad & |f^x_{i,d}| \leq \mu f^z_{i,d}, \qquad i = 1, \ldots, n_c \\ & |f^y_{i,d}| \leq \mu f^z_{i,d}, \\ & f^z_{i,d} > 0 \end{aligned} \tag{4.20}$$

that enforces **(R4)** by means of the linearized friction cone constraints, cf. (B.40). We set the friction coefficient $\mu \in \mathbb{R}^+$ to be equal for all the contacts. $\boldsymbol{G} \in \mathbb{R}^{6\times 3n_c}$ denotes the grasp matrix

$$\boldsymbol{G} = \begin{bmatrix} \cdots & \boldsymbol{I}_{3\times 3} & \cdots \\ & [\boldsymbol{p}_i - \boldsymbol{c}]_\times & \end{bmatrix}.$$

Finally, the desired torque to be exerted by the joints of the $i$-th leg is

$$\boldsymbol{\tau}_{i,d} = \boldsymbol{S}_i \, \boldsymbol{C}(\hat{\boldsymbol{q}}) \, \dot{\hat{\boldsymbol{q}}} - {}_{\mathcal{B}}\boldsymbol{J}_{p,i}\,(\hat{\boldsymbol{q}})\, \boldsymbol{f}_{i,d}. \tag{4.21}$$

## 4.5 Simulations and Experiments

In this section, we present relevant information about the implementation of the LC framework. The code to replicate the results is open source and can be downloaded at github.com/iit-DLSLab/reactive_landing_controller/. The remainder of this section will showcase results obtained from simulations and experiments, thereby validating our research approach.

### 4.5.1 Implementation details

The robot we use to demonstrate the validity of our approach is the torque-controlled quadruped Unitree Go1 Edu [Unitree, 2021]. To visualize, simulate, and interact with the robot, we use Locosim which is a platform-independent software framework designed for fast code prototyping, see Appendix A. We

implemented both the pWBC and the LC in a Python ROS node, which relies on Pinocchio [Carpentier et al., 2019] for the computation of robot kinematics and dynamics and closes the loop at 500 Hz. The firmware of Go1 supports external inputs via User Datagram Protocol (UDP) through an Ethernet connection. We coded the VHSIP dynamics in C++ to increase computational efficiency and provided Python bindings. References are computed with a frequency of 250 Hz and linearly interpolated to match the controller frequency of 500 Hz. Since Python is not real-time compliant, monitoring the loop frequency in simulation is essential before running the framework with the real hardware. A number of tests are reported in the attached video available at `https://youtu.be/wiuedeHfSEY`. For all the cases, we set the parameters reported in Table 4.1.

**Table 4.1:** Parameters for LC framework.

| Parameter | Meaning | Value |
|---|---|---|
| $l_0$ | Nominal robot height | 0.25 m |
| $\Delta^z$ | Minimum ground–$c^z$ distance | 0.10 m |
| $t_c$ | Maximum settling time | 1.2 s |
| $f_{th}^n$ | Normal force threshold | 5 N |
| $w^x$, $w^y$, $w^z$ | Leaky integrator discount factors | 0.21 Hz |
| $\mu$ | Friction coefficient | 0.8 |

### 4.5.2 Limits on the Horizontal Velocity

To emphasize the advantages of the proposed LC, we first compared it in simulation with a *naive approach* that keeps the feet at a constant position on the $X_{\mathcal{T}}Y_{\mathcal{T}}$–plane during the flying phase. The two controllers have the same reference for the CoM in the vertical direction, the same joint PD gains and the same pWBC gains. The main difference relies on the absence, in the naive approach, of the correction due to the horizontal component of the CoM velocity that is present in our LC, i.e., (4.12). Here, we consider the landing task *achieved* only if the feet make contact with the ground and the robot reaches a standing still configuration, i.e., joint velocities below a threshold, without bouncing. Our goal is to show that the modulation of the virtual foot $\boldsymbol{u}_o$ is crucial to achieving a successful landing. By executing thousands of automated drops with varying the initial conditions of (4.1)-(4.3), we detected

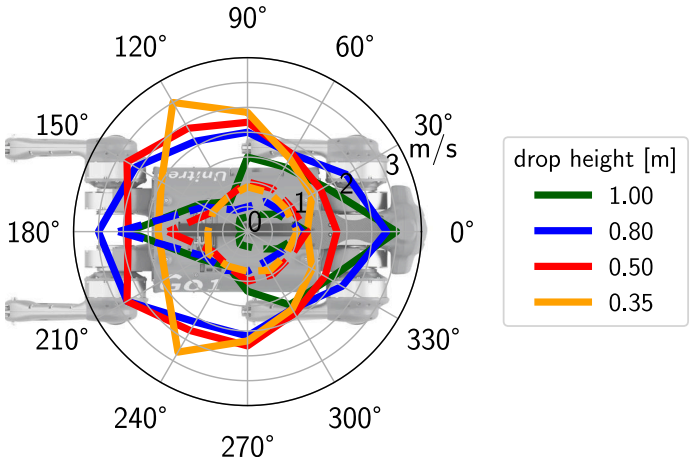



**Figure 4.6:** Simulation result: maximum attainable horizontal velocity. Polar plot of the limit magnitudes of $\dot{\boldsymbol{c}}_0^{x,y}$ for our LC (solid lines) and for the naive approach (dashed lines) with different drop heights. The angles represent the direction of $\dot{\boldsymbol{c}}_0^{x,y}$ with respect to the $X_{\mathcal{T}}$-axis, i.e., atan2 $(\dot{c}_0^y, \dot{c}_0^x)$.

the ranges of TD horizontal velocities that can be handled using either our LC or the naive controller, that are reported in the polar chart in Figure 4.6. We tested dropping horizontal velocities in the form

$$\dot{\boldsymbol{c}}_0^{x,y} = \nu \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix},$$

with $\nu = 0.0, 0.1, \ldots, 4.0$ m/s and $\psi = 0, \pi/6, \ldots, 2\pi$ rad. For all the tested dropping heights (0.35, 0.50, 0.80, 1.00 m), the LC can handle larger initial horizontal velocities in all the directions, up to 3.0 m/s. Notice that the maximum trotting speed of Go1 is 3.7 m/s, confirming our interest in landings with high horizontal velocity. The asymmetry of the polytopic regions results from the asymmetric mass distribution and joint limits. As the naive approach often fails in simulations, we decided not to test it with any experiment because it would weaken the robot's structure.

### 4.5.3 Robustness to Noise

When dealing with real hardware, it is common to have noisy measures and uncertain parameters. To understand the robustness of our controller in simulation, we added white noise to measured joint velocities, measured actuation

torques, and initial horizontal velocity. The values for the standard deviation are $\sigma_{\dot{q}} = 0.05$ rad/s, $\sigma_{\tau} = 0.2$ Nm and $\sigma_{\dot{c}^{x,y}} = 0.2$ m/s, respectively. We throw the robot from a height of 0.80 m. In order to produce a statistical analysis, we execute a batch of 10 simulations for each true value of the initial horizontal velocity. We compute the success rate within each batch as the percentage of achieved landing tasks; see the color map in Figure 4.7. Despite the noise, the overall success rate is 0.875.

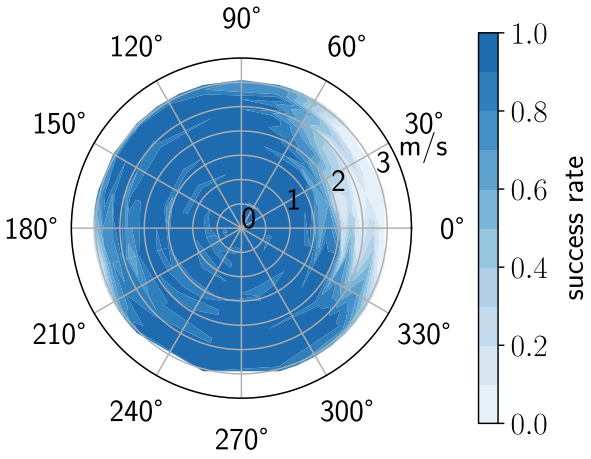



**Figure 4.7:** Simulation result: success rate for noisy inputs. Success rates dropping the Go1 robot from 0.80 m with various initial horizontal velocities. White noise affects measured joint velocities, measured joint torques, and initial horizontal velocities.

### 4.5.4 Angular perturbations

Here, we want to evaluate how tolerant our method is to angular perturbations despite our template model not describing the angular motion (see Section 4.2.3). In the simulation environment, we tested the robustness of our LC with respect to non-zero initial angular velocity and non-horizontal orientation of the trunk. We drop the robot from a height of 0.60 m (about 2.5 times its standing height) with a forward velocity of 1.0 m/s. One at a time, we vary the initial values of angles and rates. The discretization step is 5° for the angles and 5°/s for the rates. The limit values for which our LC is able to achieve the landing task are listed in Table 4.2.

**Table 4.2:** Simulation result: angular perturbations limits dropping Go1 from 0.6 m height with 1.0 m/s forward velocity. The asymmetry of the robot inertia causes the roll angle and rate bounds to be asymmetric.

| | min | max | unit |
|---|---|---|---|
| Roll angle | -40 | 35 | [°] |
| Pitch angle | -45 | 15 | [°] |
| Roll rate | -460 | 455 | [°/s] |
| Pitch rate | -365 | 215 | [°/s] |
| Yaw rate | -3035 | 3000 | [°/s] |

### 4.5.5 Experiments

We performed an extensive experimental study to assess the performance of our LC. All the tests are visible in the attached video. We dropped the 12 kg Go1 robot from various heights with different manually induced horizontal velocities in all directions (forward, backward, left, right). Figures 4.8 and 4.9 show snapshots and plots of a drop from about 0.8 m with non-zero horizontal ve-

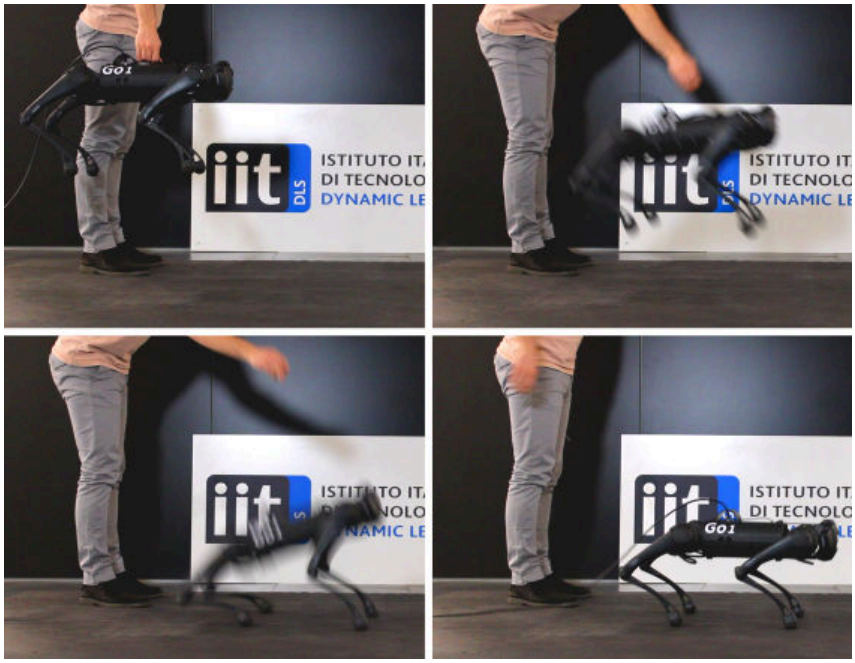


**Figure 4.8:** Experimental result: successful landing. Using the proposed LC, the Go1 robot can successfully achieve the landing task having a forward velocity of about 1.0 m/s during the flying phase.

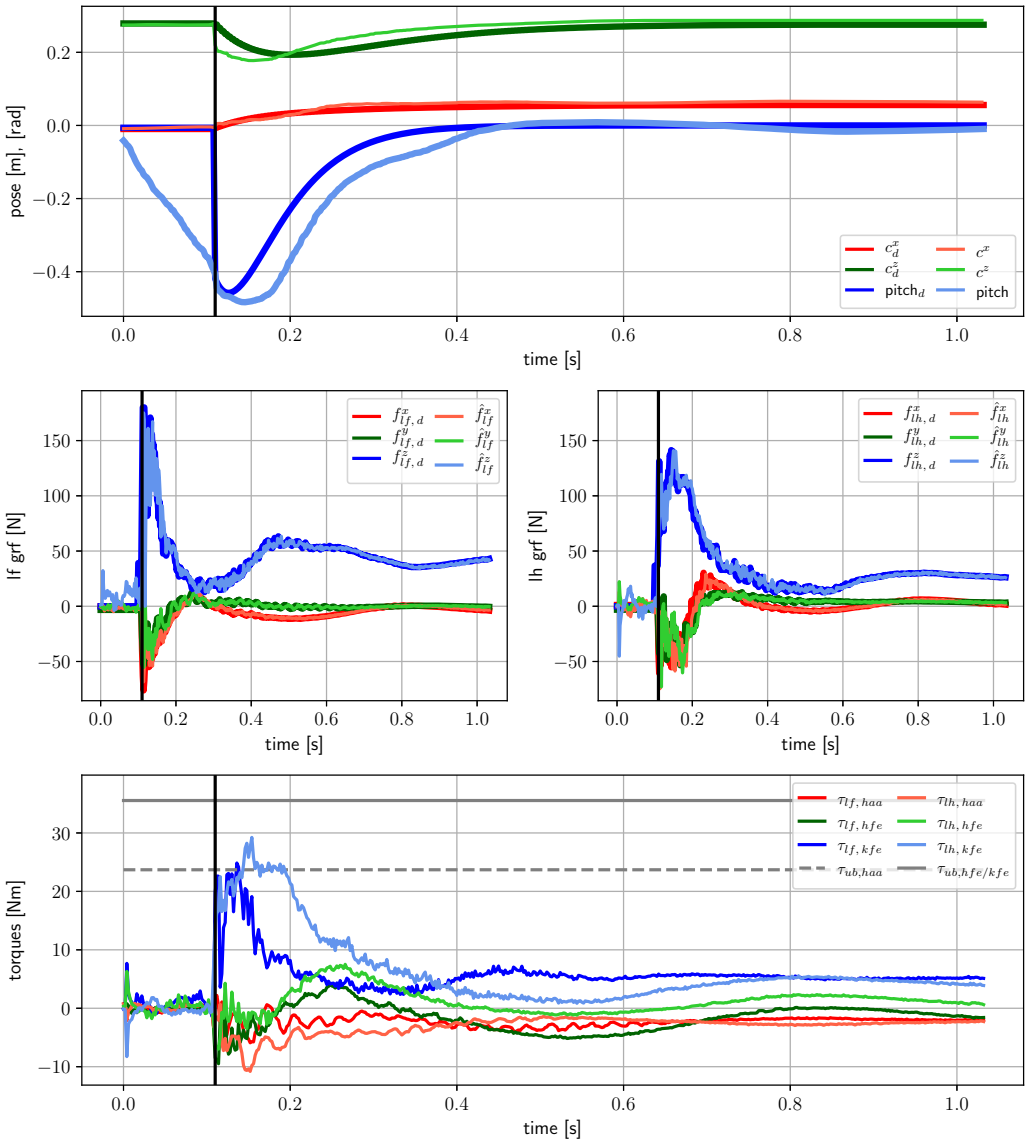



**Figure 4.9:** Experimental results: successful landing. Top: desired and actual CoM and pitch trajectories expressed in $\mathcal{T}$-frame. Left center: desired and estimated GRFs exerted on the left front foot. Right Center: desired and estimated GRFs exerted on the left hind foot. Bottom: measured torques for the left legs. The upper actuation limit for HAA mechanisms is $\tau_{ub,\,haa}$ and for both HFE and KFE ones is $\tau_{ub,\,hfe/kfe}$. Lower limits are mirrored (not reported in figure). The vertical lines denote the detection of the TD.

locity in the forward direction. They illustrate that, using only proprioceptive measures, the robot can be successfully stabilized to a standing configuration thanks to both the kinematic adjustment and the GRFs exerted after landing. Notice that joint velocities and torques never reach their limits in all the tests.

## 4.6 Summary

This chapter presented a model-based approach for a quadruped robot landing after unexpected falls or planned drops. A successful landing entails dissipating all the kinetic energy, stopping the robot with the trunk without hitting the ground, and keeping the feet in contact. Tracking the VHSIP model turned out to be a suitable candidate to avoid multiple bounces after landing **(R1)**, to prevent undesired collisions of the trunk with the ground **(R2)**, to reach a stable standing state **(R3)**, and to maintain the feet fixed after TD **(R4)**. The LC promptly reacts (500 Hz) to a fall by adjusting the limb configuration during the flying phase and tracking the trajectory optimized with the VHSIP model. In this way, the robot successfully dissipates the excess of kinetic energy, avoiding rebounds. Making use only of proprioceptive measurements, the proposed real-time framework is capable of handling substantial horizontal velocity, insensitive to TD timing uncertainties, and independent of the need to estimate the robots absolute pose and velocity. This last point makes it suitable for real world applications where external motion capture systems may not be available. Test conducted with the quadruped robot Go1 validated the proposed method either in the simulation environment and with real hardware, even in the presence of unmodeled angular perturbations and noisy inputs.

# 5

# Extended Landing Controller

*When a quadruped robot experiences a free fall with a horizontal velocity exceeding the maximum the LC can stabilize, achieving a stable standing state becomes challenging, potentially resulting in the robot's fall and hardware damage. Recognizing the constraints imposed by the robot's kinematics, actuation capabilities, and friction limits, the Extended Landing Controller (ELC) offers a possible solution to the problem considering kinetic energy dissipation over multiple landings. Specifically, during the flying phase, it computes references to be executed during the landing phase that prevent collisions between the robot's trunk and the ground and introduce an additional flying phase with reduced horizontal velocity. Preliminary findings, as demonstrated with Go1, suggest that if reducing the horizontal velocity during the landing phase is feasible, the combined utilization of ELC and LC can eventually guide the robot toward a stable standing posture. This approach acknowledges practical challenges and constraints, offering a promising path to enhance the robot's safety and recovery in extreme and unpredictable scenarios.*

Simulation and experiments using the Landing Controller (LC) framework show that it can provide a stabilizing control law only if the robot's horizontal velocity during the flying phase is within certain bounds. These are dictated by the possibility of setting the CP within the SP, as requested by the kinematic adjustment. The capturability framework [Koolen et al., 2012] establishes that if it is not possible to put the CoP on the CP, the CoM will converge towards the CP that is diverging away from the CoP, resulting in miserable failure of the landing strategy. Therefore, a different control strategy is necessary.

This chapter aims to explore the implications of kinematic, actuation, and frictional constraints on the landing of a quadruped robot. It addresses a critical question: what course of action should the robot undertake when faced with the inability to achieve kinematic adjustments? The main contributions of this chapter are highlighted in the following.

- We define the feasibility of the kinematic adjustment action within the LC framework. Furthermore, we introduce a practical criterion to evaluate it, leveraging the concept of improved feasible region.
- We relax the virtual constraint of placing the feet centroid on the CP. Indeed, we verify that as long as the CP is confined inside the SP at the landing phase, the LC can still accomplish the landing task, and no further investigation is needed. The price to pay is that there will be less robustness margin for the pWBC if tracking errors are present.
- In instances where the above condition is not met, we propose the Extended Landing Controller (ELC) as a modification of the LC framework. This reflex strategy reduces the horizontal velocity during the landing phase (instead of trying to nullify it) and incorporates an additional flying phase. The combined use of ELC and LC expands the range of initial horizontal CoM velocities that controllers can accommodate. For instance, the proposed approach successfully stabilizes in simulation the motion of the Go1 robot dropped from a height of 0.8 m with a velocity of 4.0 m/s with two Touch Down (TD).

The rest of this chapter is organized as follows. Section 5.1 presents some background information about CoM reachability and feasibility once the feet location is given. Exploiting the bounds on the joint position, the limits on the joint torques, and the friction cone constraints, we state the feasibility of the kinematic adjustment as presented in Section 4.2.4 and we define when its unfeasibility compromises the ability of the LC in leading the robot to a

stable standing state after the TD. Section 5.2 considers the problematic cases for which the CP is not inside the SP post-TD. Here, we present our planning technique, based on Bézier curves, for CoM trajectory during the landing phase that produces an additional landing phase. Section 5.3 reports the iterative algorithm at the core of the ELC framework for compressing the horizontal velocity between the first TD and the subsequent LO, leading to a feasible CoM trajectory. Implementation details and simulations validating our line of research are described in Section 5.4. Eventually, Section 5.5 summarizes the main contents of the chapter and offers some food for thought.

## 5.1 Background

In this section, we describe three distinct regions about the CoM once the position of the feet in contact is established. The *reachable region* [Abdalla et al., 2023] denotes the set of positions the CoM can reach considering the robot kinematics and the joint limits. The *feasible region* [Orsolino et al., 2020] encompasses positions attainable by the CoM within the bounds of actuation limits and friction cone constraints. Lastly, the *improved feasible region* [Abdalla et al., 2023] emerges as the intersection between the other two regions. A CoM position/trajectory inside these regions is guaranteed to be feasible regarding kinematic, actuation, and friction limits. The section culminates by leveraging this latter region to assess the feasibility of a kinematic adjustment.

### 5.1.1 Reachable Region

When the position of the feet in contact and the trunk orientation are determined, a distinct closed set emerges, encompassing all possible CoM locations that adhere to the joint kinematic constraints. This set is termed the reachable region [Abdalla et al., 2023]. Essentially, it represents in the task space, i.e., the Cartesian space for the CoM, the robot's kinematic characteristics established within the joint space. For each limb $i$, the direct kinematic function maps the limbs joints $\hat{\boldsymbol{q}}_i$ to the foot position expressed with respect to the base frame ${}_{\mathcal{B}}\boldsymbol{p}_i$:

$$ {}_{\mathcal{B}}\boldsymbol{p}_i = {}_{\mathcal{B}}\mathbf{dkf}_{pi}\left(\hat{\boldsymbol{q}}_i\right), \qquad i = 1, \ldots, n_c. \tag{5.1} $$

Without loss of generality, we utilized the understanding that the kinematics of the foot on the $i$-th leg is solely influenced by the configuration of its joints

$\hat{\boldsymbol{q}}_i$. Assuming that the foot position with respect to an inertial frame (that in the case of the LC is the $\mathcal{T}$-frame) $\boldsymbol{p}_i$ is given, the foot position with respect to the base frame can also be computed as

$$ {}_{\mathcal{B}}\boldsymbol{p}_i = {}_{\mathcal{B}}\boldsymbol{R}\left(\boldsymbol{p}_i - \boldsymbol{c}\right) + {}_{\mathcal{B}}\boldsymbol{c}, \tag{5.2} $$

being ${}_{\mathcal{B}}\boldsymbol{c}$ the CoM position expressed in base frame. Equating (5.1) and (5.2), and solving for the CoM position in the inertial frame, we get $n_c$ equations

$$ \boldsymbol{c} = \boldsymbol{F}_i\left(\hat{\boldsymbol{q}}_i,\, \boldsymbol{p}_i,\, {}_{\mathcal{B}}\boldsymbol{R}\right), \qquad \forall i = 1, \ldots, n_c \tag{5.3} $$

where the $\boldsymbol{F}_i$'s are defined by

$$ \boldsymbol{F}_i\left(\hat{\boldsymbol{q}}_i,\, \boldsymbol{p}_i,\, {}_{\mathcal{B}}\boldsymbol{R}\right) = \boldsymbol{p}_i - {}_{\mathcal{B}}\boldsymbol{R}\left(\mathbf{dfk}_{pi}\left(\hat{\boldsymbol{q}}_i\right) - {}_{\mathcal{B}}\boldsymbol{c}\right). \tag{5.4} $$

Hence, considering the foot placement of the $i$-th leg $\boldsymbol{p}_i$ and a trunk orientation ${}_{\mathcal{B}}\boldsymbol{R}$, (5.3) establishes a connection between the joint configuration of that leg $\hat{\boldsymbol{q}}_i$ and the whole system's CoM position $\boldsymbol{c}$. It follows that a given CoM position is reachable if, for each leg, there exists a joint configuration such that

1. $\hat{\boldsymbol{q}}_i$ is mapped to $\boldsymbol{c}$;
2. $\hat{\boldsymbol{q}}_i$ is within its kinematic bounds;
3. there is not loss of mobility at $\hat{\boldsymbol{q}}_i$.

Exploiting the above conditions, given the feet location $\boldsymbol{p}_i$ and the trunk orientation ${}_{\mathcal{B}}\boldsymbol{R}$, the reachable region is defined as

$$ \begin{aligned} \mathcal{Y}_r\left(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_4,\, {}_{\mathcal{B}}\boldsymbol{R}\right) = \{\boldsymbol{c}^{x,y} \in \mathbb{R}^2 \mid \forall i = 1, \ldots, 4 \quad \exists \hat{\boldsymbol{q}}_i \quad \text{such that} \\ \boldsymbol{c}^{x,y} = \boldsymbol{F}^{x,y}\left(\hat{\boldsymbol{q}}_i,\, \boldsymbol{p}_i,\, {}_{\mathcal{B}}\boldsymbol{R}\right), \\ \hat{\boldsymbol{q}}_{lb,\,i} \leq \hat{\boldsymbol{q}}_i \leq \hat{\boldsymbol{q}}_{ub,\,i}, \\ {}_{B}\boldsymbol{J}_{pi}\left(\hat{\boldsymbol{q}}_i\right) \text{ is full rank}\}, \end{aligned} \tag{5.5} $$

where $\hat{\boldsymbol{q}}_{lb,\,i}$ and $\hat{\boldsymbol{q}}_{ub,\,i}$ are, respectively, the vectors containing lower and upper bounds for the joints on the $i$-th limb and the symbol $\leq$ is meant to be applied element-wise. Notice that the reachable region is a connected set because of the rank condition on the translational Jacobian matrices. Moreover, it is bounded because of the kinematic limits and the boundedness of $\boldsymbol{F}_i$'s.

