Reactive Landing Controller for Quadruped Robots

Francesco Roscia[1], Michele Focchi[1,2], Andrea Del Prete[3], Darwin G. Caldwell[4], and Claudio Semini[1]

***Abstract*—Quadruped robots are machines intended for challenging and harsh environments. Despite the progress in locomotion strategy, 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 work, we propose an optimization-based reactive Landing Controller that uses only proprioceptive measures for torque-controlled quadruped robots that free-fall on a flat horizontal ground, knowing neither the distance to the landing surface nor the flight time. Based on an estimate of the Center of Mass horizontal velocity, the method uses the Variable Height Springy Inverted Pendulum model for continuously recomputing the feet position while the robot is falling. In this way, the quadruped is ready to attain a successful landing in all directions, even in the presence of significant horizontal velocities. The method is demonstrated to dramatically enlarge the region of horizontal velocities that can be dealt with by a naive approach that keeps the feet still during the airborne stage. To the best of our knowledge, this is the first time that a quadruped robot can successfully recover from falls with horizontal velocities up to 3 $\mathrm{m/s}$ in simulation. Experiments prove that the used platform, Go1, can successfully attain a stable standing configuration from falls with various horizontal velocities and different angular perturbations.**

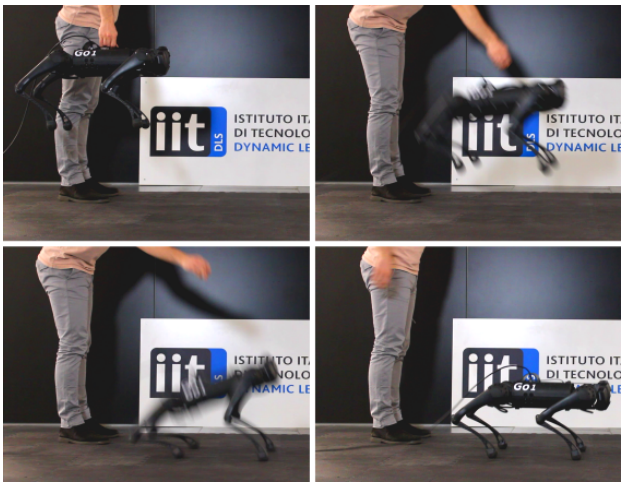


Fig. 1. Based only on proprioceptive measures, during a fall the proposed Landing Controller (LC) adjusts the limbs posture in order to drive the robot to stable standing configuration after Touch Down (TD), avoiding bounces, feet slippage and undesirable contacts with the ground.

I. Introduction

LEGGED robots are designed to traverse rough terrains. Thanks to the progress of the last decades, they have become lighter and stronger, enabling agile locomotion. Many types of gaits, such as trotting or crawling, have been investigated and successfully developed for quadrupedal robots. In contrast to advances in locomotion, relatively little research has been done on safely recovering after unexpected falls or planned drops. These abilities can be beneficial for navigating harsh environments and preventing significant damage. High horizontal velocity makes the landing problem much more challenging. There are many situation where this is not negligible, e.g., when a quadruped trots almost at its maximal speed and must do a leap without first stopping the motion. So, for a robot to be able to attains a stable standing posture after a fall, the control framework should estimate and deal with both the vertical and horizontal velocity components. Animals with righting reflexes have inspired many previous works on improving robotic landing capabilities, such as cats [1] and squirrels [2]. These works focus on dorso-ventrally reorienting the main body so that the limbs point toward the ground, e.g., mimicking cats with a flexible spine [3]. Some authors explored the inertial effects of flywheels, tails, and limbs for reorienting the robot when it falls. The work presented in [4] used a flywheel to control the tilt angle effectively; [5] proved that two flywheels with intersecting rotational axes can actuate both roll and pitch, allowing for non-planar jumps and landings. Many researchers