In general, the sets $\mathcal{Y}_r$ are nonconvex, and the problem of finding them accurately is time-consuming. Numerous methodologies have emerged to ascertain the workspace of manipulators, utilizing analytic, geometric, or numerical

methods. Analytic and geometric techniques may lead to convoluted analysis confined to a singular platform. Numerical approaches either employ joint space sampling with forward kinematics or task space sampling with inverse kinematics. [Abdalla et al., 2023] applies the latter strategy, providing an algorithm that produces an approximation of $\mathcal{Y}_r$ but remains versatile across diverse platforms. Taking inspiration from the ray-casting algorithm, a discretized search is conducted iteratively in ordered directions starting from the feet centroid and using polar coordinates $(\rho, \theta)$. The user has to provide the radial and angular steps for the search $\Delta\rho$ and $\Delta\theta$. The process generates a 2D polygon outlining the boundary of the polygonal approximation of the reachable region. In the remainder of this chapter, we use the symbol $\mathcal{Y}_r$ for referring either to the actual region or to the polygonal approximation.

Figure 5.1 illustrates slices of the reachable region of Go1 for various heights, as obtained by setting

- the trunk is horizontal: it has zero roll, pitch, and yaw, and ${}_{\mathcal{B}}\boldsymbol{R} = \boldsymbol{I}_{3\times3}$;
- the feet positions are the ones the robot assumes when in home configuration:

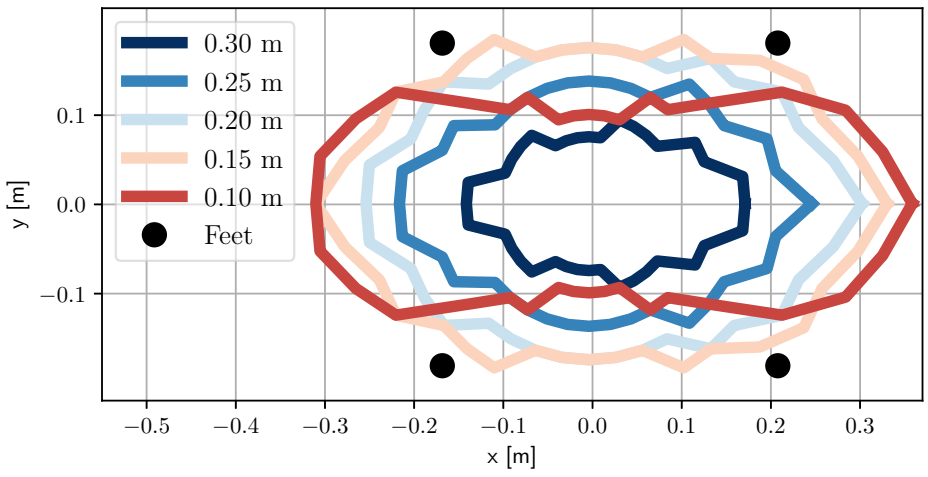



**Figure 5.1:** Each polygon represents the boundary of the reachable region of Go1 for different robot heights, considering the trunk having zero roll, pitch, and yaw and the feet in home configuration. Supporting the intuition, the regions become smaller if the robot's legs are fully extended.

$$\begin{aligned} \boldsymbol{p}_{LF,0} &= \begin{bmatrix} 0.208 \\ 0.181 \\ 0.000 \end{bmatrix} \text{ m}, & \boldsymbol{p}_{RF,0} &= \begin{bmatrix} 0.208 \\ -0.181 \\ 0.000 \end{bmatrix} \text{ m}, \\ \boldsymbol{p}_{LH,0} &= \begin{bmatrix} -0.168 \\ 0.181 \\ 0.000 \end{bmatrix} \text{ m}, & \boldsymbol{p}_{RH,0} &= \begin{bmatrix} -0.168 \\ -0.181 \\ 0.000 \end{bmatrix} \text{ m}; \end{aligned} \tag{5.6}$$

- the CoM height ranges from 0.10 m to 0.30 m with step $\Delta c^z = 0.05$ m;
- the steps for the search are $\Delta\rho = 0.02$ m and $\Delta\theta = 10.0°$.

### 5.1.2 Feasible Region

The feasibility of a CoM trajectory depends not only on the limitations imposed by the kinematic structure of the robot but also on the friction bounds and on the torques the actuators can deliver. [Orsolino et al., 2020] introduced the notion of feasible region[1] as the set of CoM positions that are consistent with friction and actuation limitations, assuming static conditions as a simplification. Under such assumption, the Newton-Euler equations (2.2),(2.3) simplifies as

$$\sum_{i=1}^{n_c} \boldsymbol{f}_i = -m\boldsymbol{g}, \tag{5.7}$$

$$\sum_{i=1}^{n_c} [\boldsymbol{p}_i]_\times \boldsymbol{f}_i - m\,[\boldsymbol{g}]_\times \boldsymbol{c} = \boldsymbol{0}, \tag{5.8}$$

with $\boldsymbol{g} = \begin{bmatrix} 0 & 0 & -g^z \end{bmatrix}^T$. Furthermore, considering the $i$-th leg, there exists a connection between the GRF on its foot $\boldsymbol{f}_i$ and the joint torques of its actuators $\boldsymbol{\tau}_i$. In static conditions, it is established by

$$\boldsymbol{g}_i(\hat{\boldsymbol{q}}_i) = \boldsymbol{\tau}_i + \boldsymbol{J}_{Ti}^T(\hat{\boldsymbol{q}}_i)\,\boldsymbol{f}_i, \qquad i = 1, \ldots, n_c, \tag{5.9}$$

where the index is used to identify quantities related to the $i$-th limb, i.e., $\boldsymbol{g}_i(\hat{\boldsymbol{q}}_i) = \boldsymbol{S}_i\boldsymbol{g}(\hat{\boldsymbol{q}})$ with $\boldsymbol{S}_i$ the selection matrix introduced in (4.15). The GRF $\boldsymbol{f}_i$ is constrained by the friction cone, which ensures unilateral and non-slippage conditions: $\boldsymbol{f}_i \in \mathcal{FC}_k$, as reported in Appendix B.3.2. On the other side, also

[1] [Orsolino et al., 2020] introduces the feasible region for legged robot possibly having the number of contacts varying with the time, possibly with shape different form the pointy one considered in this dissertation. In this chapter, we delve only with 3D contact wrenches (linear forces) on feet constantly in touch with the ground, i.e., during the landing phase.

the joint torque $\boldsymbol{\tau}_i$ is bounded, due to the actuation limitations as

$$\boldsymbol{\tau}_{lb,\,i} \leq \boldsymbol{\tau}_i \leq \boldsymbol{\tau}_{ub,\,i} \tag{5.10}$$

being $\boldsymbol{\tau}_{lb,\,i}$ ($\boldsymbol{\tau}_{ub,\,i}$) the vector collecting lower (upper) bounds for the torques. It is possible to collect all this information into a closed 2D set of all the allowed CoM positions $\boldsymbol{c}^{x,\,y}$, that is the feasible region:

$$\begin{aligned}\mathcal{Y}_{fa} = \{&\boldsymbol{c}^{x,\,y} \in \mathbb{R}^2 \mid \forall i = 1,\,\ldots,\,4,\ \exists \boldsymbol{f}_i \in \mathbb{R}^3,\ \boldsymbol{\tau}_i \in \mathbb{R}^3 \text{ such that} \\ &(\boldsymbol{c}^{x,\,y},\ \boldsymbol{f}_i,\ \boldsymbol{\tau}_i) \text{ satisfies (5.7), (5.8), (5.9)} \\ &\boldsymbol{f}_i \in \mathcal{FC}_i \\ &\boldsymbol{\tau}_{lb,\,i} \leq \boldsymbol{\tau}_i \leq \boldsymbol{\tau}_{ub,\,i}\}.\end{aligned} \tag{5.11}$$

Due to the convexity of the constraints, $\mathcal{Y}_{fa}$ is a convex set. [Orsolino et al., 2020] proposes an efficient algorithm for the computation of the feasible region based on Iterative Projection (IP) algorithm, originally employed by [Bretl and Lall, 2008] for the computation of support regions for articulated robots having multiple contacts with the environment in arbitrary locations.

Figure 5.2 depicts slices of the feasible regions of Go1 being in static con-

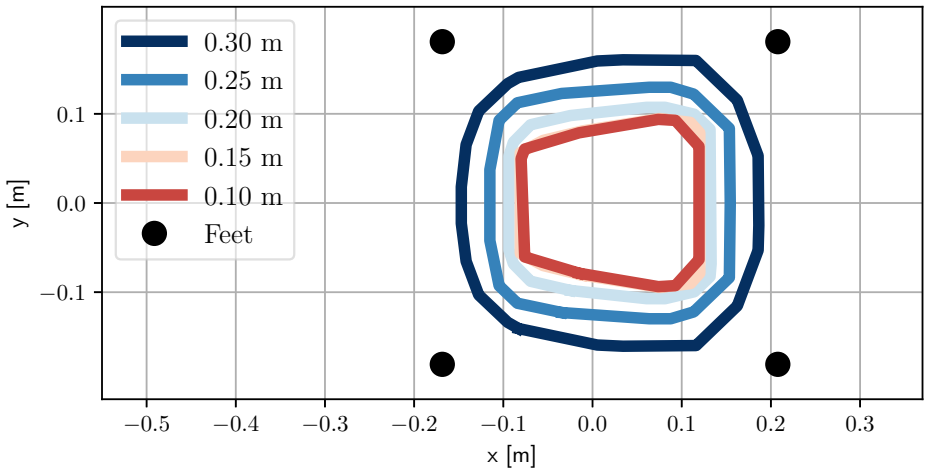



**Figure 5.2:** Each polygon represents the boundary of the feasible region of Go1 for different heights, considering the trunk having zero roll, pitch, and yaw and the feet in home configuration.

ditions with the feet as in (5.6), considering different CoM heights. Notice that the region expands by increasing the robot's height. Indeed, a stretched leg is characterized by torques with a more advantageous arm, enabling it to generate contact forces that withstand gravity more effectively. Consequently, it can endure greater loading, allowing for larger shifts in the CoM.

### 5.1.3 Improved Feasible Region

In the previous pages, we introduced two useful sets defined on the same 2D space:

- the reachable region $\mathcal{Y}_r$ characterizes the achievable positions of the CoM based on the robot's kinematics;
- the feasible region $\mathcal{Y}_{fa}$ delineates the admissible locations of the CoM while considering frictional and actuation limitations.

Planning of feasible CoM trajectories must be aware of the limitations imposed by kinematics, actuation, and friction. Therefore, [Abdalla et al., 2023] introduces the improved feasible region as

$$\mathcal{Y}_{far} = \mathcal{Y}_r \cap \mathcal{Y}_{fa}. \tag{5.12}$$

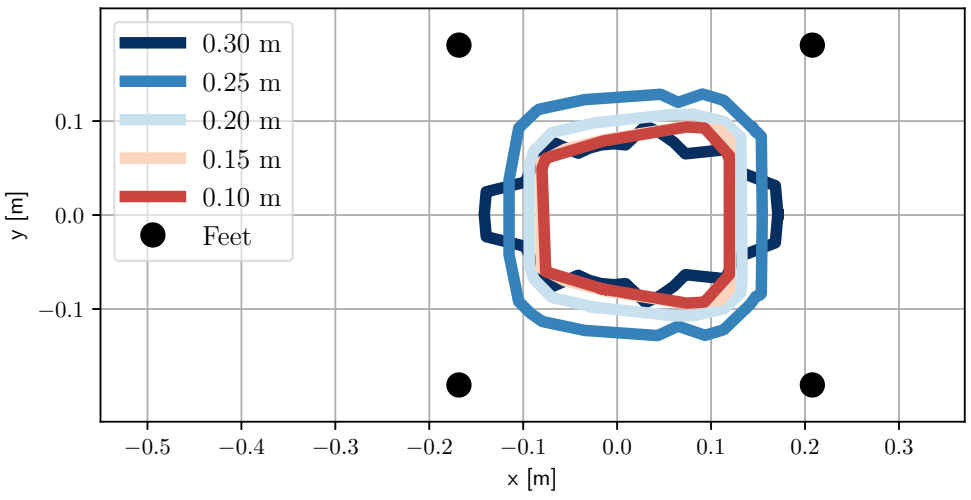



**Figure 5.3:** Each polygon represents the boundary of the improved feasible regions of Go1 for different heights, considering the trunk having zero roll, pitch, and yaw and the feet in home configuration.

In other words, $\mathcal{Y}_{far}$ represents a comprehensive 3D region of all the feasible CoM positions that satisfy the friction constraints, the joint-torque constraints, and the joint-kinematic constraints simultaneously. Given that intersections of closed nonconvex sets with closed convex ones result in closed nonconvex sets, the improved feasible region is a closed nonconvex set.

Figure 5.3 shows the improved feasible region of Go1 for different heights, obtained as the intersection between the polygons in Figure 5.1 and Figure 5.2. Table 5.1 summarizes the nomenclature and the constraints tackled by the different regions introduced above.

**Table 5.1:** Types of 2D regions

| Name | Symbol | Constraints | Main Reference |
|---|---|---|---|
| Reachable Region | $\mathcal{Y}_r$ | Kinematic | [Abdalla et al., 2023] |
| Feasible Region | $\mathcal{Y}_{fa}$ | Friction, actuation | [Orsolino et al., 2020] |
| Improved Feasible Region | $\mathcal{Y}_{far}$ | Friction, actuation, kinematic | [Abdalla et al., 2023] |

### 5.1.4 Feasibility of a Kinematic Adjustment

We can employ the regions introduced above to verify whether the kinematic adjustment requested by the LC during the flying phase is feasible or not. Let us first establish a notion of feasibility for a kinematic adjustment. From here onward, all the coordinates, unless specified, are expressed in terrain frame $\mathcal{T}$, as defined in Section 4.2.

**Definition 5.1.** *Given a point $\boldsymbol{u} \in \mathbb{R}^3$, with $u^z = 0$, a CoM location $\boldsymbol{c} \in \mathbb{R}^3$ and a trunk orientation ${}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}} \in \mathrm{SO}(3)$, the kinematic adjustment associated to $(\boldsymbol{u}, \boldsymbol{c}, {}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}})$ is said to be (kinematically) feasible during the flying phase of an aerial motion if for each leg $i = 1, \ldots, 4$ there exists a joint configuration $\hat{\boldsymbol{q}}_i$, with $\hat{\boldsymbol{q}}_{lb,i} \leq \hat{\boldsymbol{q}}_i \leq \hat{\boldsymbol{q}}_{ub,i}$, for which the centroid $\boldsymbol{\gamma}$ of the feet locations $\boldsymbol{p}_i(\hat{\boldsymbol{q}}_i) = \boldsymbol{p}_{i,0} + \boldsymbol{u}$ coincides with $\boldsymbol{u}$.*

Since the LC framework assumes a fixed height for the CoM at TD, we introduce the following definition too.

**Definition 5.2.** *Given a trunk orientation ${}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}} \in \mathrm{SO}(3)$, the point $\boldsymbol{u} \in \mathbb{R}^3$ is a LC-feasible CP if the kinematic adjustment $\left(\boldsymbol{u}, \boldsymbol{c} = \begin{bmatrix} 0 & 0 & l_0 \end{bmatrix}^T, {}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}}\right)$ is feasible.*

Directly using the above definitions, we can state that the LC framework can be used to achieve a landing task if the point $\boldsymbol{u}_o$ (computed as in (4.9)) is LC-feasible, and therefore the CoM will converge above the feet centroid during the landing phase. Changing for a moment our focus from the terrain frame $\mathcal{T}$ to the robocentric frame $\mathcal{C}$, we can say that $\boldsymbol{u}$ is LC-feasible if the IK problem in (4.13) is solvable for all the feet. This observation makes it clear that there exists a bounded set of LC-feasible CP locations, and this is dictated by the trunk orientation and the relative position of the CoM and the feet. Moreover, if the CP location does not align with the feet's centroid during the flying phase, but it is confined in the feet support polygon, there's the possibility that the joint torques and the GRFs will not be sufficient to achieve it after the TD due to actuation and friction limits. The improved feasible region $\mathcal{Y}_{far,l_0}$ computed at the CoM height $l_0$ offers us a tool for checking whether the CP location is LC-feasible or not. Since the feet centroid is the starting point for the search of the bounds for the reachable regions, and therefore for the definition of the improved feasible region, for the sake of comprehension, we introduce an additional reference frame, see Figure 5.4. The frame $\Gamma$ has the axes oriented as the ones of $\mathcal{T}$, but its origin lies on the feet centroid $\boldsymbol{\gamma}$. The two frames coincide in the unique case in which $\boldsymbol{u}_o = \mathbf{0}$, which happens only for pure vertical falls. In this frame, the kinematic adjustment aims to move the CoM position by the same measure defined by the CP location but in the opposite direction, i.e.,

$$ {}_{\Gamma}\boldsymbol{c}_d^{\,x,\,y} = -\boldsymbol{u}_o^{x,\,y}. \tag{5.13} $$

For being reachable and feasible the desired CoM location must lie inside the improved feasible region: ${}_{\Gamma}\boldsymbol{c}_d^{\,x,\,y} \in \mathcal{Y}_{far,l_0}$. Therefore, these considerations imply that the kinematic adjustment $(\boldsymbol{u}_o,\, \boldsymbol{c}_d,\, {}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}})$ is feasible if $-\boldsymbol{u}_o^{x,\,y} \in \mathcal{Y}_{far,l_0}$ with $\boldsymbol{u}_o$ being LC-feasible. In the negative case (i.e., the interesting case for this chapter), it is impossible to further shift the feet polygon so that its centroid coincides with the CP location. We will have the CoM on the boundary of $\mathcal{Y}_{far,l_0}$ up to the TD event.

The key point here is that even if the landing task is not achievable in the LC-sense, because the robot will not come to a stable standing state over its SP centroid, the CP might still be inside the SP, and thus there exists a suitable set of GRFs able to produce a CoP that can be located on the CP. To summarize, we can have three different cases.

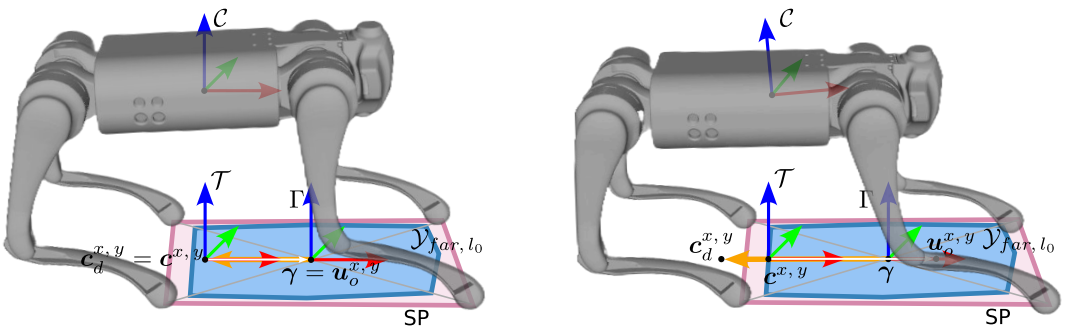



**Figure 5.4:** The frame $\Gamma$ is a translation of $\mathcal{T}$ in such a way that the feet centroid $\boldsymbol{\gamma}$ coincides with the origin of $\Gamma$. Left: the showed kinematic adjustment is feasible. Indeed, the CoM projection is confined within the feasible region $\mathcal{Y}_{far,l_0}$, and the CP coincides with the feet centroid. Right: the showed kinematic adjustment is not feasible. The robot cannot move the feet to have the CP on the centroid; therefore, the desired CoM position cannot be reached. The red, green, and blue arrows represent the $x$, $y$, $z$ axes of the CoM frame $\mathcal{C}$, of the terrain frame $\mathcal{T}$ and of the centroid frame $\Gamma$. Additionally, the yellow and white arrows denote ${}_{\Gamma}\boldsymbol{c}_d^{x,\,y}$ and $\boldsymbol{u}_o^{x,\,y}$, respectively. Notice that their lengths are equal, but they have opposite directions.

1. The CP is LC-feasible, then the kinematic adjustment sets the feet centroid on it, and the CoM stabilizes above it at the end of the landing phase. This is the case tackled by the LC framework.
2. The CP is not LC-feasible but lies inside the feet polygon. The robot is not able to complete the kinematic adjustment during the flying phase, but there exists a suitable set of GRFs that put the CoP on the CP after the TD. Consequently, the robot may stop at the end of the landing phase, albeit with the CoM not aligned above the feet centroid. In this case, the LC framework can still be employed, with a relaxation in the requirement for a stable standing state above $\boldsymbol{\gamma}$. To bolster the method's robustness, our approach considers the containment of $\boldsymbol{u}_o^{x,y}$ within a polygon smaller than that delineated by the feet: we refer to it with $\text{convHull}_{\zeta}\left(\boldsymbol{p}_i^{x,y}\right)$. This polygon shares the same centroid as the foot polygon but positions its vertices closer to the centroid by a margin value $\zeta \in \mathbb{R}^{+}$.
3. When the CP fails to meet the LC-feasibility and also lies outside the feet polygon, the robot is not stabilizable in only one landing phase. This is because the orbital energy is positive, resulting in the divergence

of the CoM. In such instances, the ELC framework supersedes the LC approach.

## 5.2 ELC Reference Planning

If the landing task is not achievable with the LC framework, we want to plan and track a reference CoM trajectory for the landing phase that uses the CP $\boldsymbol{u}_o^{x,y}$ for driving the robot to a new LO state with a lower horizontal velocity (velocity contraction). Therefore, the objective is to reduce the horizontal velocity, i.e., $\|\dot{\boldsymbol{c}}_{LO^+}^{x,y}\| \leq \|\dot{\boldsymbol{c}}_{TD}^{x,y}\|$ (the plus symbol indicates that the event is referred to the subsequent aerial motion). If $\dot{\boldsymbol{c}}_{LO^+}^{x,y}$, and thus also $\dot{\boldsymbol{c}}_{TD^+}^{x,y}$, is inside the feasibility bounds for the LC (cf. Figure 4.6), then the latter algorithm will drive the robot to a stable standing state at the end of the second aerial motion. Otherwise, another aerial motion is planned with $\|\dot{\boldsymbol{c}}_{LO^{++}}^{x,y}\| \leq \|\dot{\boldsymbol{c}}_{TD^+}^{x,y}\|$ and the process repeats. We call Extended Landing Controller (ELC) the framework for planning and tracking the new aerial motion. Retracing the steps for deriving the VHSIP model, we know that during the landing phase the horizontal dynamics are dictated by the CoP position, the CoM velocity at TD and the coupling with the vertical dynamics as

$$\ddot{\boldsymbol{c}}^{x,y} = \omega^2(t)\left(\boldsymbol{c}^{x,y} - \boldsymbol{u}^{x,y}\right) \tag{5.14}$$

with

$$\omega^2 \triangleq \frac{(g^z + \ddot{c}^z)}{c^z}.$$

In the VHSIP template, the vertical motion follows a MSD system with critically damped dynamics. This modeling approach is too restrictive to require an additional flying phase, and a different method is needed. As a matter of fact, we are looking for a vertical CoM reference, defined over the interval $[t_{TD},\ t_{LO^+}]$, that:

- starts from the nominal CoM height $c_d^z(t_{TD}) = l_0$ with the initial velocity defined by the landing task $\dot{c}_d^z(t_{TD})$;
- is lower bounded to avoid trunk collisions and is upper bounded because of the limbs kinematics;
- has limited acceleration due to actuation limits ;
- terminates with a final upward velocity allowing for an additional flying phase.

We opted for a polynomial reference. Being more flexible, this approach replaces the MSD used in the LC framework to generate vertical motion references. In particular, we decide to shape a Bézier curve, i.e., a polynomial of degree $N \in \mathbb{N}$, parameterized by $\sigma \in [0,\ 1]$, in the form

$$B_N(\boldsymbol{w};\ \sigma) = \sum_{i=0}^{N} \begin{pmatrix} N \\ i \end{pmatrix} w_i \left(1-\sigma\right)^{N-i} \sigma^i, \tag{5.15}$$

where $\boldsymbol{w} = \begin{bmatrix} w_0 & \dots & w_N \end{bmatrix}^T \in \mathbb{R}^{N+1}$ are the weighting factors (also named control points) and $\sigma^0$ and $(1-\sigma)^0$ are extended continuously to be 1 over $[0,\ 1]$. Bézier curves have the relevant properties of being differentiable, with the derivative being a Bézier curve of degree $N-1$:

$$\begin{aligned} B_N^{'}\left(\boldsymbol{w};\ \sigma\right) = \frac{\partial B_N}{\partial \sigma} &= \sum_{i=0}^{N-1} \begin{pmatrix} N-1 \\ i \end{pmatrix} w_i^{'} \left(1-\sigma\right)^{N-1-i} \sigma^i \\ &= B_{N-1}(\boldsymbol{w}';\ \sigma) \end{aligned} \tag{5.16}$$

where

$$w_i^{'} = N\left(w_{i+1} - w_i\right), \qquad i = 0,\ \dots,\ N-1. \tag{5.17}$$

The weighting factors of the higher-order derivatives can be found iteratively by applying (5.17), making the task of imposing boundary conditions particularly simple. The parameter $\sigma$ is used to set the time law. We decided to use linear interpolation, i.e.,

$$\sigma = \frac{t - t_{TD}}{T}, \qquad \text{for } t_{TD} \le t \le t_{LO^+} \tag{5.18}$$

being $T = t_{LO^+} - t_{TD}$. In this way, the time derivatives of a Beziér curve are

$$\begin{aligned} \dot{B}_N\left(\boldsymbol{w};\ \sigma\right) &= \frac{\partial B_N}{\partial \sigma}\frac{d\sigma}{dt} \\ &= \frac{1}{T} B_N^{'}\left(\boldsymbol{w};\ \sigma\right) = \frac{1}{T} B_{N-1}\left(\boldsymbol{w}^{'};\ \sigma\right) \\ \ddot{B}_N\left(\boldsymbol{w};\ \sigma\right) &= \frac{d}{dt}\left(\frac{1}{T} B_{N-1}\left(\boldsymbol{w}^{'};\ \sigma\right)\right) \\ &= \frac{1}{T^2} B_{N-1}^{'}\left(\boldsymbol{w}^{'};\ \sigma\right) = \frac{1}{T^2} B_{N-2}\left(\boldsymbol{w}^{''};\ \sigma\right) \\ &\vdots \\ \frac{d^\kappa}{dt^\kappa} B_N\left(\boldsymbol{w};\ \sigma\right) &= \frac{1}{T^\kappa} B_{N-\kappa}\left(\boldsymbol{w}^{\kappa};\ \sigma\right), \qquad \kappa \le N. \end{aligned} \tag{5.19}$$

Additionally, due to the linear dependence of the Beziér curve on the weighting factors, the following equality holds

$$\frac{1}{T^{\kappa}} B_{N-\kappa}\left(\boldsymbol{w}^{\kappa};\ \sigma\right) = B_{N-\kappa}\left(\frac{\boldsymbol{w}^{\kappa}}{T^{\kappa}};\ \sigma\right). \tag{5.20}$$

Specifically, we plan for a vertical motion reference using a Beziér polynomial of degree 5, allowing us to impose constraints that are a function of position, velocity, and acceleration and leaving additional degrees of freedom for the duration. Because the choice of the polynomial degree, here on we use $B_5\left(\boldsymbol{w};\ \sigma\right)$, $B_4\left(\frac{\boldsymbol{w}^{'}}{T};\ \sigma\right)$, $B_3\left(\frac{\boldsymbol{w}^{''}}{T^2};\ \sigma\right)$ to express the Beziér curves for position, velocity and acceleration, respectively. We find the values for the six weighting factors $w_0, \ldots, w_5$ and for the landing duration $T$ solving a Constraint Satisfaction Problem (CSP):

$$\begin{aligned}
&\text{find} && w_0, \ldots, w_5, T && && (5.21)\\
&\text{s.t.} && 0 < T \leq T_{ub} && && (5.22)\\
& && 0 \leq B_3\left(\frac{\boldsymbol{w}^{''}}{T^2};\ \sigma\right) \leq \ddot{c}^z_{ub} && \sigma \in \mathcal{I}_c && (5.23)\\
& && c^z_{lb} \leq B_5\left(\boldsymbol{w};\ \sigma\right) \leq c^z_{lb} && \sigma \in \mathcal{I}_c && (5.24)\\
& && B_4\left(\frac{\boldsymbol{w}^{'}}{T};\ 1\right) = \frac{w^{'}_4}{T} \geq 0 && && (5.25)\\
& && B_5\left(\boldsymbol{w};\ 0\right) = w_0 = l_0 && && (5.26)\\
& && B_4\left(\frac{\boldsymbol{w}^{'}}{T};\ 0\right) = \frac{w^{'}_0}{T} = \dot{c}^z_{TD} && && (5.27)
\end{aligned}$$

The problem requires that

- the landing duration must be not negative and upper bounded (5.22). The value of $T_{ub}$ will be discussed later;
- the vertical acceleration of the CoM is bounded (5.23). Indeed, it should not be negative since it would make the robot trunk vertically wobble during the landing phase. Moreover, it is upper-bounded due to actuation limitations. Instead of imposing an analytical rule for the weighting factors, we only checked such constraint on a discrete subset of control instants $\mathcal{I}_c \subset [0,\ 1]$. In particular, we use a discrete grid of step 0.1;
- the CoM height is limited to avoid either trunk collisions with the ground or overexertion of the limbs. Also (5.24) is imposed over the control

instants in $\mathcal{I}_c$;

- the final vertical CoM velocity, i.e., the vertical velocity at LO, must be positive to allow for an additional flying phase (5.25). This requirement differs from the LC setting: there, the vertical velocity at the end of a landing phase is always zero, because the MSD is asymptotically stable;
- the vertical initial position and velocity for the CoM height are imposed by the landing state (5.26), (5.27).

Notice that we can drop the dependency on $T$ in the constraints (5.25) since it is positive and, therefore, does not change the sign of the function involved.

## 5.3 ELC Loop and the Complete Framework

Here, we describe the pipeline for computing a feasible reference trajectory while the robot is in the descending part of the flying phase. Please refer to Algorithm 1. The computation starts using the LC framework and keeps using it up to the instant in which the kinematic adjustment is found to be unfeasible. Notice that, because of the linear interpolation introduced in (4.11), the kinematic adjustment cannot be identified as unfeasible right after the detection of the fall. This *smoothing* effect allows for moving the feet up to the limits. If the kinematic adjustment is feasible, then there is no need to change the strategy, and the pipeline keeps using LC. Otherwise, the controller executes the ELC loop that aims to find a feasible reference trajectory for the CoM $\boldsymbol{c}_d(t)$. In this case, we plan the horizontal dynamics using the VHSIP model in (5.14), while the vertical ones are expressed in the form of a Beziér polynomial of degree 5. The *virtual foot* of the VHSIP remains the CP computed by the LC optimization in (4.9) for the whole aerial motion. The coupling between vertical and horizontal CoM dynamics is set through the upper bound for the landing duration, $T_{ub}$. As a matter of fact, the minimum for the horizontal CoM velocity is reached at the time instant in which the CoM is above the CP, as established by (5.14). The algorithm loops until it finds a reference CoM trajectory that is contained in the improved feasible region $\mathcal{Y}_{far}$, with a LO horizontal velocity that is smaller than the TD one.

Figure 5.5 shows the resulting control framework of the ELC. It has very few differences from the one reported in Figure 4.2. As a matter of fact, the functionalities for detecting either the fall and the TD event are the same, as well as the state estimation. Nothing changes for tracking the references: IK and

PD employed during the flying phase, pWBC during the landing phase. Even the gains for the control laws can be the same. Additionally, there is no evidence of the need to change the angular motion reference. The noteworthy distinction between the two approaches regards the planning of the linear motion: ELC verifies the feasibility of the kinematic adjustment and plans for a feasible reference CoM trajectory.

**Algorithm** Extended Landing Controller

**Inputs:** $\dot{\boldsymbol{c}}_{TD}^{x,y}$, $\dot{c}_{TD}^{z}$
Use LC for computing $c_d^z(t)$, $\dot{c}_d^z(t)$, $\ddot{c}_d^z(t)$, $\omega^2(t)$, for optimizing $\boldsymbol{u}_o^{x,y}$ and for integrating $\ddot{\boldsymbol{c}}_d^{x,y} = \omega^2(t)\left(\boldsymbol{c}_D^{x,y} - \boldsymbol{u}_o^{x,y}\right)$
**if** $\boldsymbol{u}_o^{x,y}$ is LC-feasible **or** $\boldsymbol{u}_o^{x,y} \in \text{convHull}_\zeta\left(\boldsymbol{p}_i^{x,y}\right)$ **then**
  Keep using LC
**else**
  solved $\leftarrow$ False
  **while** solved $\neq$ True **do**
    $T_{ub} \leftarrow \underset{t}{\text{argmin}} \|\dot{\boldsymbol{c}}_d^{x,y}(t) - \varepsilon \dot{\boldsymbol{c}}_{TD}^{x,y}\|^2$
    Solve (5.21)-(5.27) for $w_0, w_1, \ldots, w_5, T$
    $t_{LO^+} \leftarrow t_{TD} + T$
    **for** $t_{TD} \leq t \leq t_{LO^+}$ **do**
      $\sigma \leftarrow \dfrac{t - t_{TD}}{T}$
      $c_d^z(t) \leftarrow B_5(\boldsymbol{w}; \sigma)$
      $\dot{c}_d^z(t) \leftarrow B_4\left(\dfrac{\boldsymbol{w}'}{T}; \sigma\right)$
      $\ddot{c}_d^z(t) \leftarrow B_3\left(\dfrac{\boldsymbol{w}''}{T^2}; \sigma\right)$
      $\omega^2(t) \leftarrow \dfrac{g^z + \ddot{c}_d^z}{c_d^z}$
    **end for**
    Integrate $\ddot{\boldsymbol{c}}_d^{x,y} = \omega^2(t)\left(\boldsymbol{c}_d^{x,y} - \boldsymbol{u}_o^{x,y}\right)$ for $t_{TD} \leq t \leq t_{LO^+}$
    **if** $\|\dot{\boldsymbol{c}}_{LO^+}^{x,y}\| \leq \|\dot{\boldsymbol{c}}_{TD}^{x,y}\|$ and $\boldsymbol{c}_d^{x,y}(t) \in \mathcal{Y}_{far,c_d^z(t)}$ for $t_{TD} \leq t \leq t_{LO^+}$ **then**
      solved $\leftarrow$ True
    **end if**
  **end while**
**end if**
Track $c_d^z(t)$, $\dot{c}_d^z(t)$, $\ddot{c}_d^z(t)$

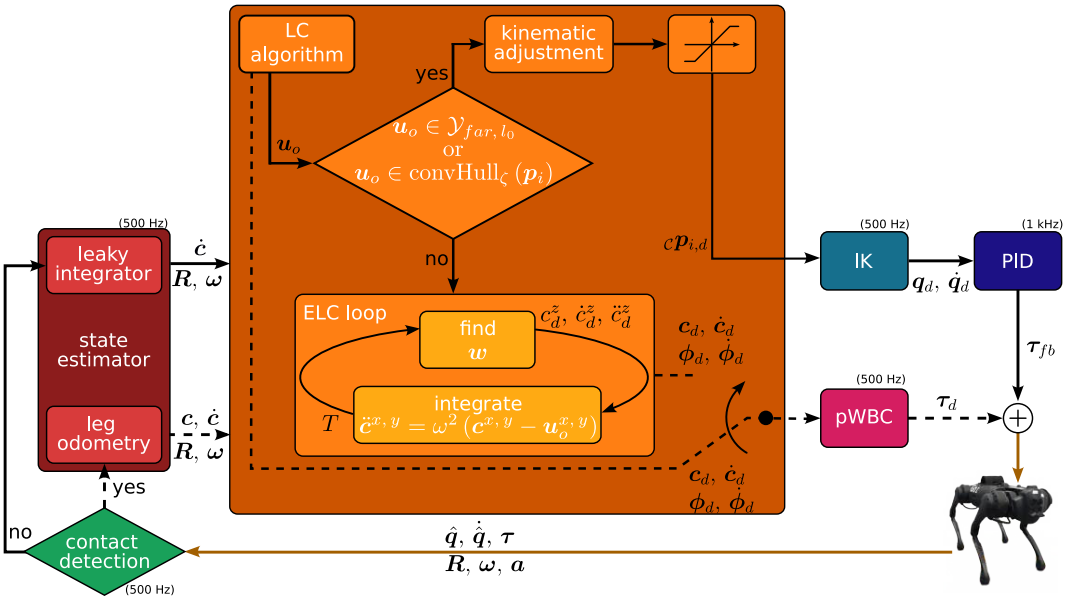



**Figure 5.5:** Overview of the ELC framework. When the robot is in the flying phase (solid black branches), the CP is continuously re-computed using LC approach, and the feet are shifted accordingly. If the CP is found not to be LC-feasible, the ELC loop is activated, planning for the jumping trajectory to be executed in the landing phase. After the TD detection (dashed branches), the last computed trajectories for the CoM (coming either from the LC or from the ELC, according to the LC-feasibility of the CP) and trunk orientation are tracked. Sensing of proprioceptive measures and actuation of control effort are always active (brown branches).

## 5.4 Validation

This section presents the implementation of the ELC framework. Even if we conducted only a preliminary analysis, the promising simulation results reported in the following serve to validate our line of research.

### 5.4.1 Implementation Details

As in the case of Chapter 4, the robot we use to demonstrate the validity of the method is the torque-controlled quadruped Unitree Go1 Edu [Unitree, 2021], and we employ Locosim to visualize, simulate, and interact with the robot (see Appendix A). The LC and the ELC are implemented in the same Python ROS Node running at a frequency of 500 Hz. The CSP for finding the Bézier coefficients is solved by defining a Nonlinear Program (NLP) having (5.22)-

(5.27) as constraints and a constant function as cost to be maximized. To increase computational efficiency, optimizing each ELC loop uses the solution of the previous NLP problem as a warm start. The first CSP is initialized using only the time of convergence of the MSD system in (4.3). The solver used is Ipopt [Wächter and Biegler, 2006], employing MA57 as linear solver [Duff, 2004]. We constructed a library of pre-computed improved feasible regions considering the CoM heights between $c^z_{lb}$ and $c^z_{ub}$ with fixed trunk rotation ${}_{\mathcal{B}}\boldsymbol{R}_{\mathcal{T}} = \boldsymbol{I}_{3\times3}$, as in Figure 5.3. Moreover, we set the margin $\zeta = 0.05$ m.

### 5.4.2 Capture Point Included in the Feet Polygon

As a first assessment, we aim to disregard the virtual constraint that mandates the coincidence of the CP with the feet centroid to achieve the landing task. To test this, we subject the robot to a free fall from a height of 0.5 m with an initial horizontal CoM velocity of 3.0 m/s in a purely forward direction. This initial condition lies beyond the limits delineated in Figure 4.6. Upon employing the LC algorithm, it identifies the CP location $\boldsymbol{u}_o$, which, while not LC-feasible, falls within the coverage of the robot's SP, see Figure 5.6 Consequently, if the CoP is positioned at this point, the robot can effectively decelerate to a stop. Tracking the VHSIP reference with the pWBC will indirectly result in GRFs that correspond to the CoP in the requested location. Being the CP forward from the feet centroid $\boldsymbol{\gamma}$ of a quantity $u^x_o - \gamma^x = 0.063$ m, the CoP can align

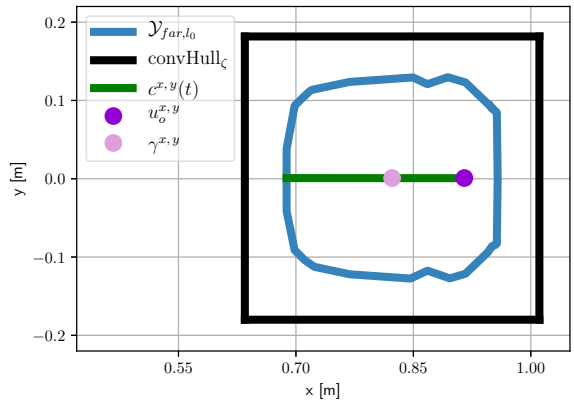



**Figure 5.6:** Simulation: CP included in the feet polygon. Since it is not LC-feasible, the CP location cannot overlap the feet centroid. Nevertheless, it is confined in the SP during the landing phase, and thus the CoM trajectory can come to a stable stop.

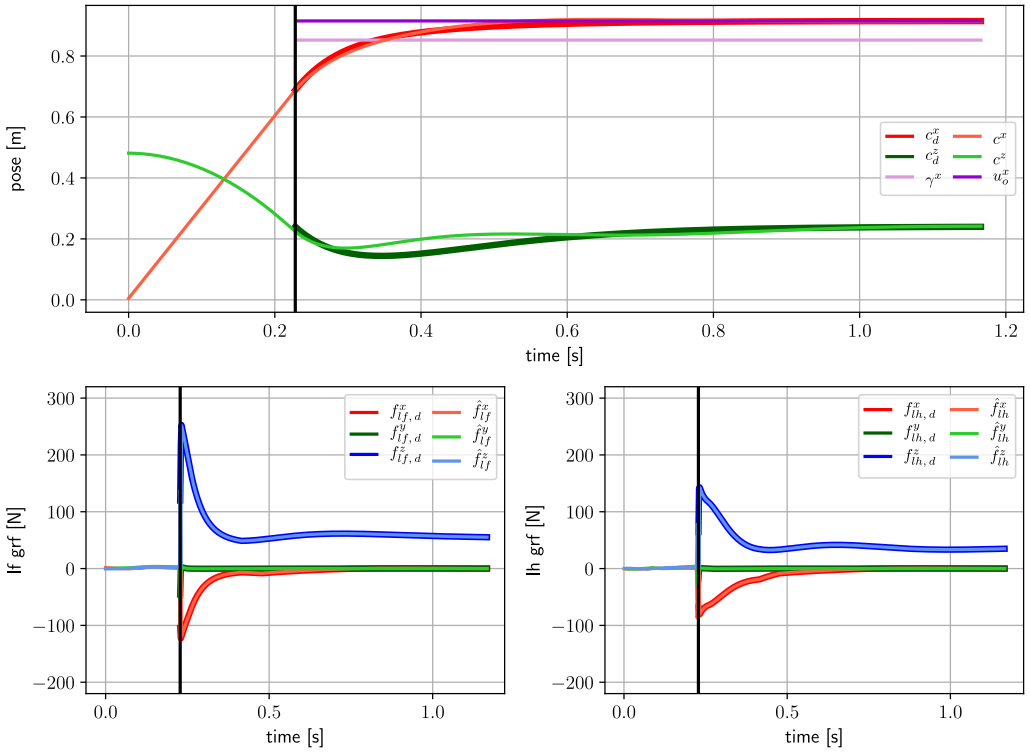



**Figure 5.7:** Simulation: CP included in the feet polygon. Top: desired and actual CoM trajectories expressed in the world frame. Left bottom: desired and estimated GRFs exerted on the left front foot. Right bottom: desired and estimated GRFs exerted on the left hind foot. Reference for pose and GRFs are defined only for the landing phases. The solid black vertical lines denote the detection of the TD. Notice that: 1) the CoM converges towards the CP location even if it does not coincide with the feet centroid and 2) the GRF applied on the front right foot is larger than the one on the left hind foot to the maintain the CoP on the CP, which is closer to the forward edge of the SP.

with the CP by applying greater vertical GRFs with the front feet. As can be observed in Figure 5.7, the resulting CoM trajectory converges above the CP.

### 5.4.3 Two flying phases

The test consists in a drop of the Go1 robot from a height of 0.80 m with an initial horizontal velocity of $\dot{\boldsymbol{c}}_0^{x,y} = \begin{bmatrix} 4.0 & 0.0 \end{bmatrix}^T$ m/s. These conditions make the landing task impossible to be achieved by the LC algorithm. As a matter of fact, the analysis conducted in Section 4.5.2 poses the limit for the horizontal velocity for drops at 0.80 m in the purely forward direction

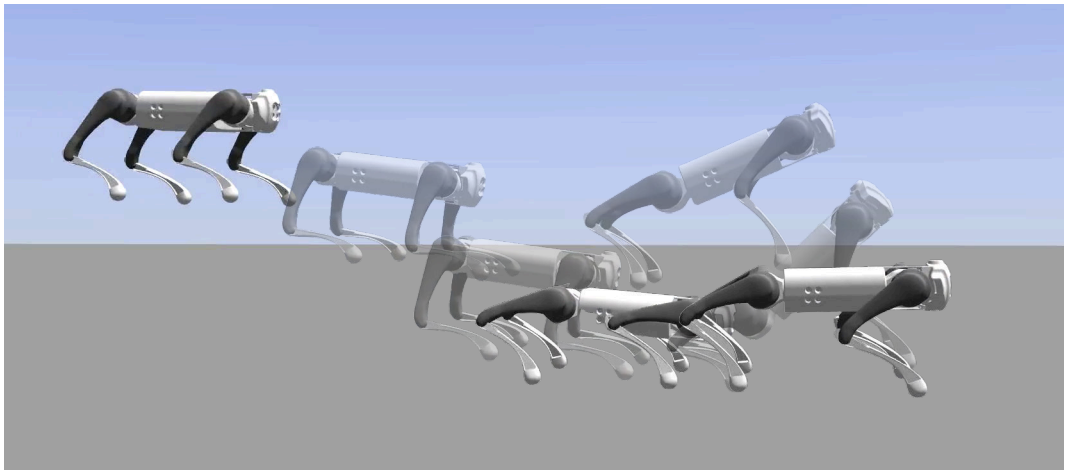


**Figure 5.8:** Simulation: two flying phases. Being the initial horizontal velocity for the CoM larger than the one the LC can stabilize, during the first flying phase, the ELC plans for a reference trajectory that permits a second aerial motion, with reduced horizontal velocity. A combined use of ELC and LC brings the robot to a stable standing state.

to $\begin{bmatrix} 3.0 & 0.0 \end{bmatrix}^T$ m/s. Even relaxing the virtual constraint results in failure. The failure is due to the impossibility of completing the kinematic adjustment required since the CP is not LC-feasible. The ELC is triggered, and it finds a feasible CoM trajectory in a single iteration. Notice that there is a contraction of the horizontal velocity of about 4 times between the beginning (TD) and the conclusion (LO) of the landing phase:

$$\dot{\boldsymbol{c}}_{d,\,TD} = \begin{bmatrix} 4.00 \\ 0.00 \\ -2.97 \end{bmatrix} \text{ m/s}, \qquad \dot{\boldsymbol{c}}_{d,\,LO^+} = \begin{bmatrix} 0.97 \\ 0.00 \\ 2.00 \end{bmatrix} \text{ m/s}.$$

with the LO occurring when $c^z_{LO^+} = 0.285$ m. The latter are satisfactory values since the resulting apex, i.e., the point of inversion of motion, for the second flying phase is at 0.489 m. Taking a look at Figure 4.6, we can see that the red polygon, corresponding to drops from 0.5 m, contains the value of $\dot{\boldsymbol{c}}^{x,\,y}_{d,\,LO^+}$: we have no doubts that the LC will be able to bring the robot to a stable standing state after the second TD. The angular perturbation generated during the second flying phase does not affect the possibility of fulfilling the aerial motion. Comprehensive illustrations about the task are reported in Figure 5.8 and Figure 5.9, respectively, comprising snapshots of the entire motion process, and plots of the robot's pose, GRFs and joint torques.

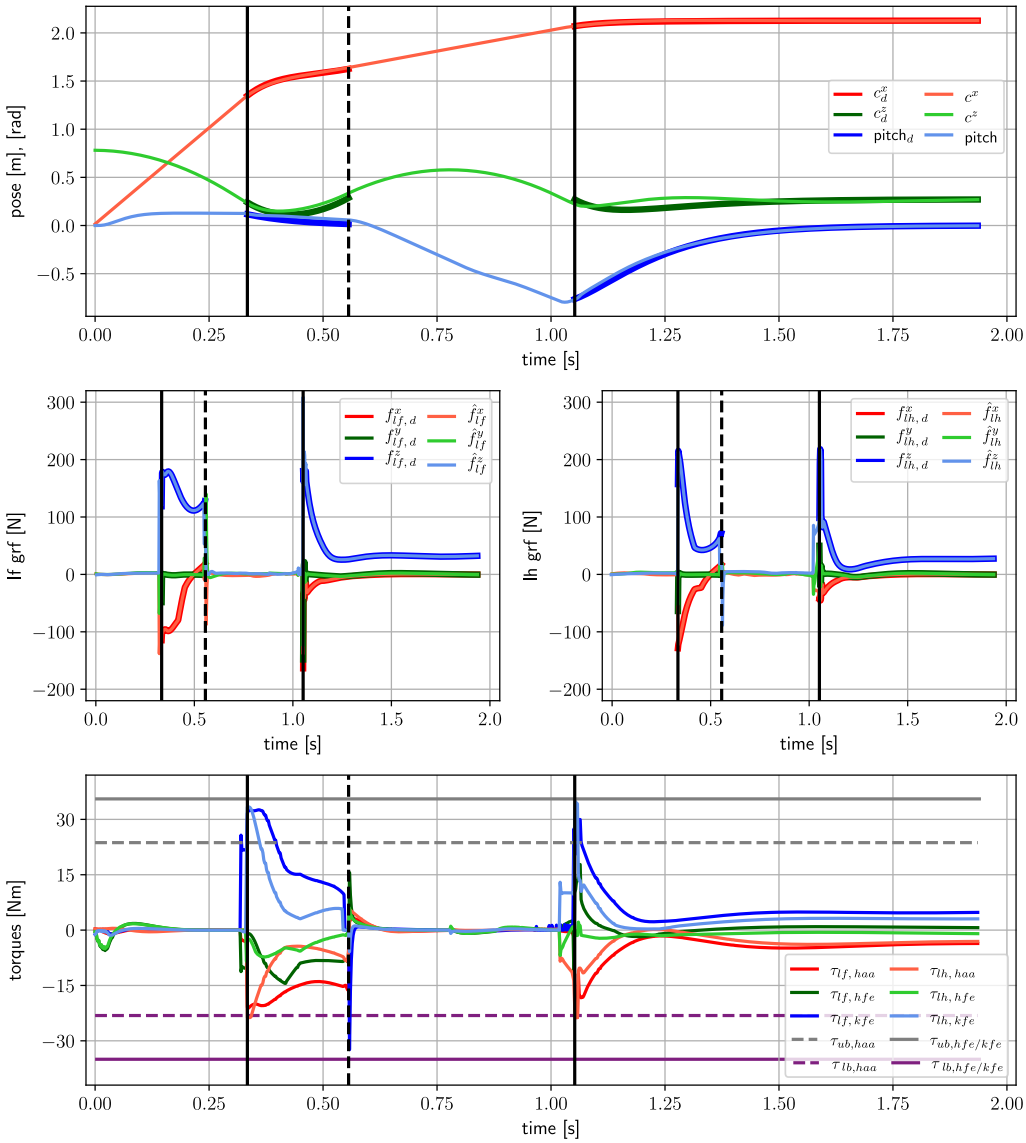



**Figure 5.9:** Simulation: two flying phases. Top: desired and actual CoM and pitch trajectories expressed in the world frame. Left center: desired and estimated GRFs exerted on the left front foot. Right center: desired and estimated GRFs exerted on the left front foot. Bottom: measured torques for the left legs. Reference for pose and GRFs are defined only for the landing phases. Torque bounds are never reached. The solid black vertical lines denote the detection of the two TDs; the dashed black vertical lines denote the LO at the end of the first flying phase.

## 5.5 Summary and Remarks

The present chapter discussed a possible amplification of the capabilities of the LC framework introduced in Chapter 4. We focused our attention on the feasibility of the kinematic adjustment. We recalled powerful tools for describing the set of reachable and feasible CoM positions. The improved feasible region turned out to be a suitable tool to infer the feasibility of the CP. The mere focus on such property was not satisfactory, and thus we proposed a modification of the LC algorithm by introducing the ELC loop. It consists of a CSP that plans the vertical motion reference and the integration of the VHSIP horizontal dynamics. The loop stops when a feasible CoM reference is found, achieving a compression on the horizontal velocity. Being a preliminary analysis, currently, we have no guarantees of convergence of the ELC loop. Nevertheless, the promising simulation results validate our line of research.

The ELC explores additional aerial motions to assist in stabilization. A pertinent query arises when the LC fails to stabilize the robot post the first landing phase. The intriguing aspect is whether, for given initial conditions, there exists a minimum number of landing phases to ensure the robot's safe and stable position. Conversely, if we fix the maximum number of landing phases, what are the initial CoM conditions that the ELC can guide to a stable standing state? In the case of bipedal locomotion, [Zaytsev et al., 2015] claims that any state a model can reach can likely be achieved in at most two steps. Does this result extend to the domain of aerial motions for quadruped robots? Further investigation is needed.

# 6

# Conclusions

*This chapter condensates all the previous considerations, providing a comprehensive overview of aerial motion and the specific achievements realized throughout this thesis. The research topic of aerial motion for quadruped robots has promising applications that extend beyond its immediate context. The insights and solutions derived from our work hold the potential to transcend into various domains, including overcoming insurmountable obstacles, space exploration, entertainment, and more. The chapter concludes by shedding light on the persistent challenges that continue to demand solutions and elucidates potential avenues for extending the impact of our findings.*

## 6.1 Discussion

Aerial motions for quadruped robots present a myriad of intricate research challenges that span various domains: mathematical formulation, mechanical aspects, control design, and computation resource availability. In this dissertation, and in particular in Chapters 3, 4 and 5, I attempted to find suitable solutions to the problem of safe landing after unexpected falls or planned drops. There are a few points on the described methods that need careful discussion. We decided to present all of them here with the aim of providing the reader with a complete picture of the dissertation.

1. Any abrupt alteration in the flywheels' spin impacts the overall angular dynamics of the whole robot. Therefore, flywheels act as reservoirs for accumulating rotational energy, preserving it for subsequent utilization when the need arises. This functionality enables efficient management and controlled deployment of rotational energy in various cases, contributing significantly to the versatility and adaptability of systems incorporating flywheel mechanisms. The specific energy of a flywheel relies on its rotational inertia, and therefore on the geometry and the material density. Both these factors exert an influence on the robot's payload capacity. Flywheels designed for executing extensive rotations within a limited timeframe require larger physical dimensions and/or increased mass, directly influencing the overall load-bearing capacity of the robot. This correlation between flywheel size, rotational capabilities, and mass directly affects the robot's maneuverability and agility, dictating trade-offs between performance and overall weight distribution.
2. The predominant focus within the legged robotics community revolves around devising a cohesive planning and control architecture capable of executing a large spectrum of movements, like walking, running, and jumping among others. This trend emphasizes employing whole-body models as opposed to simplified versions. In contrast, Chapters 4 and 5 opt for specialized landing frameworks utilizing reduced order models, deviating from the conventional use of comprehensive unified frameworks. This divergence should not be surprising, and it offers some implications. Both the LC and the ELC serve as reflex strategies, swiftly activated upon detecting a free fall, effectively preventing catastrophic crashes, preserving the hardware from potential damage, and stabilizing the robot in a standing position. Moreover, the choice prioritizes

computational efficiency over comprehensive dynamical description. Although the impedance model (VHSIP) lacks a detailed depiction of the robot's angular dynamics, the entire framework adeptly accomplishes the landing task for a wide range of angular perturbations.

3. The landing framework detailed in Chapter 4 relies on a limited set of measurements, namely joint position, velocity, torque (estimated from the current), and the accelerometer data from the IMU. By defining the terrain frame $\mathcal{T}$, this method circumvents the need for estimating the linear and angular position and velocity of the robot base concerning an external inertial frame during the flying phase. This characteristic is particularly advantageous as it renders the LC inherently suitable for real-world applications, especially in scenarios where access to motion capture systems might be limited or unavailable. It's crucial to note that this is achievable because the CoM dynamics adhere to a holonomic constraint (the ballistic trajectory) during the flight phase, allowing its evolution to be completely described by the elapsed time from the fall and the initial conditions. Moreover, during the landing phase, the robot is requested not to move its feet, and thus its base pose and velocity can be retrieved using leg odometry, again relying only on a few proprioceptive measurements.
4. The TD detection approach used in both Chapters 4 and 5 aligns with the concepts mentioned in the previous point. Leveraging measurements from the encoders, we devised an estimation method for the GRFs, offering a dependable means to ascertain the contact status of individual feet. Our deliberate choice to avoid mechanical switches or contact sensors stems from the concern that the substantial forces generated during impact could potentially harm these costly and delicate devices. Instead, relying on encoder data serves as a more robust and cost-effective solution, ensuring the reliability of the TD detection process without compromising the safety or integrity of the system.
5. The benefits of continuously recalculating the optimal CoP while executing the kinematic adjustments from the fall detection to the TD detection are at least twofold:
    - The kinematic adjustment aims to keep the feet on the same plane, parallel to the landing surface. Let us assume that, for any reason, such a requirement is not fulfilled and the feet are not aligned. Of course, if the feet were *far* from their references, the proposed LC

would fail, with the robot irremediably falling. So, let us consider only the cases in which the mismatch is minor. For our algorithm, the TD condition is triggered only when *all* the feet are in contact with the ground, that is when the normal component of the estimated ground reaction force on all feet is larger than a given threshold. For the above-mentioned case, the algorithm would detect the TD a few milliseconds after the first contact is made, delaying a little the initiation of tracking the template model dynamics. Given that our test in Section 4.5.3 shows that the method is robust to noise on the initial CoM velocity, the joint velocities, and the actuation torques, this suggests that the method is insensitive to slightly asynchronous foot TD events.

6. It is important to keep in mind that when dealing with unexpected falls or with jumps on surfaces with relevant height variation, the landing task may not be feasible at all. Designing a landing strategy does not mean developing a framework able to drive a quadruped robot to a stable state for *any* initial condition. The robotic research community has to focus on practical cases, e.g., what happens to the 0.25 m tall Go1 robot if it runs at its maximal trotting speed (3.7 m/s) and, without stopping its motion, it is requested to jump off 0.5, 0.8, 1.0 m high boxes? The primary objective behind the development outlined in Chapter 5 was to optimize the robot's capabilities effectively, considering its friction, actuation, and kinematic limitations.
7. Relaxing the virtual constraint that mandates the CoP to be at the feet centroid post-TD compromises robustness. Essentially, this constraint provides the most margin for accommodating tracking uncertainties in the pWBC. Conversely, allowing the CoP to not coincide with the feet centroid demands higher accuracy from the tracking controller.

## 6.2 Summary

With about 50 years of research, legged robots became efficient and robust machines for navigating challenging and harsh environments. There have been meaningful contributions to locomotion strategies, encompassing improvements in both stability and agility. Nevertheless, the field of aerial motions remains a prominent domain of research. Integrating controlled aerial motions into the repertoire of tasks for quadruped robots presents complex

challenges that require further exploration and innovation. This thesis represents a significant advancement to the field. Specifically focusing on aerial reorientation and landing strategies, it marks substantial progress in the quest for enhanced stability, agility, and safety in four-legged robot locomotion.

The OCS, designed with two flywheels, stands out for its ability to control the trunk orientation during the flying phase of the aerial motion. Since the conservation of the angular momentum is a nonholonomic constraint, it is possible to control the robot's orientation by adjusting the position of the bodies composing it. However, typical quadruped robots are designed to have lightweight legs, resulting in a small effect on the total inertia. Single flywheel, heavy boot, tail, and gyroscope have been explored in the literature for enhancing controllability, but all of them have specific drawbacks. Through comprehensive simulations, we demonstrated the effectiveness of our OCS in rejecting disturbances during flight, stabilizing the platform after TD, and executing rapid reorientation maneuvers, exemplified by a backflip, even in reduced gravity environments. Beyond our accomplishments, the versatility of the OCS opens the door to diverse applications, including efficient posture adjustment for quadruped robots navigating uneven terrains.

The second major contribution addresses the critical aspect of safe landings for quadruped robots, particularly after unexpected falls or planned drops. The model-based landing approach relies on a template, i.e., a model defined in lower dimensional spaces compared to the full-dynamics space. Tracking the VHSIP model turned out to be a suitable candidate to avoid multiple bounces, undesired collisions and feet slippage after TD, while reaching a stable standing state. The strategy, reactive at 500 Hz and relying solely on proprioceptive measurements, demonstrated superior performance compared to a naive landing strategy in extensive simulations with significant horizontal velocities and noisy measurements. Real-world experimentation on the robot Go1 validated the framework's practical viability, showcasing its adaptability to different drop scenarios and emphasizing its superiority over other strategies.

The primary objective behind the development outlined in Chapter 5 was to optimize the robot's capabilities effectively. Initially, we relaxed the virtual constraint that mandates the optimal CP alignment with the feet centroid. This modification demonstrated that the framework can stabilize the robot for higher velocities than those that limit the LC performance. Furthermore, we drew inspiration from gymnasts for designing the ELC. It decreases horizontal velocity post-TD and permits the execution of a second flying phase. This

dual-jump motion strategy appears promising in identifying a comprehensive set of initial conditions that can be stabilized for drops.

## 6.3 Future Works

In considering future endeavors, it is essential to emphasize the versatility of OCS, LC, and its improved version ELC, highlighting their potential applicability beyond quadruped robots.

The methodology presented in Chapter 3 transcends the constraints of a specific platform, offering adaptability for reorienting diverse mechanical structures. As a practical example, construction workers can be equipped with backpacks housing two flywheels with incident rotation axes. In the event of a fall from scaffolding, this setup could effectively reorient the human body for touching the ground with an upright posture, employing the same controller proposed in this thesis. At the same time, retrorockets can reduce the fall velocity, yielding a safe and stable landing. To enhance the proposed control strategy, future work includes performing experiments with real platforms. The refinement of the strategy for controlling the robot's angular velocity may benefit from advanced techniques like Nonlinear Model Predictive Control (NMPC), incorporating preview control to consider future samples of the orientation reference. Strategies based on Brocket's theorem could play a fundamental role in stabilizing the yaw to a desired value, fully exploiting the nonholonomic constraint of angular momentum conservation through preliminary roll and pitch maneuvers.

Likewise, the framework introduced in Chapter 4 seamlessly extends to other legged robots, such as bipeds, paving the way for broader applications. Future research directions aim to enhance model descriptiveness by relaxing the assumption of negligible angular momentum variation, thereby expanding the feasible range of horizontal velocities and tolerable angular perturbations.

Chapter Chapter 5 has expanded the landing framework's capabilities by eliminating the virtual constraint of the CP on the feet centroid and considering including kinematic, frictional, and actuation limitations. Additionally, it introduces the potential for a secondary flying phase to achieve a stable standing state. However, the ELC leaves some lines of research open. Firstly, it is crucial to establish the boundaries of its effectiveness, considering parameters such as initial horizontal velocity and drop height. Determining the specific

conditions that result in failure is imperative for understanding and utilization. Furthermore, in cases where the robot fails to stabilize after the second landing phase, the ELC explores additional aerial motions that might aid stabilization. Can specific initial conditions be linked to the minimum number of landing phases required to ensure the robot's safe and stable positioning? This approach is in its initial phase and demands thorough investigation and empirical testing to establish its reliability and efficacy in practical applications.

The integration of the OCS with landing frameworks can provide comprehensive control over aerial orientation and posture adjustment. This necessitates in-depth analyses and experimentation to unveil the full potential of the combined approach. Moreover, the envisioned trajectory extends towards accommodating landings on non-horizontal surfaces and soft terrains, rendering legged robots adaptable to diverse real-world conditions. The proposed backup plan and its extension contribute to a holistic approach to ensuring the robustness and versatility of legged robots during aerial maneuvers.

In conclusion, the future trajectory encompasses refining individual methods, exploring their integration, and extending their applicability to diverse robotic platforms and challenging terrains. Continuous analysis, experimentation, and innovation will propel the field of aerial robotics toward new frontiers, unlocking capabilities and addressing real-world challenges.

# A

# Locosim: an Open-Source Cross-Platform Robotics Framework

*The architecture of a robotics software framework influences the effort and time it takes for end users to test new concepts in a simulation environment and to control real hardware. Many years of activity in the field allowed us to sort out crucial requirements for a framework tailored for robotics: modularity and extensibility, source code reusability, feature richness, and user-friendliness. We implemented these requirements and collected best practices in Locosim, a cross-platform framework for simulation and real hardware. In this appendix, we discover the architecture and the benefits of Locosim, together with some use cases that highlight its potential.*

*Locosim has been designed and implemented together with Dr. Michele Focchi.*

Published. Conference paper: Michele Focchi*, Francesco Roscia*, and Claudio Semini, "Locosim: an Open-Source Cross-Platform Robotics Framework," In *Synergetic Cooperation between Robots and Humans. Proceedings of the CLAWAR 2023 Conference* , Springer Nature Switzerland, 2024, https://doi.org/10.1007/978-3-031-47272-5_33.


* Equal contribution.

Writing software for robotic platforms can be arduous, time-consuming, and error-prone. In recent years, the number of research groups working in robotics has grown exponentially, each group having platforms with peculiar characteristics. The choice of kinematic, actuation systems, and sensing technology is virtually unlimited, and code reuse is fundamental to getting new robots up and running in the shortest possible time. In addition, it is pervasive for researchers willing to test new ideas in simulation without wasting time in coding, for instance, using high-level languages for rapid code prototyping. To pursue these goals, in the past years, several robotics frameworks have been designed for teaching or for controlling specific platforms, e.g., OpenRAVE [Diankov, 2010], Drake [Tedrake and the Drake Development Team, 2019] and SL [Schaal, 2009].

To avoid roboticists *reinventing the wheel* whenever they buy or build a new robot, we present our framework Locosim[1]. Locosim is designed with the primary goal of being platform-independent, dramatically simplifying the task of interfacing a robot with planners and controllers. Locosim consists of a ROS control node [Quigley et al., 2009] (the *low-level* controller), written in C++, that interfaces a custom Python ROS node (the *high-level* planner/controller) with either a Gazebo simulator [Koenig and Howard, 2004] or the real hardware. The planner relies on Pinocchio [Carpentier et al., 2019] for computing the robot's kinematics and dynamics and closes the control loop at a user-defined frequency. The benefits of the proposed framework are multiple.

- Locosim is platform-independent. It supports a list of robots with different morphology (e.g., quadrupeds, arms, hybrid structures, see Figure A.1) and provides features for fast designing and adding new robots.
- Locosim implements functions needed for all robots. Once the robot description is provided, no effort is spent on libraries for kinematics, dynamics, logging, plotting, or visualization. These valuable tools ease the synthesis of a new planner/controller.
- Locosim is easy to learn. The end user invests little time in training and gets an open-source framework with all the benefits of Python and ROS.
- Locosim is modular. Because it heavily uses the inheritance paradigm, classes of increasing complexity can provide different features depending on the nature of the specific robotic application. For instance, the controller for fixed-base robotic arms with a grasping tool can be reused for

[1]Locosim can be downloaded from `www.github.com/mfocchi/locosim`.

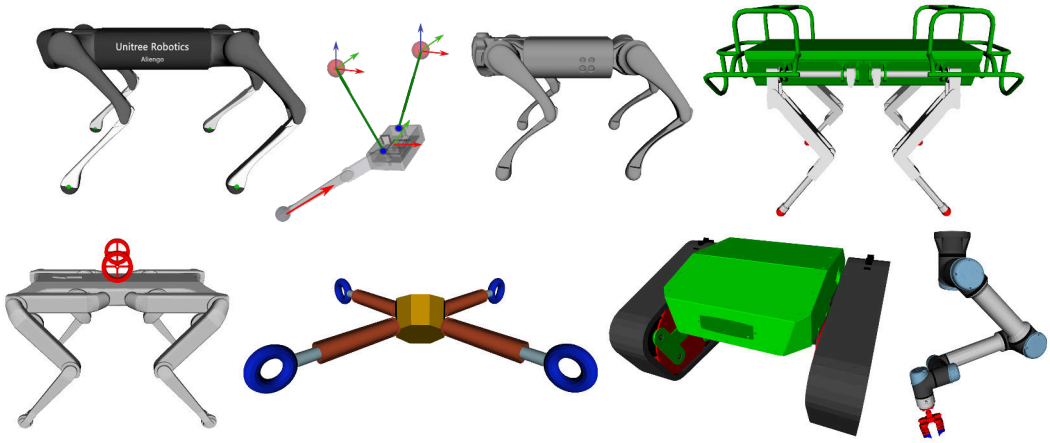



**Figure A.1:** Examples of robots already included in Locosim (from left to right, top to bottom): Aliengo [Unitree, 2019], CLIO [Focchi et al., 2023], Go1 [Unitree, 2021], HyQ [Semini et al., 2011], Solo with Flywheels [Roscia et al., 2023a], Starbot, Tractor Maxxi, UR5 [Robot, 2008] with Gripper (images are not in scale).

a four-legged robot with flywheels since it is built upon the same base class.

- Locosim is extensible. Our framework is modifiable, and the end-user can add any supplementary functionality.
- Locosim is easy to install. It can be used either inside a virtual machine, or a docker container, or natively by manually installing dependencies.

The remainder of this appendix is organized as follows. In Appendix A.1, we highlight the critical requirements of a cross-platform robotics framework. In Appendix A.2, we detail the structure and features of Locosim, providing standardization for robot descriptions and hardware interfaces. In Appendix A.3, we discuss use-case examples of our framework, either with the real robot or with its simulated digital twin. Eventually, we condense the results and present future works in Appendix A.4.

## A.1 Key aspects of a robotics framework

In the most general sense, a *robotics framework* is a software architecture of programs and data that adhere to well-defined rules for operating robots. *End-users* are people who will ultimately use the robotics framework and potentially bring some modifications. The robotics framework is the center of a triangle,

having at its vertices the end-user, the robotic platform, and the simulation environment (see Figure A.2). The simulation must replicate the behavior of the robot with a sufficient degree of accuracy. The end-user must be able to test new features and ideas in the simulation environment before running them on the robot platform. A robotics framework should provide the link among these three. In this context, we identify a list of crucial requirements that a robotics framework should incorporate: generality, modularity, reusability, extensibility, rapid prototyping vs. performance, feature-rich, and end-users development.

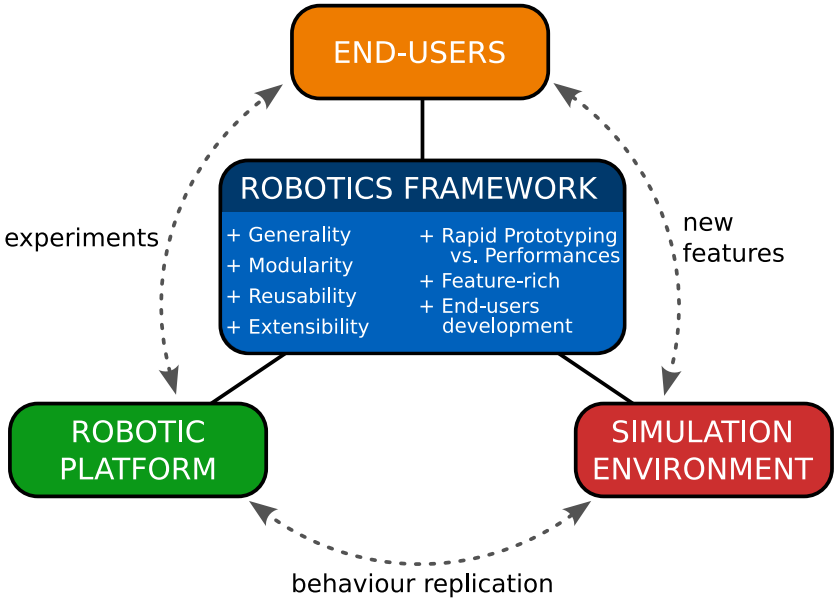



**Figure A.2:** End-users, robotic platform, and simulation environment make a triad only if an effective robotics framework can join them.

### Generality

It is essential to release software free from unnatural restrictions and limitations. An end-user may require to design software for controlling a single robot, for multi-robot cooperation, for swarm coordination, or for reinforcement learning, which requires an abundant number of robots in the training phase. It should be possible to model any kinematic structure (floating base/fixed base robot, kinematic loops, etc.).

### Modularity

A robotics framework should provide separate building blocks of functionalities according to the specific needs of the robot or the application. These building blocks must be replaceable and separated by clear interfaces. The end-user may want to visualize only a specific robot in a particular joint configuration, move the joints interactively, or plan motions and understand the effects of external forces. Each of these functions must be equipped with tools for debugging and testing, e.g., unit tests. Replacing each module with improved or updated versions with additional functionalities should be easy.

### Reusability

Pieces of a program's source code should be reused by reassembling them in more complex programs to provide a desired set of functionalities with minor or no modifications. From this perspective, parametrization is a simple but effective method, e.g., the end-user should be able to use the same framework with different robots, changing only a single parameter. In the same way, *digital twins* [Ramasubramanian et al., 2022] can be realized by varying the status of a flag that selects between the real hardware and the simulation environment. This avoids writing different codes for the simulator and the real robot.

### Extensibility

A robotics framework must be designed with the foresight to support the addition and evolution of hardware and software that may not exist at implementation time. This property can be achieved by providing a general set of application programming interfaces (APIs). Concepts typical of Object-Oriented Programming, such as inheritance and polymorphism, play a crucial role in extensibility.

### Rapid Prototyping vs. Performance

A framework should allow for fast code prototyping of researchers' ideas. More specialized controllers/planners are built from simpler ones in the form of recursive inheritance (*matryoshka principle*). In this way, end-users have unit tests at different levels of complexity. With fast code prototyping, end-users can quickly write software without syntax and logic errors. However, they do

not have any assurance about the performance: such code is just good enough for the case of study in a simulation environment. Stringent requirements appear when executing codes on real robots, e.g., short computation time and limited memory usage. Thus, the framework must expose functionalities that can deal with performance.

#### Feature-rich

Most of the end-users need a sequence of functionalities when working with robots. These include but are not limited to the computation of kinematics and dynamics, logging, plotting, and visualization. A robotics framework should provide them, and they must be easily accessible.

#### End-users Development

Besides its implementation details, a robotics framework should provide methods, techniques, and tools that allow end-users to create, modify, or extend the software [Lieberman et al., 2006] in an easy way. It should run on widely used Operating Systems (OSs) and employ renowned high-level programming languages that facilitate software integration. Clear documentation for installation and usage must be provided, and modules should have self-explanatory names and attributes.

## A.2 Locosim Description

Locosim was born as a didactic framework to simulate both fixed- and floating-base robots. Quickly, it evolved to be a framework for researchers who want to program newly purchased robots in a short time. Locosim runs on machines with Ubuntu as OS and employs ROS as middleware. Within Locosim, end-users can write robot controllers/planners in Python. Check Table A.1 for the supported versions of Ubuntu, ROS, and Python. Additionally, we provide Locosim inside a Virtual Machine and in a Docker container.

**Table A.1:** Locosim supports the following versions of OS, ROS, and Python.

| OS | ROS Version | Python Version |
|---|---|---|
| Ubuntu 16 (deprecated) | Kinetic | 2.7 |
| Ubuntu 18 | Melodic | 3.5 |
| Ubuntu 20 | Noetic | 3.7 |

### A.2.1 Architecture

Locosim consists of four components: Robot Descriptions, Robot Hardware Interfaces, ROS Impedance Controller, and Robot Control. We illustrate each component in the following[1].

#### Robot Descriptions

The Robot Descriptions component contains dedicated packages for the characterization of each robot. Each robot description has a specific folder structure to be respected. For instance, the package containing files to describe the robot `myrobot`, a generic mobile robot platform, must be named `myrobot_description`. With the main focus on fast prototyping and human readability, the robot description is written in Xacro (XML-based scripting language) and avoids code replication through macros, conditional statements, and parameters. It can import descriptions of some parts of the robot from `urdfs` sub-folder or meshes describing the geometry of rigid bodies from the `meshes` sub-folder. At run time, the Xacro file is converted into URDF, allowing the end-user to change some parameters. The `gazebo.urdf.xacro` launches `ros_control` package and the `gazebo_ros_p3d` plugin, which publishes the pose of the robot trunk in the topic `/ground_truth` (needed only for floating-base robots). The robot description directory must contain the files `upload.launch` and `rviz.launch`. The former processes the Xacro generating the URDF and loads it into the parameter server; the latter allows to check the kinematics without having to start the simulator by providing the `conf.rviz` file. This is the configuration for the ROS visualizer RViz, which can be different for every robot. Additionally, the file `ros_impedance_controller.yaml` must be provided for each robot: it contains the sequence of joint names, joint PID gains, and the home configuration. The Python script `go0` will be used

[1]Some of the functions in Locosim components, which are pretty established for the robotics community, are named after [Schaal, 2009].

during the simulation startup to drive the robot to the home configuration.

### Robot Hardware Interfaces

This folder contains drivers for the real hardware platforms supported by Locosim. They implement the interface that bridges the communication between the controller and the real robot, abstracting the specificity of each robot and exposing the same interface (e.g., `EffortInterface`). For instance, the UR5 robot provides three possible ROS hardware interfaces through its driver: *Effort*, *Position*, and *Velocity*, hiding the details of the respective underlying low-level controllers.

### ROS Impedance Controller

ROS Impedance Controller is a ROS package written in C++ that implements the *low-level* joint controller in the form of PID plus feedforward effort (force or torque, depending on the nature of the joint). The `/joint_state_publisher` publishes the actual position, velocity, and effort for each robot joint, either computed by the physics engine or coming from the robot's onboard sensors. By default, the loop frequency is set to 1 kHz. This and other parameters can be regulated in the launch file called `ros_impedance_controller.launch`. For planners and controllers requiring high computational power, the simulation speed can be slowed down at will, and the entire framework will be synchronized accordingly. Robots with specific needs can be dealt with by specifying a custom launch file. This is the case of the CLIO climbing robot that requires the model of the mountain to which it is attached. The `robot_name` parameter is used to load the correct robot description. If the `real_robot` flag is set to true, the robot hardware interface is initialized; otherwise, the Gazebo simulation starts running first the `go0` script from the robot description. In any case, Rviz will be opened with the robot's specific configuration file `conf.rviz` inside the correspondent `robot_description` folder. The end-user can manually change the location where the robot is spawned in the simulator with the `spawn` parameters expressed in the world frame. Physics parameters for the simulator are stored in the sub-folder `worlds`.

### Robot Control

From the end-user perspective, this is the most crucial component. It embraces classes and methods for computation of the robot's kinematics and dy-

namics, logging and plotting time series variables, and real-time visualization on Rviz. Within this component, *high-level* planning/control strategies are implemented. The codes of the component are entirely written in Python and have a few dependencies: above all, NumPy [Harris et al., 2020] and Pinocchio [Carpentier et al., 2019]. The former offers tools for manipulating multidimensional arrays; the latter implements functions for the robot's kinematics and dynamics. Pinocchio can be an essential instrument for researchers because of its efficient computations. Nevertheless, it can be time-consuming and cumbersome to understand for newcomers. To facilitate its usage, we developed a `custom_robot_wrapper` for building a `robot` object, computing robot mass, center of mass, Jacobians, centroidal inertia, and so on with easy-to-understand interfaces.

For the end-user, the starting point for building up a robot planner is the class `BaseControllerFixed`, suitable for fixed-base robots, and the derived class `BaseController`, which handles floating-base ones. In Tables A.2 and A.3, we report a list of the main attributes and methods of `BaseController` and of its parent `BaseControllerFixed`. For having a more complex and specific controller, the end-user can create its own class, inheriting from one of the previous two and adding additional functionalities. For instance, the class `QuadrupedController` inherits from `BaseController`, and it is specific for

**Table A.2:** Main attributes and methods of the `BaseControllerFixed` (BCF) and of the `BaseController` (BC) classes. All vectors (unless specified) are expressed in the world frame.

| | Name | Meaning | Class |
|---|---|---|---|
| Attributes | `q`, `q_des` | actual / desired joint positions | BCF, BC |
| | `qd`, `qd_des` | actual /desired joint velocities | BCF, BC |
| | `tau`, `tau_ffwd` | actual / feed-forward joint torques | BCF, BC |
| | `x_ee` | position of the end-effector expressed in base frame | BCF |
| | `contactForceW` | contact force at the end-effector | BCF |
| | `contactMomentW` | contact moment at the end-effector | BCF |
| | `basePoseW` | base position and orientation in Euler angles | BC |
| | `baseTwistW` | base linear and angular velocity | BC |
| | `b_R_w` | orientation of the base link | BC |
| | `contactsW` | position of the contacts | BC |
| | `grForcesW` | ground reaction forces on contacts | BC |

**Table A.3:** Main methods of the `BaseControllerFixed` (BCF) and of the `BaseController` (BC) classes. For methods with the same name, the derived class loads the parent class's method and adds additional specific elements.

| | Name | Meaning | Class |
|---|---|---|---|
| Methods | `loadModelAnd-Publishers()` | creates the object `robot` (Pinocchio wrapper) and loads publishers for visual features (`ros_pub`), joint commands and declares subscriber to `/ground_truth` and `/joint_states` | BCF, BC |
| | `startFramework()` | executes `ros_impedance_controller.launch` | BCF, BC |
| | `send_des_jstate()` | publishes `/command` topic with setpoints for joint positions, velocities and feed-forward torques | BCF, BC |
| | `startupProcedure()` | initializes PID gains | BCF, BC |
| | `initVars()` | initializes class attributes | BCF, BC |
| | `logData()` | fill in the variables `x_log` with the content of `x` for plotting purposes, it needs to be called at every loop | BCF, BC |
| | `receive_jstate()` | callback associated to the subscriber `/joint_states`, fills in actual joint positions, velocities and torques | BCF, BC |
| | `receive_pose()` | callback associated to the subscriber `/ground_truth`, fills in the actual base pose and twist, and publishes the fixed transform between world and base | BC |
| | `updateKinematics()` | from `q`, `qd`, `basePoseW`, `baseTwistW`, computes position and velocity of the robot's center of mass, the contacts position, the associated Jacobians, the ground reaction forces, the centroidal inertia, the mass matrix and the non-linear effects | BC |

quadruped robots. `Ur5Generic` adds to `BaseControllerFixed` the features of controlling the real robot, a gripper, and a camera attached (or not) to the robotic arm UR5 (see Figure A.3). The controller class is initialized with the string `robot_name`, e.g., we write `BaseController(myrobot)` for the controller of `myrobot`. The end-user must pay particular attention to this string because it creates the link with the robot description and the robot hardware interface if needed. The `robot_name` is used for accessing the dictionary `robot_param` too. Among the other parameters of the dictionary, the flag `real_robot` permits using the same code for both the real (if set to true) and simulated (false) robot, resulting in the digital twin concept. The `BaseControllerFixed` class contains a `ControllerManager` to seamlessly swap between the control modes and controller types if the real hardware supports more than one. For instance, UR5 has two control modes (point, trajectory) and two controller

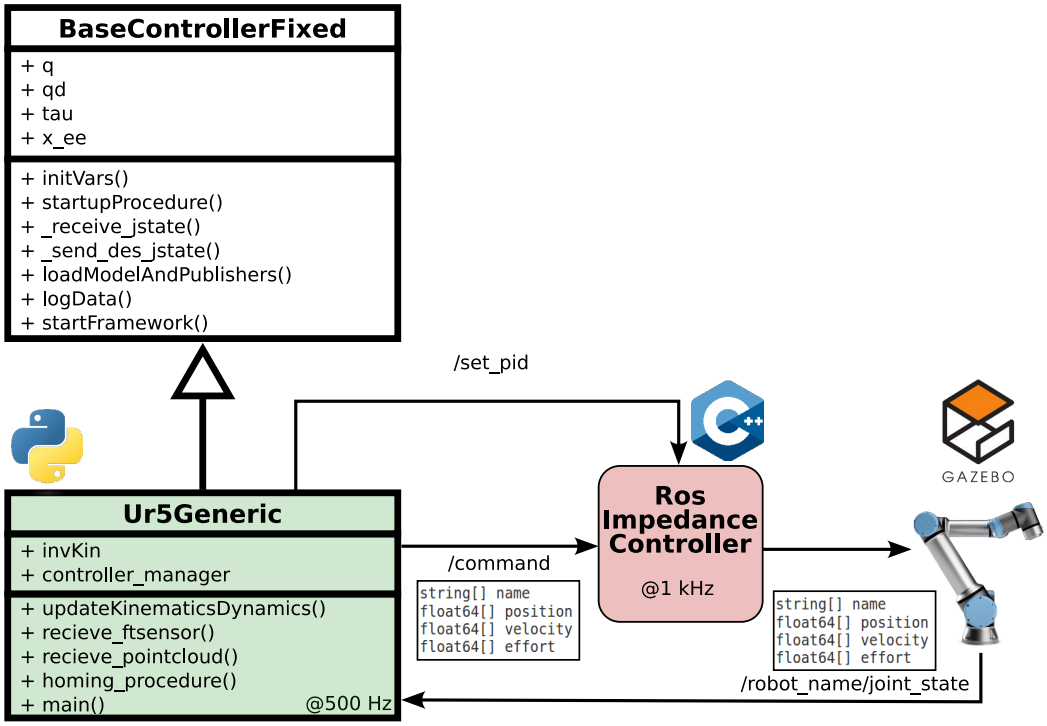



**Figure A.3:** Schematic representation of a typical use-case of Locosim where the end-user wants to simulate the UR5 robot arm. An instance of `Ur5Generic`, which is a derived class of `BaseControllerFixed`, sends the command to the robot though `ros_impedance_controller` and it receives back the actual state. `Ur5Generic` implements features to manage the real robot and the gripper, perform a homing procedure at startup, and a class for inverse kinematics.

types (position, torque), whereas Go1 supports a single low-level torque controller. Additionally, the `GripperManager` class manages the gripper in different ways for the simulation (i.e., adding additional finger joints) and for the real robot (on-off opening/closing service call to the UR5 driver as specified by the manufacturer), hiding this complexity to the end-user. The method `startFramework()` of the `baseControlerFixed` class, while overloaded in the `ur5Generic` class, allow us to decide if to start the simulation or the driver for communicating with the real robot. This function deals with robot and task specificity by taking the list `additional_args` as input to propagate supplementary arguments.

### A.2.2 Design Choices

To fulfill the requirements stated in Appendix A.1, we made a number of choices. We want to focus on why we selected ROS as middleware, Python as (preferred) programming language, and Pinocchio as computational tool for the robot's kinematics and dynamics.

#### Why ROS

The ROS community is spread worldwide. Over the last decade, the community produced hundreds or thousands of open-source tools and software: from device drivers and interface packages of various sensors and actuators to tools for debugging, visualization, and off-the-shelf algorithms for multiple purposes. With the notations of nodes, publishers, and subscribers, ROS quickly solves the arduous problem of resource handling between many processes. ROS is reliable: the system can still work if one node crashes. On the other hand, learning ROS is not an effortless task for newcomers. Moreover, modeling robots with URDF can take lots of time, as well as starting simulations [Joseph and Cacace, 2018]. Locosim relieves end-users from these complications by adopting a *common skeleton* infrastructure for the robot description and for the high-level planner/controller.

#### Why Python

Among the general-purpose programming languages, Python is one of the most used. It is object-oriented and relatively straightforward to learn and use compared to other languages. The availability of open-source libraries and

packages is virtually endless. These reasons make Python perfect for fast prototyping software. Being an interpreted language, Python may suffer in terms of computation time and memory allocation, colliding with the real-time requirements of the real hardware. In these cases, when testing the code in simulation, the end-user may consider using profiling tools to look for the most demanding parts of the code. Before executing the software on the real robot, these parts can be optimized within Python or translated into C++ code, providing Python bindings. For the same performance reasons, the most critical part of the framework, the low-level controller, is directly implemented in C++.

#### Why Pinocchio

Pinocchio [Carpentier et al., 2019] is one of the most efficient tools for computing poly-articulated bodies' kinematics and dynamics. Differently from other libraries in which a meta-program generates some source code for a single specific robot model given in input, as in the case of RobCoGen [Frigerio et al., 2016], Pinocchio is a dynamic library that loads at runtime any robot model. This characteristic makes it suitable for a cross-platform framework. It implements the state-of-the-art Rigid Body Algorithms based on revisited Roy Featherstones algorithms [Featherstone, 2008] and exploits the sparsity of the structure of the dynamics to achieve fast computations. Pinocchio is open-source, primarily written in C++ with Python bindings, and several projects rely on it. Pinocchio also provides derivatives of the kinematics and dynamics, which are essential in gradient-based optimization.

## A.3 Use Cases

We want to emphasize the valuable features of Locosim with practical use cases[2]. First, we report the procedure necessary to visualize the Aliengo robot in RViz. Then, we describe how to implement a controller for the pick and place task for the UR5 manipulator and how to run it either in the simulation environment or with the real robot.

[2]A video showing the above-mentioned and other use cases can be found here:
`https://youtu.be/ZwV1LEqK-LU`

### A.3.1 Visualize a Robot: kinematics check

As the first use case, we illustrate the procedure to visualize the quadruped robot Aliengo (see Figure A.1) in RViz and how to manually interact with its joints. This is a debugging tool which is crucial during the design process of a new robot because it allows the testing of the kinematics without added complexity. In the Robot Description package, we create a folder dedicated to Aliengo named `aliengo_description`. In the `robots` folder, we add the Xacro file for describing the robot's kinematic and dynamic structure. We make use of the flexibility of the Xacro language to simplify the writing process: we include files that describe a leg, the transmission model, and the meshes for each of the bodies. We create the `launch` folder containing the files `upload.launch` and `rviz.launch`. Launching `rviz.launch` from the command line, the end-user can visualize the robot in RViz and manually move the joints by dragging the sliders of the `joint_state_publisher_gui`. The `conf.rviz` file sets a custom configuration of a visual plugin for RViz. With this simple use case, we can effectively understand the importance of the key aspects formalized so far. Being extensible, Locosim allows for the introduction of any new robots, reusing parts of codes already present.

### A.3.2 Simulation and Real Robot

As a second example, we present a pick-and-place task with the anthropomorphic arm UR5 (6 degrees of freedom, see Figure A.1). The pipeline of planning and control starts with the launch file `ros_impedance_controller.launch`. This is common for all the robots: it loads the `robot_description` parameter calling the `upload.launch`, and it starts the Gazebo simulator or the robot driver according to the status of the `real_robot` flag. Additionally, it loads the controller parameters (e.g., PID gains) located in the robot's `robot_description` package. In the simulation case, the launch file spawns the robot at the desired location. In the real robot case, the robot driver is running (in a ROS node) on an onboard computer while the planner runs on an external computer, both sharing the same ROS network. Both in simulation and with real robot, the RViz GUI shows the robot configuration. Another node running on the same computer reads from a fixed frame ZED2 camera and publishes a point cloud message of the environment on a ROS topic. We extract the coordinates of the plastic bricks that are present in the workspace. With `Ur5Generic`, we plan trajectories for the end-effector

position and orientation to grasp and relocate the bricks. We set an inverse kinematic problem to find a joint reference trajectory. It is published in the `/ur5/command` topic, together with feed-forward torques for gravity compensation. The low-level `ros_impedance_controller` provides feedback for tracking the joint references, based on the actual state in `/ur5/joint_state`. On the real robot, there is no torque control, and only position set-points are provided to the `/ur5/joint_group_pos_controller/command` topic as requested by the robot driver provided by the manufacturer. All this is dealt with by the `controller_manager` class, transparent to the end users. The power of Locosim lies in the fact that it is possible to use the same robot control code to simulate the task and execute it on the real robot, as reported in Figure A.4. Notice that using namespaces, e.g., `/ur5/`, allows to manage multiple robots simultaneously.

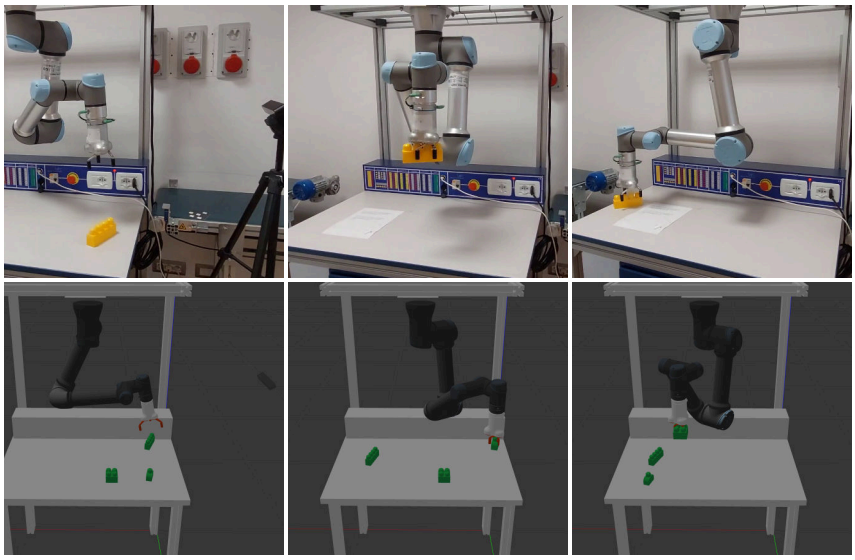


**Figure A.4:** Execution of the autonomous pick-and-place task with the anthropomorphic arm UR5. Top: the end-user can drive the real hardware setting `real_robot` to true. Bottom: the same task can be performed in the simulation environment by setting `real_robot` to false.

## A.4 Conclusions

Locosim is a platform-independent framework for working with robots, either in a simulation environment or with real hardware. Integrating features for

computation of robots' kinematics and dynamics, logging, plotting, and visualization, Locosim is a considerable help for roboticists who need a starting point for rapid code prototyping. If needed, performance can be ensured by implementing the critical parts of the software in C++, providing Python bindings. We proved the usefulness and versatility of our framework with use cases.

Future works include the following objectives: Locosim will be able to handle multiple platforms simultaneously to be used in the fields of swarm robotics and collaborative robotics. We want to provide support for ROS2 since ROS lacks relevant qualifications such as real-time, safety, certification, and security. Additionally, other simulators like Coppelia Sim [Rohmer et al., 2013] and PyBullet [Coumans and Bai, 2016] will be added to Locosim.

# B

# Dynamics of Articulated Motion

Robotic systems we are considering so far are, from a mechanical point of view, a collection of rigid bodies constrained to move together by a set of articulations. Traditionally, the dynamics of such systems, referred to as *Lagrangian dynamics*, are derived via the *Lagrangian formulation*. In this exposition, I present an original derivation through the use of Gauss' principle, according to [Wieber, 2006]. This approach serves to illuminate the intrinsic relationships within the dynamics of articulated body systems. In this section, we embark on a multifaceted exploration by setting forth several key objectives.

- To establish the direct inclusion of the *Newton-Euler equations* for the entire articulated system into its Lagrangian dynamics. This endeavor lays the groundwork for a comprehensive understanding of the system motion.
- To gain insights into the effects of joint torques and gravity on the movements of a free-falling articulated system, highlighting the significant disparity of the conservation of linear and angular momenta.
- To shed light on the role played by contact forces in governing the motion of articulated systems. We establish constraints on them for contact stability and explore them in achieving stable landings.

## B.1 Gauss' Principle of Least Constraints

The articulations among rigid bodies can be implicitly defined by representing the pose of the various components of the system using the generalized coordinates $\boldsymbol{q} \in \mathcal{Q}$. The configuration space $\mathcal{Q}$ is an $n$-dimensional smooth manifold, locally diffeomorphic to an open subset of $\mathbb{R}^n$. Given a trajectory

$\boldsymbol{q}(t) \in \mathcal{Q}$, with $t$ parameter in $\mathbb{R}$, the generalized velocity at a configuration $\boldsymbol{q}$ is the vector $\dot{\boldsymbol{q}}$ belonging to the tangent space $T_q(\mathcal{Q})$ [De Luca and Oriolo, 1995]. Position and orientation of the $k$-th rigid body can be described as the nonlinear *direct kinematic function* of the generalized coordinates:

$$\begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{R}_k \end{bmatrix} = \begin{bmatrix} \mathbf{dkf}_{pk}(\boldsymbol{q}) \\ \mathbf{dkf}_{Rk}(\boldsymbol{q}) \end{bmatrix} = \mathbf{dkf}_k(\boldsymbol{q}) \tag{B.1}$$

Its linear and angular velocity and acceleration are related to the generalized velocities $\dot{\boldsymbol{q}}$ and accelerations $\ddot{\boldsymbol{q}}$ thanks to the translation and rotational Jacobian matrices:

$$\begin{aligned} \dot{\boldsymbol{p}}_k &= \boldsymbol{J}_{pk}(\boldsymbol{q})\,\dot{\boldsymbol{q}}, \\ \boldsymbol{\omega}_k &= \boldsymbol{J}_{Rk}(\boldsymbol{q})\,\dot{\boldsymbol{q}}, \end{aligned} \tag{B.2}$$

and

$$\begin{aligned} \ddot{\boldsymbol{p}}_k &= \boldsymbol{J}_{pk}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{pk}(\boldsymbol{q},\,\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}, \\ \dot{\boldsymbol{\omega}}_k &= \boldsymbol{J}_{Rk}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{Rk}(\boldsymbol{q},\,\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}. \end{aligned} \tag{B.3}$$

*Gauss' principle of least constraint* [Gauss, 1829] is a variational principle for the dynamics of classical mechanical systems. It is a reinterpretation of D'Alembert's principle[1]: the Gauss' principle recasts the research of acceleration at each instant of time into a least square problem [Brogliato, 1996]. It states that the accelerations of a system of rigid bodies subject to some holonomic constraints (see Appendix C) deviates the least possible from the acceleration that it would have had without the constraints [Routh, 1905]. The deviation is measured using the following kinetic metric:

$$\mathcal{D} \triangleq \sum_k \frac{1}{2}\left(\ddot{\boldsymbol{p}}_k - \underline{\ddot{\boldsymbol{p}}}_k\right)^T m_k \left(\ddot{\boldsymbol{p}}_k - \underline{\ddot{\boldsymbol{p}}}_k\right) + \frac{1}{2}\left(\dot{\boldsymbol{\omega}}_k - \underline{\dot{\boldsymbol{\omega}}}_k\right)^T {}_k\boldsymbol{I}_k \left(\dot{\boldsymbol{\omega}}_k - \underline{\dot{\boldsymbol{\omega}}}_k\right), \tag{B.4}$$

where $m_k$ is the mass of the $k$-th body, ${}_k\boldsymbol{I}_k$ is its inertia matrix expressed at its CoM, and $\underline{\ddot{\boldsymbol{p}}}_k$ and $\underline{\dot{\boldsymbol{\omega}}}_k$ are the linear and angular accelerations that it would have had without the constraints, respectively. Such accelerations correspond

[1] D'Alembert's principle is an extension of the principle of virtual work, expanding upon the concept that a system with holonomic constraints reaches an equilibrium state if and only if the overall virtual work carried out by the interaction forces becomes zero. In this context, the virtual displacements remain arbitrary but consistent with the constraints governing the system [Lanczos, 1986].

to the solutions of the Newton-Euler equations

$$m_k \underline{\ddot{\boldsymbol{p}}}_k = \boldsymbol{f}_k, \tag{B.5}$$

$${}_k\boldsymbol{I}_k \, \underline{\dot{\boldsymbol{\omega}}}_k - \left[ {}_k\boldsymbol{I}_k \, \boldsymbol{\omega}_k \right]_\times \boldsymbol{\omega}_k = \boldsymbol{\mu}_k, \tag{B.6}$$

where $\boldsymbol{f}_k \in \mathbb{R}^3$ and $\boldsymbol{\mu}_k \in \mathbb{R}^3$ are the external force and moment acting on the body. Above, $[\,\cdot\,]_\times$ is the skew-symmetric matrix associated with the vector product, i.e.,

$$\begin{aligned} &[\,\cdot\,]_\times : \mathbb{R}^3 \to \mathbb{R}^{3\times 3} \\ &\boldsymbol{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \mapsto [\,\boldsymbol{a}\,]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \end{aligned} \tag{B.7}$$

It should be noted that the Euler equation is expressed with respect to a frame attached to the rigid body, as well as the angular velocity $\boldsymbol{\omega}_k$, which explains the presence of a gyroscopic term $\left[ {}_k\boldsymbol{I}_k \, \boldsymbol{\omega}_k \right]_\times \boldsymbol{\omega}_k$.

Pay attention to the fact that if the system is free of any constraints, then trivially, one retrieves Newton-Euler equations (B.5)–(B.6). Otherwise, solving them for $\underline{\ddot{\boldsymbol{p}}}_k$ and $\underline{\dot{\boldsymbol{\omega}}}_k$, and plugging (B.2)-(B.3) into the definition of the deviation $\mathcal{D}$ (B.4), the optimality conditions for its minimization yields the classical Lagrangian dynamics for a system of articulated rigid bodies:

$$\frac{\partial \mathcal{D}}{\partial \ddot{\boldsymbol{q}}} = \boldsymbol{0} \qquad \Rightarrow \qquad \boldsymbol{M}\left(\boldsymbol{q}\right) \ddot{\boldsymbol{q}} + \boldsymbol{N}\left(\boldsymbol{q},\, \dot{\boldsymbol{q}}\right) \dot{\boldsymbol{q}} = \boldsymbol{\mathcal{F}} \tag{B.8}$$

where $\boldsymbol{M}\left(\boldsymbol{q}\right) \in \mathbb{R}^{n\times n}$ is the mass matrix, $\boldsymbol{N}\left(\boldsymbol{q},\, \dot{\boldsymbol{q}}\right) \dot{\boldsymbol{q}} \in \mathbb{R}^n$ contains the dynamical effects due to Coriolis and centrifugal forces, and $\boldsymbol{\mathcal{F}} \in \mathbb{R}^n$ are the generalized forces acting on the mechanical system. Specifically, their expressions are:

$$\boldsymbol{M}\left(\boldsymbol{q}\right) = \sum_k \boldsymbol{J}_{pk}^T m_k \boldsymbol{J}_{pk} + \boldsymbol{J}_{Rk}^T \, {}_k\boldsymbol{I}_k \, \boldsymbol{J}_{Rk}, \tag{B.9}$$

$$\boldsymbol{N}\left(\boldsymbol{q},\, \dot{\boldsymbol{q}}\right) = \sum_k \boldsymbol{J}_{pk}^T m_k \dot{\boldsymbol{J}}_{pk} + \boldsymbol{J}_{Rk}^T \, {}_k\boldsymbol{I}_k \, \dot{\boldsymbol{J}}_{Rk} - \boldsymbol{J}_{Rk}^T \left[ {}_k\boldsymbol{I}_k \, \boldsymbol{J}_{Rk}\dot{\boldsymbol{q}} \right]_\times \boldsymbol{J}_{Rk}, \tag{B.10}$$

$$\boldsymbol{\mathcal{F}} = \sum_k \boldsymbol{J}_{pk}^T \boldsymbol{f}_k + \boldsymbol{J}_{Rk}^T \boldsymbol{\mu}_k. \tag{B.11}$$

The central point to emphasize is that while the inertia matrix definition (B.9) aligns with the one found in typical robotics textbooks [Murray et al., 1994;

Siciliano et al., 2009; Spong et al., 1984], the same cannot be said for the definition of the nonlinear effects (B.10). Traditionally, these are expounded using the *Christoffel symbols of the first kind*, which obscure the underlying structure that is manifest in the presented formulation. This underlying structure is poised to offer significant utility in the analysis of articulated motion dynamics. For instance, the skew-symmetric property of the matrix $\dot{\boldsymbol{M}} - 2\boldsymbol{N}$ is self-evident.

For the sake of comprehensiveness, I hereby present the conventional formulation for nonlinear effects to underscore how enigmatic it can be. The element in the generic position $(i, j)$ (reads: $i$-th row, $j$-th column) of $\boldsymbol{N}$ is

$$n_{ij} = \sum_h c_{ijh} \dot{q}_h$$

where the coefficients

$$c_{ijh} = \frac{1}{2} \left( \frac{\partial m_{ij}}{\partial q_h} + \frac{\partial m_{ih}}{\partial q_j} - \frac{\partial m_{jh}}{\partial q_i} \right)$$

are the Christoffel symbols.

## B.2 Inner Structure of Floating-base Systems' Dynamics

### B.2.1 Inner Structure in the Generalized Coordinates

In the case of legged systems (or, more generically, free-floating articulated systems), the generalized coordinates combine three different information: the configuration of the joints $\hat{\boldsymbol{q}}$ (belonging to a subset of $\mathbb{R}^{n_j}$) with the global position $\boldsymbol{p}_{\mathcal{B}} \in \mathbb{R}^3$ and orientation $\boldsymbol{\phi}_{\mathcal{B}} \in \mathrm{SO}(3)$ of a reference frame attached to one rigid body of the system, often termed *base*,

$$\boldsymbol{q} = \begin{bmatrix} \hat{\boldsymbol{q}} \\ \boldsymbol{p}_{\mathcal{B}} \\ \boldsymbol{\phi}_{\mathcal{B}} \end{bmatrix}. \tag{B.12}$$

### B.2.2 Inner Structure in the Kinematic Equations

The partition of the generalized coordinates into three categories is a recurring feature in the kinematic and dynamic equations of the articulated structure.Let us consider the $k$-th body. Its position and rotation matrix $\boldsymbol{p}_k$ and $\boldsymbol{R}_k$ with respect to the inertial frame are obtained by compositions with the base position $\boldsymbol{p}_{\mathcal{B}}$ and rotation matrix $\boldsymbol{R}_{\mathcal{B}}(\boldsymbol{\phi}_{\mathcal{B}})$:

$$\begin{aligned} \boldsymbol{p}_k &= \boldsymbol{p}_{\mathcal{B}} + \boldsymbol{R}_{\mathcal{B}}\, {}_{\mathcal{B}}\boldsymbol{p}_k \\ \boldsymbol{R}_k &= \boldsymbol{R}_{\mathcal{B}}\, {}_{\mathcal{B}}\boldsymbol{R}_k \end{aligned} \tag{B.13}$$

with ${}_{\mathcal{B}}\boldsymbol{p}_k$ and ${}_{\mathcal{B}}\boldsymbol{R}_k$ being the position and the rotation matrix of the $k$-th body with respect to the base reference frame. They depend only on the robot kinematics through the direct kinematic function expressed in base frame:

$$\begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{p}_k \\ {}_{\mathcal{B}}\boldsymbol{R}_k \end{bmatrix} = \begin{bmatrix} {}_{\mathcal{B}}\mathbf{dkf}_{pk}(\hat{\boldsymbol{q}}) \\ {}_{\mathcal{B}}\mathbf{dkf}_{Rk}(\hat{\boldsymbol{q}}) \end{bmatrix} = {}_{\mathcal{B}}\mathbf{dkf}_k(\hat{\boldsymbol{q}}) \tag{B.14}$$

A similar result appears in terms of differential kinematics. Let denote with ${}_{\mathcal{B}}\dot{\boldsymbol{p}}_k$ and ${}_{\mathcal{B}}\boldsymbol{\omega}_k$ the linear and angular velocities of the $k$-th body with respect to the base reference frame. To obtain the linear and angular velocities $\dot{\boldsymbol{p}}_k$ and $\boldsymbol{\omega}_k$ in the inertial frame, they must be composed with $\dot{\boldsymbol{p}}_{\mathcal{B}}$ and $\boldsymbol{\omega}_{\mathcal{B}}$ through the following composition rules (remember that the angular velocities are expressed in local frames):

$$\begin{aligned} \dot{\boldsymbol{p}}_k &= \boldsymbol{R}_{\mathcal{B}}(\boldsymbol{\phi}_{\mathcal{B}})\, {}_{\mathcal{B}}\dot{\boldsymbol{p}}_k + \dot{\boldsymbol{p}}_{\mathcal{B}} + [\boldsymbol{R}_{\mathcal{B}}(\boldsymbol{\phi}_{\mathcal{B}})\, \boldsymbol{\omega}_{\mathcal{B}}]_{\times} (\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}), \\ \boldsymbol{R}_k(\boldsymbol{q})\, \boldsymbol{\omega}_k &= \boldsymbol{R}_k(\boldsymbol{q})\, {}_{\mathcal{B}}\boldsymbol{\omega}_k + \boldsymbol{R}_{\mathcal{B}}(\boldsymbol{\phi}_{\mathcal{B}})\, \boldsymbol{\omega}_{\mathcal{B}}, \end{aligned} \tag{B.15}$$

being $\boldsymbol{R}_{\mathcal{B}}(\boldsymbol{\phi}_{\mathcal{B}})$ and $\boldsymbol{R}_k(\boldsymbol{q})$ the rotation matrices of the base frame and of the frame attached to the $k$-th rigid body with respect to the inertial reference frame, respectively. Now we have to introduce two facts. First, the velocities ${}_{\mathcal{B}}\dot{\boldsymbol{p}}_k$ and ${}_{\mathcal{B}}\boldsymbol{\omega}_k$ are linearly dependent on the vector $\dot{\hat{\boldsymbol{q}}}$ as

$$\begin{aligned} {}_{\mathcal{B}}\dot{\boldsymbol{p}}_k &= {}_{\mathcal{B}}\boldsymbol{J}_{pk}(\hat{\boldsymbol{q}})\, \dot{\hat{\boldsymbol{q}}}, \\ {}_{\mathcal{B}}\boldsymbol{\omega}_k &= {}_{\mathcal{B}}\boldsymbol{J}_{Rk}(\hat{\boldsymbol{q}})\, \dot{\hat{\boldsymbol{q}}}. \end{aligned}$$

We refer to ${}_{\mathcal{B}}\boldsymbol{J}_{pk}$ and ${}_{\mathcal{B}}\boldsymbol{J}_{Rk}$ as the translations and rotational Jacobians in base frame. Second, the angular velocity $\boldsymbol{\omega}_{\mathcal{B}}$ of the base frame can be related

to the angular rate $\dot{\theta}_{\mathcal{B}}$ as

$$\boldsymbol{\omega}_{\mathcal{B}} = \boldsymbol{J}_{Rb}\left(\boldsymbol{\phi}_{\mathcal{B}}\right)\dot{\boldsymbol{\phi}}_{\mathcal{B}}.$$

In the special case in which the base orientation is described with the Roll-Pitch-Yaw convention, $\boldsymbol{\phi}_{\mathcal{B}} = \begin{bmatrix} \varphi & \vartheta & \psi \end{bmatrix}^T$ and

$$\boldsymbol{J}_{Rb}\left(\boldsymbol{\phi}_{\mathcal{B}}\right) = \boldsymbol{T}\left(\boldsymbol{\phi}_{\mathcal{B}}\right) = \begin{bmatrix} 0 & -\sin\phi & \cos\varphi\sin\vartheta \\ 0 & \cos\phi & \sin\varphi\cos\vartheta \\ 1 & 0 & 0 \end{bmatrix}. \tag{B.16}$$

With these two facts and dropping the dependencies, (B.15) rewrites as

$$\begin{aligned} \dot{\boldsymbol{p}}_k &= \boldsymbol{R}_{\mathcal{B}}\,{}_{\mathcal{B}}\boldsymbol{J}_{pk}\,\dot{\hat{\boldsymbol{q}}} + \dot{\boldsymbol{p}}_{\mathcal{B}} - \left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_{\times}\boldsymbol{R}_{\mathcal{B}}\,\boldsymbol{J}_{Rb}\dot{\boldsymbol{\phi}}_{\mathcal{B}} \\ \boldsymbol{\omega}_k &= {}_{\mathcal{B}}\boldsymbol{J}_{Rk}\,\dot{\hat{\boldsymbol{q}}} + \boldsymbol{R}_k^T\boldsymbol{R}_{\mathcal{B}}\,\boldsymbol{J}_{Rb}\,\boldsymbol{\phi}_{\mathcal{B}} \end{aligned} \tag{B.17}$$

and we can finally observe that the translation and rotation Jacobians exhibit exactly the same structure of the generalized coordinate vector:

$$\begin{aligned} \boldsymbol{J}_{pk} &= \begin{bmatrix} \boldsymbol{R}_{\mathcal{B}}\,{}_{\mathcal{B}}\boldsymbol{J}_{pk} & \boldsymbol{I}_{3\times3} & -\left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_{\times}\boldsymbol{R}_{\mathcal{B}}\,\boldsymbol{J}_{Rb} \end{bmatrix}, \\ \boldsymbol{J}_{Rk} &= \begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{J}_{Rk} & \boldsymbol{O}_{3\times3} & \boldsymbol{R}_k^T\boldsymbol{R}_{\mathcal{B}}\,\boldsymbol{J}_{Rb}, \end{bmatrix}, \end{aligned} \tag{B.18}$$

where $\boldsymbol{O}_{3\times3}$ and $\boldsymbol{I}_{3\times3}$ represent the $3\times3$ zero and the identity matrix, respectively.

### B.2.3 Inner Structure in the Dynamic Equations

The structure of the generalized coordinates (B.12) is mirrored in the structure of the inertia and nonlinear effects matrices, as well as in the definition of the generalized forces. Indeed, substituting $\boldsymbol{J}_{pk}^T$ and $\boldsymbol{J}_{Rk}^T$ into (B.9)-(B.11) leads to

$$\boldsymbol{M}\left(\boldsymbol{q}\right) = \sum_k \begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{J}_{pk}^T\,\boldsymbol{R}_{\mathcal{B}}^T m_k \boldsymbol{J}_{pk} + {}_{\mathcal{B}}\boldsymbol{J}_{Rk}^T\,{}_k\boldsymbol{I}_k\,\boldsymbol{J}_{Rk} \\ m_k\boldsymbol{J}_{pk} \\ \boldsymbol{J}_{Rb}^T\boldsymbol{R}_{\mathcal{B}}^T\left(\left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_{\times} m_k\boldsymbol{J}_{pk} + \boldsymbol{R}_{k_k}\boldsymbol{I}_k\,\boldsymbol{J}_{Rk}\right) \end{bmatrix}, \tag{B.19}$$

$$\boldsymbol{N}\left(\boldsymbol{q},\,\dot{\boldsymbol{q}}\right) = \sum_k \begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{J}_{pk}^T\,\boldsymbol{R}_{\mathcal{B}}^T m_k \dot{\boldsymbol{J}}_{pk} + {}_{\mathcal{B}}\boldsymbol{J}_{Rk}^T\,{}_k\boldsymbol{I}_k\,\dot{\boldsymbol{J}}_{Rk} - {}_{\mathcal{B}}\boldsymbol{J}_{Rk}^T\left[{}_k\boldsymbol{I}_k\,\boldsymbol{J}_{Rk}\dot{\boldsymbol{q}}\right]_\times \boldsymbol{J}_{Rk} \\ m_k\dot{\boldsymbol{J}}_{pk} \\ \boldsymbol{J}_{Rb}^T\boldsymbol{R}_{\mathcal{B}}^T\left(\left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_\times m_k\dot{\boldsymbol{J}}_{pk} + \boldsymbol{R}_{k\,k}\boldsymbol{I}_k\,\dot{\boldsymbol{J}}_{Rk} - \boldsymbol{R}_k\left[{}_k\boldsymbol{I}_k\,\boldsymbol{J}_{Rk}\dot{\boldsymbol{q}}\right]_\times \boldsymbol{J}_{Rk}\right) \end{bmatrix}, \tag{B.20}$$

$$\boldsymbol{\mathcal{F}} = \sum_k \begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{J}_{pk}^T\,\boldsymbol{R}_{\mathcal{B}}^T\boldsymbol{f}_k + {}_{\mathcal{B}}\boldsymbol{J}_{Rk}^T\,\boldsymbol{\mu}_k \\ \boldsymbol{f}_k \\ \boldsymbol{J}_{Rb}^T\boldsymbol{R}_{\mathcal{B}}^T\left(\left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_\times \boldsymbol{f}_k + \boldsymbol{R}_k\boldsymbol{\mu}_k\right) \end{bmatrix}. \tag{B.21}$$

After employing Gauss' principle so far, we can now observe that the partition into three distinct blocks of the generalized coordinates gives three distinct lines in the Lagrangian dynamics equation (B.8). Let us focus on the last line With simple computation exploiting (B.2)-(B.3), the second line corresponds to the equality between the linear momentum of the articulated system and the sum of all the forces applied to it

$$\sum_k m_k\ddot{\boldsymbol{p}}_k = \sum_k \boldsymbol{f}_k. \tag{B.22}$$

Conversely, the third line rewrites as

$$\boldsymbol{J}_{Rb}^T\boldsymbol{R}_{\mathcal{B}}^T\sum_k\left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_\times m_k\ddot{\boldsymbol{p}}_k + \boldsymbol{R}_{k\,k}\boldsymbol{I}_k\,\dot{\boldsymbol{\omega}}_k - \boldsymbol{R}_k\left[{}_k\boldsymbol{I}_k\,\boldsymbol{\omega}_k\right]_\times\boldsymbol{\omega}_k = \boldsymbol{J}_{Rb}^T\boldsymbol{R}_{\mathcal{B}}^T\sum_k\left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_\times\boldsymbol{f}_k + \boldsymbol{R}_k\boldsymbol{\mu}_k \tag{B.23}$$

and it represents the balance between the angular momentum of the articulated system and the sum of all the torques acting on it. Both the above equations are expressed with respect to $\boldsymbol{p}_{\mathcal{B}}$ in an inertial reference frame. In summary, (B.22) and (B.23) are a Newton and an Euler equation describing the dynamics of the whole system.

### B.2.4 Characterization of the System's Inputs

We are going to examine three distinct categories of generalized forces commonly encountered in systems of articulated bodies: the gravity $\boldsymbol{\mathcal{F}}_g$, the control forces $\boldsymbol{\mathcal{F}}_\tau$, and contact forces $\boldsymbol{\mathcal{F}}_c$.

The gravity force and torque acting on the $k$-th rigid body are

$$\begin{aligned} \boldsymbol{f}_k &= m_k \boldsymbol{g} \\ \boldsymbol{\mu}_k &= \mathbf{0}, \end{aligned} \tag{B.24}$$

denoting with $\boldsymbol{g}$ the vector of the gravity vector field. Plugging the latter into (B.21) leads to

$$\boldsymbol{\mathcal{F}}_g = \sum_k \begin{bmatrix} {}_{\mathcal{B}}\boldsymbol{J}_{pk}^T \boldsymbol{R}_{\mathcal{B}}^T m_k \boldsymbol{g} \\ m_k \boldsymbol{g} \\ \boldsymbol{J}_{Rb}^T \boldsymbol{R}_{\mathcal{B}}^T \left[\boldsymbol{p}_k - \boldsymbol{p}_{\mathcal{B}}\right]_\times m_k \boldsymbol{g} \end{bmatrix} \tag{B.25}$$

Observing the structure of the translation and rotation Jacobians reported in (B.18), we immediately derive that

$$\boldsymbol{J}_{pk} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{g} \\ \mathbf{0} \end{bmatrix} = \boldsymbol{g} \qquad \text{and} \qquad \boldsymbol{J}_{Rk} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{g} \\ \mathbf{0} \end{bmatrix} = \mathbf{0}.$$

In light of these facts, one can relate the gravity force to the inertia matrix in (B.19) as

$$\boldsymbol{\mathcal{F}}_g = \boldsymbol{M}(\boldsymbol{q}) \begin{bmatrix} \mathbf{0} \\ \boldsymbol{g} \\ \mathbf{0} \end{bmatrix}.$$

This leads to the expression of the dynamics (B.8) of the system under the sole action of gravity:

$$\boldsymbol{M}(\boldsymbol{q}) \left( \ddot{\boldsymbol{q}} - \begin{bmatrix} \mathbf{0} \\ \boldsymbol{g} \\ \mathbf{0} \end{bmatrix} \right) + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}} = \mathbf{0}.$$

An obvious observation must be reported here. Even in the case of complex, articulated systems, the gravity force exerts a linear acceleration along the direction of the gravity vector field. Often, the generalized force vector $-\boldsymbol{\mathcal{F}}_g$, which represents the nonlinear effects due to gravity, is denoted by $\boldsymbol{g}(\boldsymbol{q})$ (in this document, we always explicit the dependency on the configuration vector $\boldsymbol{q}$ to avoid confusion between the gravitational torque $\boldsymbol{g}(\boldsymbol{q})$ and the gravity acceleration $\boldsymbol{g}$).

Regarding control forces, we assume that the only forces affecting the system are internal forces that operate between the various components of the

system. These internal forces result from the action of muscles or actuators on the system's articulations. Applying Newton's law of action and reaction, we quickly deduce that their summation takes the following form

$$\boldsymbol{\mathcal{F}}_{\tau} = \begin{bmatrix} \boldsymbol{\tau} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{B.26}$$

Commonly, $\boldsymbol{\tau}$ is referred to as the vector of joint torques. In this work, we identify with $\boldsymbol{S}$ the so-called joint selection matrix, i.e., the linear operator such that $\boldsymbol{S}^T\boldsymbol{\tau} = \boldsymbol{\mathcal{F}}_{\tau}$.

We conclude this overview of the inputs by having a look at the forces that act when the system is in contact with the surrounding environment. The structure of the Lagrangian dynamics (B.8) makes it clear that contact forces play a crucial role on motion. The previous analysis applies to various contact situations: walking, running, or jumping, but also sliding and rolling scenarios such as when skiing or skating. Even if there exist many models for describing contacts [Brogliato, 1996], they always have to satisfy constraints on their direction and amplitude due to unilateral constraints and limited friction. We summarize all these limitations in a vector inequality, relating contact forces to the position of the system as

$$\boldsymbol{\mathcal{A}}\left(\boldsymbol{q},\, \boldsymbol{\mathcal{F}}_c\right) \leq \mathbf{0}. \tag{B.27}$$

It is a frequent practice to express the full dynamics outlined in (B.8) in an alternate manner, emphasizing the dynamic impact of each force input as follows:

$$\boldsymbol{M}\left(\boldsymbol{q}\right)\ddot{\boldsymbol{q}} + \boldsymbol{N}\left(\boldsymbol{q},\, \dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \boldsymbol{g}\left(\boldsymbol{q}\right) = \boldsymbol{S}^T\boldsymbol{\tau} + \sum_{k=1}^{n_c} \boldsymbol{J}_k^T \begin{bmatrix} \boldsymbol{f}_k \\ \boldsymbol{\mu}_k \end{bmatrix}, \tag{B.28}$$

being $n_c$ the number of contacts between the articulated system and the surrounding environment, and having stacked the translation and rotational Jacobian defined in (B.2) into a single matrix as

$$\boldsymbol{J}_k = \begin{bmatrix} \boldsymbol{J}_{pk} \\ \boldsymbol{J}_{Rk} \end{bmatrix}. \tag{B.29}$$

The torque $\boldsymbol{N}\left(\boldsymbol{q},\, \dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \boldsymbol{g}\left(\boldsymbol{q}\right) = \boldsymbol{h}\left(\boldsymbol{q},\, \dot{\boldsymbol{q}}\right)$ is usually named vector of nonlinear

dynamical effects, and it is due to Coriolis, centrifugal and gravity force.

## B.3 Constraints on Contact Forces

In the previous section, we have clarified the critical role contact force $\boldsymbol{\mathcal{F}}_c$ plays in moving the CoM of an articulated system. This force is the resultant of external forces and moments, typically applied to the limbs' appendage, i.e., feet or hands. For conducting our analysis, we rely on the following assumption: the contact area between a rigid body and the environment is considered to have negligible dimension. It is reasonably applicable to most quadruped robots, as they typically feature point feet. Let us focus only on $k$-th body, which is considered to be in touch with the external environment. In the most general context of biomechanics, the external force $\boldsymbol{f}_k$ exerted by the ground is named Ground Reaction Force (GRF). Notice that, because of the above assumption, the external moment is always zero: $\boldsymbol{\mu}_k = \mathbf{0}$. We define the contact frame $\Psi$ with the $XY$-plane on the contact surface and the $Z$-axis orthogonal. For the sake of simplicity in notation, we will use the terms *tangential* and *normal* when describing vectors in relation to this frame. For instance,

$$\boldsymbol{f}_k^t = \begin{bmatrix} {}_\Psi f_k^x & {}_\Psi f_k^y \end{bmatrix}^T \qquad \text{and} \qquad f_k^n = {}_\Psi f_k^z$$

are the components of the external force, respectively, tangential and normal to the contact surface, see Figure B.1.

### B.3.1 Unilateral Constraint

From our argument, we can exclude the specific scenarios involving systems equipped with mechanisms like suction cups, Velcro, or other means for securing the body to the surrounding environment. Therefore, the articulated system is capable of exerting a pushing force on the contact surface, i.e.,

$$f_k^n \geq 0. \tag{B.30}$$

This limitation is the well-known *unilateral constraint*, and the sign comes with the convection that forces exerted by the articulated system are positive. The relationship between the normal components of external force and the body's position can be established using either rigid or compliant models to address impacts and other nonsmooth behaviors [Brogliato, 1996]. In the rigid contact

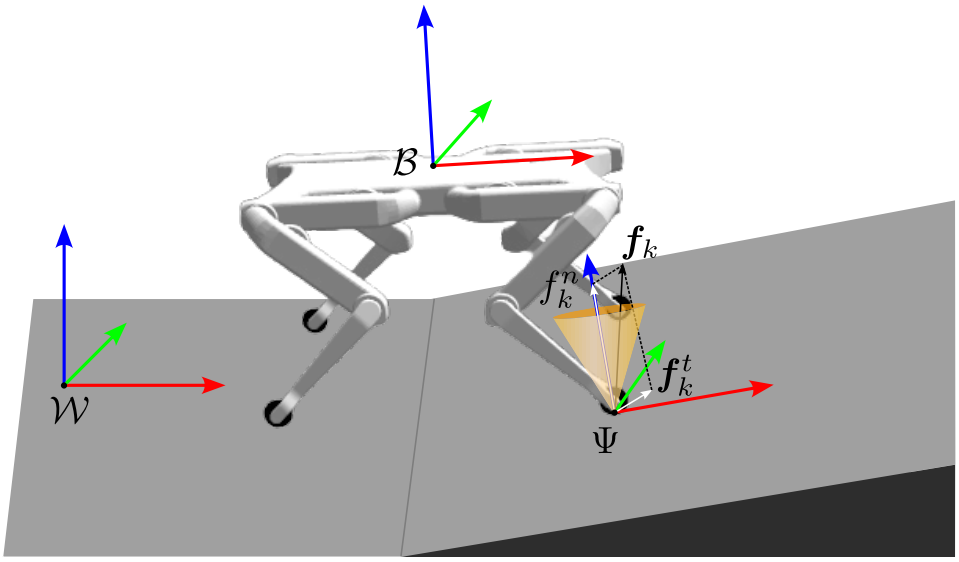



**Figure B.1:** Visual representation of the normal and tangential components of the GRF exerted on a foot of the Solo robot. The red, green, and blue arrows represent the $x$, $y$, $z$ axes of the world frame $\mathcal{W}$, of the base frame $\mathcal{B}$ and of the contact frame $\Psi$. The GRF is depicted together with its friction cone.

model, the normal force is non-negative when there is contact, or it is null when there is no contact:

$$\begin{cases} f_k^n \geq 0 & \text{if} \quad p_k^n = 0, \\ f_k^n = 0 & \text{if} \quad p_k^n > 0. \end{cases} \tag{B.31}$$

The above equation says that both the normal force and position with respect to the contact surface must be non-negative, but at least one must be zero. Therefore, it can be summarized into the complementarity constraint [Pang and Trinkle, 2000]:

$$f_k^n,\, p_k^n \geq 0, \quad f_k^n\, p_k^n = 0.$$

The rigid contact model does not account for any penetration of contact points beneath the contact surfaces. Notice that the constraint can also be formulated using the foot velocity as:

$$f_k^n \geq 0, \quad f_k^n\, \dot{p}_k^n = 0.$$

The above equation means that when the body is moving, the force exerted on it is null, whilst the force is non-negative if the body is in contact. While this thesis does not delve into the intricacies of penetrations, for the sake of comprehensiveness, it is worth noting that several compliant models have been proposed (a survey can be found in [Alves et al., 2015]). These models, collectively, can be summarized as follows:

$$\begin{cases} f_k^n \geq f_k^n\left(p_k^n, \dot{p}_k^n\right) & \text{if} \quad p_k^n \leq 0, \\ f_k^n = 0 & \text{if} \quad p_k^n > 0. \end{cases} \tag{B.32}$$

Compliant contact models offer a more precise representation of the deformation in contact bodies but often lead to numerically stiff differential equations. This stiffness can introduce computational bottlenecks or add complexity to the analysis. In contrast, rigid models are easier to integrate and analyze, making them the preferred choice for theoretical or numerical investigations involving the stability of legged robots or trajectory planning. Notice that both the models in (B.31)-(B.32) preserve the unilateral constraint.

### B.3.2 Friction Cones

The contact interaction between the $k$-th rigid body and the environment can be categorized into two modes: fixed or sliding. A commonly used model of friction in robotics is Coulomb's law. This experimental law states that the friction force magnitude $f_k^n$ in the tangent plane at the contact interface is related to the normal force magnitude $\boldsymbol{f}_k^n$ [Kao et al., 2008]. Either of the following situations can occur [Wieber et al., 2016]:

- When the contact point is sliding on the contact surface, the tangential force is proportional to the normal one and is in the opposite direction of the motion:
$$\boldsymbol{f}_k^t = -\mu f_k^n \frac{\dot{\boldsymbol{p}}_k^t}{\|\dot{\boldsymbol{p}}_k^t\|} \quad \text{if} \quad \dot{\boldsymbol{p}}_k^t \neq 0, \tag{B.33}$$
being $\mu \in \mathbb{R}^+$ the static friction coefficient. Notice that the magnitude of the friction force is independent of the speed of sliding;
- When the contact point is sticking, i.e., it is fixed, the norm of the tangential force is bounded
$$\|\boldsymbol{f}_k^t\| \leq \mu f_k^n \quad \text{if} \quad \dot{\boldsymbol{p}}_k^t = 0. \tag{B.34}$$

The model is applicable when dealing with rigid, dry materials. The friction coefficient $\mu$ varies based on the specific materials involved in the contact, falling within the range of 0.1 to 1.

We can derive a sufficient condition for stability from the Coulomb friction law. A point contact remains in the fixed contact mode while its contact force satisfies the inequalities:

$$\text{unilaterality :} \qquad f_k^n > 0, \tag{B.35}$$

$$\text{non} - \text{slippage :} \qquad \|\boldsymbol{f}_k^t\| \leq \mu f_k^n. \tag{B.36}$$

The above equations are known as *friction cone* constraints and have a geometric interpretation. All forces that can be applied to the $k$-th body appendage that does not change its position is confined inside the cone $\mathcal{FC}_k$ having as axis the normal the contact surface and as half-angle atan $(\mu)$:

$$\mathcal{FC}_k = \{\boldsymbol{f}_k \in \mathbb{R}^3 | f_k^n > 0,\ \|\boldsymbol{f}_k^t\| \leq \mu f_k^n\}. \tag{B.37}$$

Although more realistic, this model presents some computational challenges. A common practice is to consider the linear approximation of (B.36):

$$\text{unilaterality :} \qquad f_k^n > 0, \tag{B.38}$$

$$\text{non} - \text{slippage :} \qquad \frac{\boldsymbol{f}_k^t}{\tilde{\mu} f_k^n} \in \mathcal{P}_n \tag{B.39}$$

where $\mathcal{P}_n$ is a regular $n$-sided polygon inscribed in the unit circle. This approximation can be made as close as desired to the original model by increasing the number of edges. A common selection is to use $n = 4$, resulting in a four-sided friction pyramid. The constraint can be succinctly expressed in matrix notation as

$$\boldsymbol{F}_k \boldsymbol{f}_k \leq \boldsymbol{0}, \tag{B.40}$$

where $\boldsymbol{F}_k = \boldsymbol{F}\boldsymbol{R}_\Psi^T$ represents the linearized friction cone model in the global frame, while $\boldsymbol{F}_k$ is utilized for the local contact frame, and its expression is

$$\boldsymbol{F} = \begin{bmatrix} -1 & 0 & -\tilde{\mu} \\ +1 & 0 & -\tilde{\mu} \\ 0 & -1 & -\tilde{\mu} \\ 0 & +1 & -\tilde{\mu} \\ 0 & 0 & -1 \end{bmatrix}. \tag{B.41}$$

For $\tilde{\mu} = \mu$, the linearized Coulomb cone is an outer approximation of the circular one; it is an inner approximation for $\tilde{\mu} = \mu/\sqrt{2}$. Pay attention to the fact that (B.40) is an expression for (B.27) in the case of a single contact.

# C

# Holonomic and Nonholonomic Mechanical Systems

In this appendix, our focus shifts toward holonomic and nonholonomic mechanical systems, an intriguing domain that has garnered interest in the robotic field. The significance of this classical subject has been amplified by its relevance to advanced robotic structures, including legged robots, space manipulators, and multi-fingered robot hands. Nonholonomic behavior in robotic systems implies that the mechanism can be completely controlled with a reduced number of actuators. On the other hand, both planning and control are much more difficult than in conventional holonomic systems and require special techniques.

Consider a mechanical system made by $n$ rigid bodies and described by the generalized coordinate vector $\boldsymbol{q}$, as in Appendix B. In numerous scenarios, the motion of the system is constrained, either from the inherent structure of the mechanism or from the specific methods employed for its actuation and control [De Luca and Oriolo, 1995]. There exist various classifications of the constraints. They may be expressed as equalities or inequalities (respectively, *bilateral* or *unilateral* constraints), and they may depend explicitly on time or not (*rheonomic* or *scleronomic* constraints). Here, we focus only on bilateral scleronomic constraints.

Motion restrictions that can be put in the form

$$h_i(\boldsymbol{q}) = 0 \qquad i = 1, \ldots, k < n \tag{C.1}$$

are termed *holonomic*[1] (or *integrable*) constraints, assuming that the functions $h_i : \mathcal{Q} \to \mathbb{R}$ are of class $C^\infty$, i.e., smooth, and independent. Holonomic constraints reduce space of attainable configuration to an $(n-k)$-dimensional smooth submanifold, embedded in $\mathcal{Q}$. A mechanical system whose constraints, if any, are all holonomic, is called a holonomic system. In principle, the Implicit Function Theorem allows one to solve (C.1) by expressing $k$ generalized coordinates in terms of the remaining $nk$, and eliminating them from the formulation of the problem. Nevertheless, this procedure has only local validity and may introduce singularities. A convenient alternative is to replace the original generalized coordinates with a reduced set of variables that effectively characterizes the DoFs of the system. The minimal vector of generalized coordinates is $(n-k)$-dimensional and its components are simultaneously

- independent: none of the generalized coordinates can be obtained as linear combinations of other ones,
- complete: the motion of the constrained set is completely determined by the generalized coordinates included in the set.

Holonomic constraints are generally the result of mechanical interconnection between the various bodies of the system. E.g., prismatic and revolute joints commonly used in robotic manipulators are a source of constraints in the form (C.1). As a matter of fact, a fixed-base kinematic chain of $n$ rigid bodies is a holonomic system, whose configuration space is $(\mathbb{R}^3 \times \mathrm{SO}(3))^n$ (position and orientation of each body). Since each elementary joint has a single GRF, i.e., it imposes five constraints, a minimal representation of the system has dimension $6n - 5n = n$.

System constraints whose expression involves both generalized coordinates and velocities as

$$a_i(\boldsymbol{q}, \dot{\boldsymbol{q}}) = 0 \qquad i = 1, \ldots, k < n$$

are called kinematic constraints. They limit the instantaneous admissible motion of the system by restricting the set of generalized velocities that can be attained at each configuration. In mechanics, such constraints are most usually encountered as linear functions of the generalized coordinates (*Pfaffian form*):

$$\boldsymbol{a}_i^T(\boldsymbol{q})\,\dot{\boldsymbol{q}} = 0 \qquad i = 1, \ldots, k < n, \qquad \text{or} \qquad \boldsymbol{A}^T(\boldsymbol{q})\,\dot{\boldsymbol{q}} = 0. \tag{C.2}$$

The functions $a_i : \mathcal{Q} \to \mathbb{R}$ are of class $C^\infty$ and independent.

[1] 'Holonomic' comes from the greek word ὅλος (read as 'olos') that means 'whole', 'entire'.

Clearly, the existence of $k$ holonomic constraints implies an equal number of kinematic constraints

$$\frac{\partial h_i(\boldsymbol{q})}{\partial t} = \frac{\partial h_i(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = 0 \qquad i = 1, \ldots, k,$$

but the converse is not necessarily true. Kinematic constraints of the form (C.2) may or may not be integrable as (C.1). In the negative case, the kinematic constraint, as well as the mechanical system itself, is called *nonholonomic*[2].

Nonholonomic constraints limit the system mobility in a completely different way if compared to holonomic constraints. To appreciate this fact, let us consider a single Pfaffian constraint

$$\boldsymbol{a}^T(\boldsymbol{q})\,\dot{\boldsymbol{q}} = \sum_{j=1}^{n} a_j(\boldsymbol{q})\,\dot{q}_j = 0. \tag{C.3}$$

If it is holonomic, then it can be integrated as

$$h(\boldsymbol{q}) = c,$$

with $\partial h/\partial \boldsymbol{q} = \gamma(\boldsymbol{q})\,\boldsymbol{a}^T(\boldsymbol{q})$ and and $c$ is an integrating constant depending on the initial configuration $\boldsymbol{q}_0$, i.e., $c = h(\boldsymbol{q}_0)$. Above $\gamma(\boldsymbol{q}) \neq 0$ is an integrating factor. In this case, there is a loss of *accessibility* in the configuration space. the motion of the mechanical system is confined to the $(n-1)$-dimensional submanifold dictated by a particular level surface of the scalar function $h(\boldsymbol{q}) = c$. Conversely, if (C.3) is nonholonomic, then the generalized velocities are instantaneously constrained to belong to a subspace of dimension $n-1$, i.e., to the null space of $\boldsymbol{a}^T(\boldsymbol{q})$, but it is still possible to reach any configuration in $\mathcal{Q}$. Therefore, the number of GRF is reduced to $n-1$, but the number of generalized coordinates cannot be reduced.

In summary, nonholonomic systems are, roughly speaking, mechanical systems with constraints on their velocity that are not derivable from position constraints [Bloch et al., 2005]. They arise, for instance, in mechanical systems that have rolling contact (for example, the rolling of wheels without slipping) or certain kinds of sliding contact (such as the sliding of skates). Another kind of nonholonomic behavior is found in free-floating articulated systems, such as manipulators used in space operations or legged systems during an aerial

[2]The term 'nonholonomic system' was coined by Hertz back in 1894 [Hertz, 1894].

phase. In the absence of contacts, the conservation of angular momentum is a non-integrable Pfaffian constraint, as pointed out in Section 2.2.

# D

# First Integrals of the Dynamics

The expression *first integrals* refers to quantities that remain constant throughout the system's dynamic evolution. They provide a useful understanding of systems' behaviors, often indicating conserved quantities or specific properties of the system. In the context of mechanical systems, they can denote, e.g., energy conservation, momentum conservation, or other significant invariants. In this appendix, we only refer to time-invariant, globally defined first integrals. More exhaustive discussions can be found in books about Ordinary Differential Equation (ODE) and classical mechanics, e.g., [Arnold, 1992; Goldstein et al., 2002].

Let $M$ be a smooth manifold of dimension $n$ and consider the vector field $\boldsymbol{f}$ mapping each point $\boldsymbol{x} \in M$ to a tangent vector $\boldsymbol{f}(\boldsymbol{x})$ in the tangent space $T_x M$. The vector field defines a system of autonomous first order Ordinary Differential Equation (ODE) in the unknown variables $\boldsymbol{x}(t) : \mathbb{R} \supseteq I \to M$:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{D.1}$$

A first integral of the dynamics is a non-constant continuously-differentiable function $F : M \to \mathbb{R}$ which is locally constant on any solution of (D.1), i.e.,

$$\frac{d}{dt} F(\boldsymbol{x}(t)) = 0, \qquad \forall \boldsymbol{x}(t) \text{ solving (D.1).} \tag{D.2}$$

The latter is equivalent to require that the derivative of $F$ in the direction of the field $\boldsymbol{f}$ (i.e., the Lie derivative $L_{\boldsymbol{f}} F$) must be zero:

$$\frac{d}{dt} F(\boldsymbol{x}(t)) = \frac{\partial F}{\partial \boldsymbol{x}} \frac{d\boldsymbol{x}}{dt} = L_{\boldsymbol{f}} F = 0$$

The name first integral is a relic of the times when mathematicians tried to solve all differential equations by integration. At that time, the name *integral* (or *partial integral*) was given to what we call a *solution*. Notice that if $F$ is a first integral of (D.1), each trajectory $\boldsymbol{x}(t)$ belongs to one and only one of the level sets of $F$.

Typically, the existence of such a function is not guaranteed. However, specific scenarios reveal automatic existence, such as in Hamiltonian systems where the energy function is a first integral. According to the Noether theorem, any (Lie-) symmetry also generates another first integral, like linear and angular momentum, aligning with the equations' translational and rotational invariance. In some other exceptional examples, e.g., the Lotka-Volterra system, a first integral can be found and shows that the solutions are periodic on concentric orbits.

### First Integrals of LIP Dynamics

In Section 2.2.2 we introduce the notion of orbital energy of the LIP model claiming that it is a first integral. Here, we want to show a procedure for computing it. Let us introduce the state of the LIP dynamics as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c^x \\ \dot{c}^x \end{bmatrix}. \tag{D.3}$$

Writing (2.12) as an autonomous system yields

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_2 \\ \omega^2 x_1 \end{bmatrix}. \tag{D.4}$$

We seek for a function $F : \mathbb{R}^2 \to \mathbb{R}$ such that

$$\frac{\partial F}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} = \begin{bmatrix} \phi_1(\boldsymbol{x}) & \phi_2(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} x_2 \\ \omega^2 x_1 \end{bmatrix} = 0, \tag{D.5}$$

with $\phi_1, \phi_2 : \mathbb{R}^2 \to \mathbb{R}$ being the partial derivatives of $F$:

$$\phi_1(\boldsymbol{x}) = \frac{\partial F}{\partial x_1} \qquad \phi_2(\boldsymbol{x}) = \frac{\partial F}{\partial x_2} \tag{D.6}$$

Differently saying, given a solution $\boldsymbol{x}$ of the dynamics, we require that $\boldsymbol{f}(\boldsymbol{x})$ belongs to the null space of the Jacobian of $F$. Inspecting (D.5), a possible

choice for $\phi_1, \phi_2$ is

$$\phi_1(\boldsymbol{x}) = -k\omega^2 x_1 \qquad \phi_2(\boldsymbol{x}) = kx_2 \tag{D.7}$$

with $k \in \mathbb{R}$. Integrating (D.7) about the state and summing up, the family of first integrals parameterized by $k$ yields:

$$F(k;\, \boldsymbol{x}) = -\frac{1}{2}k\omega^2 x_1^2 + \frac{1}{2}kx_2^2. \tag{D.8}$$

By choosing $k = 1$ and recovering the original meaning of the state variables, one can recognize (2.17).

# E

# Orientation Error

Let us consider two rotation matrices $\boldsymbol{R}_a$, $\boldsymbol{R}_b \in \mathrm{SO}(3)$ describing two orientations in the 3-dimensional space with respect to the same reference frame, e.g., the world frame. The matrix $\boldsymbol{E} = \boldsymbol{R}_a^T \boldsymbol{R}_b$ represents the difference rotation matrix. If $\boldsymbol{R}_a$ and $\boldsymbol{R}_b$ describes actual and desired orientations, $\boldsymbol{E}$ is the error matrix between the two. Directly using $\boldsymbol{E}$ for controlling rotations is limited, due to difficulty of simultaneously handling the 9 elements of the matrix and the 6 constraints due to the fact that $\boldsymbol{E}\boldsymbol{E}^T = \boldsymbol{E}^T\boldsymbol{E} = \boldsymbol{I}_{3\times 3}$ since $\boldsymbol{E} \in \mathrm{SO}(3)$. Therefore, it is convenient to describe the orientation error with a minimal representation, i.e., using a vector $\boldsymbol{e} \in \mathbb{R}^3$. In the following, a procedure for defining such error vector is reported, relying on the angle-axis parametrization. The rotation defined by $\boldsymbol{E}$ can be represented with a unit vector $\hat{\boldsymbol{r}} \in \mathbb{R}^3$, indicating the direction of an axis of rotation, and an angle $\varphi \in [-\pi/2, \pi/2]$, describing the magnitude and sense of the rotation about the axis. These relate to the rotation matrix $\boldsymbol{E}$ as

$$\varphi = \arccos\left(\frac{\mathrm{tr}\,(\boldsymbol{E}) - 1}{2}\right), \qquad \hat{\boldsymbol{r}} = \frac{1}{2\,\sin\varphi}\begin{bmatrix} E_{32} - E_{23} \\ E_{13} - E_{31} \\ E_{21} - E_{12} \end{bmatrix},$$

where $\mathrm{tr}\,(\boldsymbol{E})$ is the trace of the matrix $\boldsymbol{E}$. The error vector in the actual frame is defined in [Luh et al., 1980] as

$${}_a\boldsymbol{e} = \hat{\boldsymbol{r}}\,\sin\varphi = \frac{1}{2}\begin{bmatrix} E_{32} - E_{23} \\ E_{13} - E_{31} \\ E_{21} - E_{12} \end{bmatrix}. \tag{E.1}$$

To restore the error in the original reference frame, one needs to pre-multiply by $\boldsymbol{R}_a$:

$$\boldsymbol{e} = \boldsymbol{R}_{a\,a}\boldsymbol{e}. \tag{E.2}$$

There are several other techniques for finding the error vector, relying on the different representations of rotations in the 3-dimensional space: angle-axis, Euler angles and unit quaternions. Further details can be found in [Diebel, 2006; Nakanishi et al., 2008; Solà, 2015; Yuan, 1988].

# Bibliography


Abdelrahman Abdalla, Michele Focchi, Romeo Orsolino, and Claudio Semini. An efficient paradigm for feasibility guarantees in legged locomotion. *IEEE Transactions on Robotics*, 2023. https://doi.org/10.1109/tro.2023.3280431.

Janete Alves, Nuno Peixinho, Miguel Tavares da Silva, Paulo Flores, and Hamid M Lankarani. A comparative study of the viscoelastic constitutive models for frictionless contact interfaces in solids. *Mechanism and Machine Theory*, 85:172–188, 2015. https://doi.org/10.1016/j.mechmachtheory.2014.11.020.

Jiajun An, TY Chung, Chun Ho David Lo, Carlos Ma, Xiangyu Chu, and KW Samuel Au. Development of a bipedal hopping robot with morphable inertial tail for agile locomotion. In *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, pages 132–139. IEEE, 2020. https://doi.org/10.1109/biorob49111.2020.9224428.

Jiajun An, Xin Ma, Chun Ho David Lo, Weeshen Ng, Xiangyu Chu, and Kwok Wai Samuel Au. Design and experimental validation of a monopod robot with 3-dof morphable inertial tail for somersault. *IEEE/ASME Transactions on Mechatronics*, 2022. https://doi.org/10.1109/tmech.2022.3167990.

Keisuke Arikawa and Tsutomu Mita. Design of multi-dof jumping robot. In *2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3992–3997. IEEE, 2002. https://doi.org/10.1109/robot.2002.1014358.

Vladimir I. Arnold. *Ordinary differential equations*. Springer Science & Business Media, 1992.

Jean-Pierre Aubin. *Viability theory*. Birkhäuser Boston, MA, 2009. https://doi.org/10.1007/978-0-8176-4910-4.

Victor Barasuol, Jonas Buchli, Claudio Semini, Marco Frigerio, Edson R. De Pieri, and Darwin G. Caldwell. A reactive controller framework for quadrupedal locomotion on challenging terrain. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2554–2561. IEEE, 2013. https://doi.org/10.1109/icra.2013.6630926.

Guillaume Bellegarda and Quan Nguyen. Robust quadruped jumping via deep reinforcement learning. *arXiv preprint arXiv:2011.07089*, 2020.

Mike B. Bennett and Greg C. Taylor. Scaling of elastic strain energy in kangaroos and the benefits of being big. *Nature*, 378(6552):56–59, 1995. https://doi.org/10.1038/378056a0.

Jeffrey T Bingham, Jeongseok Lee, Ravi N Haksar, Jun Ueda, and C Karen Liu. Orienting in mid-air through configuration changes to achieve a rolling landing for reducing impact after a fall. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3610–3617. IEEE, 2014. https://doi.org/10.1109/iros.2014.6943068.

Gerardo Bledt. *Regularized predictive control framework for robust dynamic legged locomotion*. PhD thesis, Massachusetts Institute of Technology, 2020. URL `https://dspace.mit.edu/handle/1721.1/125485`.

Reinhard Blickhan. The spring-mass model for running and hopping. *Journal of biomechanics*, 22(11-12):1217–1227, 1989. https://doi.org/10.1016/0021-9290(89)90224-8.

Anthony M. Bloch, Jerrold E. Marsden, and Dmitry V. Zenkov. Nonholonomic dynamics. *Notices Of The American Mathematical Society (AMS)*, 52(3): 324–333, 2005.

Timothy Bretl and Sanjay Lall. Testing static equilibrium for legged robots. *IEEE Transactions on Robotics*, 24(4):794–807, 2008. https://doi.org/10.1109/tro.2008.2001360.

Bernard Brogliato. Distributional model of impacts. In *Nonsmooth Impact Mechanics: Models, Dynamics and Control*, pages 1–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. https://doi.org/10.1007/BFb0027734.

H. Benjamin Brown and Yangsheng Xu. A single-wheel, gyroscopically stabilized robot. In *1996 IEEE International Conference on*

*Robotics and Automation ICRA*, volume 4, pages 3658–3663. IEEE, 1996. https://doi.org/10.1109/robot.1996.509270.

Travis L. Brown and James P. Schmiedeler. Reaction wheel actuation for improving planar biped walking efficiency. *IEEE Transactions on Robotics*, 32:1290–1297, 2016. https://doi.org/10.1109/tro.2016.2593484.

Riccardo Bussola, Michele Focchi, Andrea Del Prete, Daniele Fontanelli, and Luigi Palopoli. Efficient reinforcement learning for jumping monopods. *arXiv preprint arXiv:2309.07038*, 2023.

Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019. https://doi.org/10.1109/sii.2019.8700380.

Chi-Tsong Chen. *Linear system theory and design*, chapter 4, pages 106 – 117. Oxford University Press, 3 edition, 1999.

Matthew Chignoli and Sangbae Kim. Online trajectory optimization for dynamic aerial motions of a quadruped robot. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7693–7699. IEEE, 2021. https://doi.org/10.1109/icra48506.2021.9560855.

Matthew Chignoli and Patrick M. Wensing. Variational-based optimal control of underactuated balancing for dynamic quadrupeds. *IEEE Access*, 8:49785–49797, 2020. https://doi.org/10.1109/access.2020.2980446.

Matthew Chignoli, Savva Morozov, and Sangbae Kim. Rapid and reliable quadruped motion planning with omnidirectional jumping. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6621–6627. IEEE, 2022. https://doi.org/10.1109/icra46639.2022.9812088.

Jennifer Chu. MIT Cheetah robot lands the running jump, 2015. URL `https://news.mit.edu/2015/cheetah-robot-lands-running-jump-0529`. Accessed on 10 December 2023.

Xiangyu Chu, Chun Ho David Lo, Carlos Ma, and Kwok Wai Samuel Au. Null-space-avoidance-based orientation control framework for underactuated,

tail-inspired robotic systems in flight phase. *IEEE Robotics and Automation Letters*, 4(4):3916–3923, 2019. https://doi.org/10.1109/lra.2019.2928759.

Erwin Coumans and Yunfei Bai. Pybullet, a Python module for physics simulation for games, robotics and machine learning. 2016.

Alessandro De Luca and Giuseppe Oriolo. Modelling and control of nonholonomic mechanical systems. In *Kinematics and dynamics of multi-body systems*, pages 277–342. Springer, 1995. https://doi.org/10.1007/978-3-7091-4362-9_7.

Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9. IEEE, 2018. https://doi.org/10.1109/iros.2018.8594448.

Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL `http://www.programmingvision.com/rosen_diankov_thesis.pdf`.

James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

Yanran Ding and Hae-Won Park. Design and experimental implementation of a quasi-direct-drive leg for optimized jumping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 300–305. IEEE, 2017. https://doi.org/10.1109/iros.2017.8202172.

Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8484–8490. IEEE, 2019. https://doi.org/10.1109/icra.2019.8793669.

Yanran Ding, Chuanzheng Li, and Hae-Won Park. Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3998–4005. IEEE, 2020.

Yanran Ding, Abhishek Pandala, Chuanzheng Li, Young-Ha Shin, and Hae-Won Park. Representation-free model predictive control for dynamic mo-

tions in quadrupeds. *IEEE Transactions on Robotics*, 37(4):1154–1171, 2021a. https://doi.org/10.1109/TRO.2020.3046415.

Yanran Ding, Mengchao Zhang, Chuanzheng Li, Hae-Won Park, and Kris Hauser. Hybrid sampling / optimization-based planning for agile jumping robots on challenging terrains. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2839–2845. IEEE, 2021b. https://doi.org/10.1109/icra48506.2021.9561939.

Iain S Duff. Ma57 – A code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software (TOMS)*, 30(2):118–144, 2004. https://doi.org/10.1145/992200.992202.

Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2008. https://doi.org/10.1007/978-1-4899-7560-7.

Michele Focchi, Andrea Del Prete, Ioannis Havoutis, Roy Featherstone, Darwin G Caldwell, and Claudio Semini. High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, 41:259–272, 2017. https://doi.org/10.1007/s10514-016-9573-1.

Michele Focchi, Mohamed Bensaadallah, Marco Frego, Angelika Peer, Daniele Fontanelli, Andrea Del Prete, and Luigi Palopoli. Clio: a novel robotic solution for exploration and rescue missions in hostile mountain environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7742–7748. IEEE, 2023. https://doi.org/10.1109/icra48891.2023.10160440.

Michele Focchi, Francesco Roscia, and Claudio Semini. Locosim: An open-source cross-platform robotics framework. In Ebrahim Samer El Youssef, Mohammad Osman Tokhi, Manuel F. Silva, and Leonardo Mejia Rincon, editors, *Synergetic Cooperation between Robots and Humans. Proceedings of the CLAWAR 2023 Conference*, pages 395–406. Springer Nature Switzerland, 2024. https://doi.org/10.1007/978-3-031-47272-5_33.

Marco Frigerio, Jonas Buchli, Darwin G. Caldwell, and Claudio Semini. RobCoGen: a code generator for efficient kinematics and dynamics of articulated robots, based on domain specific languages. *Journal of Software Engineering for Robotics (JOSER)*, 7(1):36–54, 2016.

Yuni Fuchioka, Zhaoming Xie, and Michiel Van de Panne. Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5092–5098. IEEE, 2023. https://doi.org/10.1109/icra48891.2023.10160562.

Mohanarajah Gajamohan, Michael Merz, Igor Thommen, and Raffaello D'Andrea. The Cubli: A cube that can jump up and balance. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3722–3727. IEEE, 2012. https://doi.org/10.1109/iros.2012.6385896.

Gabriel García, Robert Griffin, and Jerry Pratt. Time-varying model predictive control for highly dynamic motions of quadrupedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7344–7349. IEEE, 2021. https://doi.org/10.1109/icra48506.2021.9561913.

Carl Friedrich Gauss. Über ein neues allgemeines Grundgesetz der Mechanik. 1829.

Christian Gehring, Stelian Coros, Marco Hutter, Carmine Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Péter Fankhauser, Jemin Hwangbo, Markus A. Hoepflinger, and Roland Siegwar. Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot. *IEEE Robotics & Automation Magazine*, 23(1):34–43, 2016. https://doi.org/10.1109/mra.2015.2505910.

Herbert Goldstein, Charles Poole, and John Safko. *Classical mechanics*. American Association of Physics Teachers, 2002.

Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Thomas Flayols, Jonathan Fiene, Alexander Badri-Spröwitz, and Ludovic Righetti. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5 (2):3650–3657, 2020. https://doi.org/10.1109/lra.2020.2976639.

Sehoon Ha, Yuting Ye, and C Karen Liu. Falling and landing motion control for character animation. *ACM Transactions on Graphics (TOG)*, 31(6):1–9, 2012. https://doi.org/10.1145/2366145.2366174.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585 (7825):357–362, September 2020. https://doi.org/10.1038/s41586-020-2649-2.

Heinrich Hertz. Die Prinzipen der Mechanik in neuem Zusammenhange dargestellt. In *Gesamelte Werke, Band III*. Barth, 1894.

Alexander Herzog, Ludovic Righetti, Felix Grimminger, Peter Pastor, and Stefan Schaal. Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 981–988. IEEE, 2014. https://doi.org/10.1109/iros.2014.6942678.

Jessica K. Hodgins and Marc H. Raibert. Biped gymnastics. *The International Journal of Robotics Research*, 9(2):115–128, 1990. https://doi.org/10.1177/027836499000900209.

At L. Hof, Marnix G.J. Gazendam, and Wim E. Sinke. The condition for dynamic stability. *Journal of biomechanics*, 38(1):1–8, 2005. https://doi.org/10.1016/j.jbiomech.2004.03.025.

Nathaniel H. Hunt, Judy Jinn, Lucia F. Jacobs, and Robert J. Full. Acrobatic squirrels learn to leap and land on tree branches without falling. *Science*, 373(6555):697–700, 2021. https://doi.org/10.1126/science.abe5753.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. https://doi.org/10.1126/scirobotics.aau5872.

Se Hwan Jeon, Sangbae Kim, and Donghyun Kim. Online optimal landing control of the MIT Mini Cheetah. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 178–184. IEEE, 2022. https://doi.org/10.1109/icra46639.2022.9811796.

Aaron M. Johnson, Thomas Libby, Evan Chang-Siu, Masayoshi Tomizuka, Robert J Full, and Daniel E Koditschek. Tail assisted dynamic self righting. In *Adaptive mobile robotics*, pages 611–620. World Scientific, 2012. https://doi.org/10.1142/9789814415958_0079.

Lentin Joseph and Jonathan Cacace. *Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System*. Packt Publishing Ltd, 2018.

Ardian Jusufi, Yu Zeng, Robert J. Full, and Robert Dudley. Aerial Righting Reflexes in Flightless Animals. *Integrative and Comparative Biology*, 51(6): 937–943, 09 2011. https://doi.org/10.1093/icb/icr114.

Shuuji Kajita and Kazuo Tani. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In *1991 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1405–1406. IEEE, 1991. https://doi.org/10.1109/robot.1991.131811.

Thomas R. Kane and M. P. Scher. A dynamical explanation of the falling cat phenomenon. *International journal of solids and structures*, 5(7):663–670, 1969. https://doi.org/10.1016/0020-7683(69)90086-9.

Imin Kao, Kevin Lynch, and Joel W. Burdick. *Contact Modeling and Manipulation*, pages 647–669. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. https://doi.org/10.1007/978-3-540-30301-5_28.

Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini Cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE, 2019. https://doi.org/10.1109/icra.2019.8793865.

Nathan Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154. IEEE, 2004. https://doi.org/10.1109/iros.2004.1389727.

Hendrik Kolvenbach, Elias Hampp, Patrick Barton, Radek Zenkl, and Marco Hutter. Towards jumping locomotion for quadruped robots on the Moon. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5459–5466, 2019. https://doi.org/10.1109/iros40897.2019.8967552.

Twan Koolen, Tomas De Boer, John Rebula, Ambarish Goswami, and Jerry Pratt. Capturability - based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models. *The international journal of robotics research*, 31(9):1094–1113, 2012. https://doi.org/10.1177/0278364912452673.

Vince Kurtz, He Li, Patrick M. Wensing, and Hai Lin. Mini Cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4635–4641. IEEE, 2022. https://doi.org/10.1109/icra46639.2022.9812120.

Cornelius Lanczos. *Variational principles of mechanics*. Dover, 1986.

Chenhao Li, Marin Vlastelica, Sebastian Blaes, Jonas Frey, Felix Grimminger, and Georg Martius. Learning agile skills via adversarial imitation of rough partial demonstrations. In *Conference on Robot Learning (CoRL)*, pages 342–352. PMLR, 2023.

He Li and Patrick M. Wensing. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, 2020. https://doi.org/10.1109/lra.2020.3007475.

Henry Lieberman, Fabio Paternò, and Volker Wulf. *End user development*, volume 9. Springer, 2006. https://doi.org/10.1007/1-4020-5386-X.

Yiping Liu, Patrick M. Wensing, David E. Orin, and Yuan F. Zheng. Trajectory generation for dynamic walking in a humanoid over uneven terrain using a 3D-actuated dual-SLIP model. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 374–380. IEEE, 2015. https://doi.org/10.1109/iros.2015.7353400.

JYSM Luh, M Walker, and R Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, 1980. https://doi.org/10.1109/TAC.1980.1102367.

Daniel J. Lynch, Kevin M. Lynch, and Paul B. Umbanhowar. The soft-landing problem: Minimizing energy loss by a legged robot impacting yielding terrain. *IEEE Robotics and Automation Letters*, 5(2):3658–3665, 2020. https://doi.org/10.1109/lra.2020.2977260.

Étienne-Jules Marey. Photographs of a tumbling cat. *Nature*, 51(1308):80–81, 1894. https://doi.org/10.1038/051080a0.

Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542. IEEE, 2020. https://doi.org/10.1109/icra40945.2020.9196673.

Nikita Mikhalkov, Alexey Prutskiy, Semyon Sechenev, Dmitry Kazakov, Alexey Simulin, Dmitry Sokolov, and Igor Ryadchikov. Gyrubot: nonanthropomorphic stabilization for a biped. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4976–4982. IEEE, 2021. https://doi.org/10.1109/icra48506.2021.9561214.

Enrico Mingo Hoffman and Antonio Paolillo. Exploiting visual servoing and centroidal momentum for whole-body motion control of humanoid robots in absence of contacts and gravity. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2979–2985. IEEE, 2021. https://doi.org/10.1109/icra48506.2021.9560739.

Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

Mostafa R. A. Nabawy, Girupakaran Sivalingam, Russell J. Garwood, William J. Crowther, and William I. Sellers. Energy and time optimal trajectories in exploratory jumps of the spider Phidippus Regius. *Scientific reports*, 8(1):1–15, 2018. https://doi.org/10.1038/s41598-018-25227-9.

Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008. https://doi.org/10.1177/0278364908091463.

Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018. https://doi.org/10.1109/lra.2018.2800124.

Chuong Nguyen and Quan Nguyen. Contact-timing and trajectory optimization for 3D jumping on quadruped robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11994–11999. IEEE, 2022. https://doi.org/10.1109/iros47612.2022.9981284.

Chuong Nguyen, Lingfan Bao, and Quan Nguyen. Continuous jumping for legged robots on stepping stones via trajectory optimization and model predictive control. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 93–99. IEEE, 2022. https://doi.org/10.1109/cdc51059.2022.9993259.

Quan Nguyen, Matthew J Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the mit cheetah 3 robot. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7448–7454. IEEE, 2019. https://soi.org/10.1109/ICRA.2019.8794449.

Espen Oland and Rune Schlanbusch. Reaction wheel design for cubesats. In *2009 4th International Conference on Recent Advances in Space Technologies*, pages 778–783. IEEE, 2009. https://doi.org/10.1109/rast.2009.5158296.

Romeo Orsolino, Michele Focchi, Carlos Mastalli, Hongkai Dai, Darwin G Caldwell, and Claudio Semini. Application of wrench-based feasibility analysis to the online trajectory optimization of legged robots. *IEEE Robotics and Automation Letters*, 3(4):3363–3370, 2018. https://doi.org/10.1109/lra.2018.2836441.

Romeo Orsolino, Michele Focchi, Stéphane Caron, Gennaro Raiola, Victor Barasuol, Darwin G Caldwell, and Claudio Semini. Feasible region: An actuation-aware extension of the support region. *IEEE Transactions on Robotics*, 36(4):1239–1255, 2020. https://doi.org/10.1109/TRO.2020.2983318.

Jong-Shi Pang and Jeff Trinkle. Stability characterizations of rigid body contact problems with coulomb friction. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 80(10):643–663, 2000. https://doi.org/10.1002/1521-4001(200010)80:10<643::aid-zamm643>3.0.co;2-e.

Hae-Won Park, Meng Yee Chuah, and Sangbae Kim. Quadruped bounding control with variable duty cycle via vertical impulse

scaling. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3245–3252. IEEE, 2014. https://doi.org/10.1109/iros.2014.6943013.

Hae-Won Park, Patrick M. Wensing, and Sangbae Kim. High-speed bounding with the MIT Cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017. https://doi.org/10.1177/0278364917694244.

Hae-Won Park, Patrick M. Wensing, and Sangbae Kim. Jumping over obstacles with MIT Cheetah 2. *Robotics and Autonomous Systems*, 136:103703, 2021. https://doi.org/10.1016/j.robot.2020.103703.

Amir Patel and Martin Braae. Rapid turning at high-speed: Inspirations from the cheetah's tail. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5506–5511. IEEE, 2013. https://doi.org/10.1109/iros.2013.6697154.

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017. ISSN 0730-0301. https://doi.org/10.1145/3072959.3073602.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018. https://doi.org/10.1145/3197517.3201311.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

Ioannis Poulakakis and Jessy W. Grizzle. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793, 2009. https://doi.org/10.1109/tac.2009.2024565.

Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS international conference on humanoid robots*, pages 200–207. IEEE, 2006. https://doi.org/10.1109/ichr.2006.321385.

Jerry E. Pratt and Sergey V. Drakunov. Derivation and application of a conserved orbital energy for the inverted pendulum bipedal walking model. In *2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4653–4660. IEEE, 2007. https://doi.org/10.1109/robot.2007.364196.

Ji Qi, Haibo Gao, Haitao Yu, Mingying Huo, Wenyu Feng, and Zongquan Deng. Integrated attitude and landing control for quadruped robots in asteroid landing mission scenarios using reinforcement learning. *Acta Astronautica*, 2022. https://doi.org/10.1016/j.actaastro.2022.11.028.

Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Learning humanoid locomotion with transformers. *arXiv preprint arXiv:2303.03381*, 2023.

Marc H. Raibert. *Legged robots that balance*. MIT press, 1986. https://doi.org/10.1109/mex.1986.4307016.

Aswin K Ramasubramanian, Robins Mathew, Matthew Kelly, Vincent Hargaden, and Nikolaos Papakostas. Digital twin for human–robot collaboration in manufacturing: Review and outlook. *Applied Sciences*, 12(10):4811, 2022. https://doi.org/10.3390/app12104811.

Siavash Rezazadeh, Christian Hubicki, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Andy Abate, and Jonathan Hurst. Spring-mass walking with ATRIAS in 3D: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot. In *Dynamic Systems and Control Conference*, volume 57243. American Society of Mechanical Engineers, 2015. https://doi.org/10.1115/dscc2015-9899.

Universal Robot. Ur5, 2008. URL `https://www.universal-robots.com/products/ur5-robot/`. Accessed on 10 December 2023.

Eric Rohmer, Surya P. N. Singh, and Marc Freese. CoppeliaSim (formerly V-REP): A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 1321–1326. IEEE, 2013. https://doi.org/10.1109/iros.2013.6696520.

Francesco Roscia, Andrea Cumerlotti, Andrea Del Prete, Claudio Semini, and Michele Focchi. Orientation Control System: Enhancing Aerial Maneuvers for Quadruped Robots. *Sensors*, 23(3):1234, 2023a. https://doi.org/10.3390/s23031234.

Francesco Roscia, Michele Focchi, Andrea Del Prete, Darwin G. Caldwell, and Claudio Semini. Reactive landing controller for quadruped robots. *IEEE Robotics and Automation Letters*, 2023b. https://doi.org/10.1109/lra.2023.3313919.

Edward John Routh. *Dynamics of a system of rigid bodies, part 1.* Dover Publications, 1905.

Nikita Rudin, Hendrik Kolvenbach, Vassilios Tsounis, and Marco Hutter. Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning. *IEEE Transactions on Robotics*, 38:317–328, 2021. https://doi.org/10.1109/tro.2021.3084374.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 91–100. PMLR, 2022.

S. Schaal. The SL simulation and real-time control software package. Technical report, Los Angeles, CA, 2009. URL `http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf`.

John M. Schmitt and Jonathan Clark. Modeling posture-dependent leg actuation in sagittal plane locomotion. *Bioinspiration & biomimetics*, 4(4): 046005, 2009. https://doi.org/10.1088/1748-3182/4/4/046005.

Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of HyQ–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011. https://doi.org/10.1109/biorob.2008.4762913.

Claudio Semini, Victor Barasuol, Thiago Boaventura, Marco Frigerio, Michele Focchi, Darwin G Caldwell, and Jonas Buchli. Towards versatile legged robots through active impedance control. *The*

*International Journal of Robotics Research*, 34(7):1003–1020, 2015. https://doi.org/10.1177/0278364915578839.

Junjie Shen, Yeting Liu, Xiaoguang Zhang, and Dennis Hong. Optimized jumping of an articulated robotic leg. In *2020 17th International Conference on Ubiquitous Robots (UR)*, pages 205–212. IEEE, 2020. https://doi.org/10.1109/ur49135.2020.9144799.

Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2009. https://doi.org/10.1007/978-1-84628-642-1.

Joan Solà. Quaternion kinematics for the error-state kalman filter. *ArXiv*, 2015.

Zhitao Song, Linzhu Yue, Guangli Sun, Yihu Ling, Hongshuo Wei, Linhai Gui, and Yun-Hui Liu. An optimal motion planning framework for quadruped jumping. *arXiv preprint arXiv:2207.12002*, 2022.

Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, 1984.

Yunxi Tang, Jiajun An, Xiangyu Chu, Shengzhi Wang, Ching Yan Wong, and KW Samuel Au. Towards safe landing of falling quadruped robots using a 3-dof morphable inertial tail. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1141–1147. IEEE, 2023. https://doi.org/10.1109/icra48891.2023.10161422.

Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL `https://drake.mit.edu`. Accessed on 10 December 2023.

Unitree. Aliengo, 2019. URL `https://www.unitree.com/en/aliengo`. Accessed on 10 December 2023.

Unitree. Go1, 2021. URL `https://www.unitree.com/en/go1/`. Accessed on 10 December 2023.

Vasileios Vasilopoulos, Konstantinos Machairas, and Evangelos Papadopoulos. Quadruped pronking on compliant terrains using a reaction wheel. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3590–3595. IEEE, 2016. https://doi.org/10.1109/icra.2016.7487542.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006. https://doi.org/10.1007/s10107-004-0559-y.

Garrett Wenger, Avik De, and Daniel E. Koditschek. Frontal plane stabilization and hopping with a 2DOF tail. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 567–573. IEEE, 2016. https://doi.org/10.1109/iros.2016.7759110.

Pierre-Brice Wieber. Holonomy and nonholonomy in the dynamics of articulated motion. In *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*, pages 411–425. Springer, 2006. https://doi.org/10.1007/978-3-540-36119-0_20.

Pierre-Brice Wieber, Russ Tedrake, and Scott Kuindersma. Modeling and control of legged robots. In *Springer handbook of robotics*, pages 1203–1234. Springer, 2016. https://doi.org/10.1007/978-3-319-32552-1_48.

Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, volume 39, pages 213–224. Wiley Online Library, 2020. https://doi.org/10.1111/cgf.14115.

Xiaobin Xiong and Aaron D Ames. Bipedal hopping: Reduced-order model embedding via optimization-based control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3821–3828. IEEE, 2018. https://doi.org/10.1109/iros.2018.8593547.

Xiaobin Xiong and Aaron D Ames. Sequential motion planning for bipedal somersault via flywheel SLIP and momentum transmission with task space control. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3510–3517. IEEE, 2020. https://doi.org/10.1109/iros45743.2020.9341467.

Yuxiang Yang, Xiangyun Meng, Wenhao Yu, Tingnan Zhang, Jie Tan, and Byron Boots. Continuous versatile jumping using learned action residuals. In *Learning for Dynamics and Control Conference*, pages 770–782. PMLR, 2023.

Justin K. Yim, Bajwa Roodra Pratap Singh, Eric K. Wang, Roy Featherstone, and Ronald S. Fearing. Precision robotic leaping and landing using stance-phase balance. *IEEE Robotics and Automation Letters*, 5(2):3422–3429, 2020. https://doi.org/10.1109/lra.2020.2976597.

Hyungjoo Yoon and Panagiotis Tsiotras. Spacecraft adaptive attitude and power tracking with variable speed control moment gyroscopes. *Journal of Guidance, Control, and Dynamics*, 25(6):1081–1090, 2002. https://doi.org/10.2514/2.4987.

JS Yuan. Closed-loop manipulator control using quaternion feedback. *IEEE Journal on Robotics and Automation*, 4(4):434–440, 1988. https://doi.org/10.1109/56.809.

Petr Zaytsev, S Javad Hasaneini, and Andy Ruina. Two steps is enough: No need to plan far ahead for walking balance. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6295–6300. IEEE, 2015. https://doi.org/10.1109/icra.2015.7140083.

Jingwen Zhang, Junjie Shen, Yeting Liu, and Dennis W Hong. Design of a jumping control framework with heuristic landing for bipedal robots. *arXiv preprint arXiv:2304.00536*, 2023.

Jianguo Zhao, Jing Xu, Bingtuan Gao, Ning Xi, Fernando J. Cintron, Matt W. Mutka, and Li Xiao. MSU jumper: A single-motor-actuated miniature steerable jumping robot. *IEEE Transactions on Robotics*, 29(3):602–614, 2013. https://doi.org/10.1109/tro.2013.2249371.

# List of Publications

## Journal Papers

**Francesco Roscia**, Andrea Cumerlotti, Andrea Del Prete, Claudio Semini, and Michele Focchi, "Orientation Control System: Enhancing Aerial Maneuvers for Quadruped Robots," in *MDPI Sensors*, vol. 23, no. 3, 1234, 2023, https://doi.org/10.3390/s23031234.

**Francesco Roscia**, Michele Focchi, Andrea Del Prete, Darwin G. Caldwell, and Claudio Semini, "Reactive Landing Controller for Quadruped Robots," in *IEEE Robotics and Automation Letters*, vol. 8, issue 11, 2023, https://doi.org/10.1109/LRA.2023.3313919.

## Conference Papers

Michele Focchi*, **Francesco Roscia***, and Claudio Semini, "Locosim: an Open-Source Cross-Platform Robotics Framework," In *Synergetic Cooperation between Robots and Humans. Proceedings of the CLAWAR 2023 Conference*, Springer Nature Switzerland, 2024, https://doi.org/10.1007/978-3-031-47272-5_33.

* Equal contribution.

# Curriculum Vitæ

## Francesco Roscia

14 January 1996 Born in Rome, Italy

## Education

2010 – 2015 Secondary School
Liceo Scientifico Statale Plinio Seniore, Rome

2015 – 2018 Bachelor in Computer and Automation Engineering
Sapienza, Università di Roma, Rome

*Thesis:* Effects of Singular Configurations on Kinematic Control of the Robot Manipulator KUKA KR5 SIXX R650

*Supervisor:* Prof. A. De Luca

2018 – 2020 Master of Science in Automation Engineering
Sapienza, Università di Roma, Rome

*Thesis:* Gap-crossing with the CENTAURO Robot: Planning Via Probabilistic Sampling and Nonlinear Optimization

*Supervisors:* Prof. G. Oriolo, Dr. N. Tsagarakis, Dr. E. Mingo Hoffmann

2020 – 2024 Ph. D. in Bioengineering and Robotics - Curriculum Advanced Robotics
Istituto Italiano di Tecnologia, Genoa, and Università di Genova, Genoa

*Thesis:* Towards Safe and Stable Landing Control for Quadruped Robots

*Supervisors:* Dr. M. Focchi, Prof. A. Del Prete, Dr. C. Semini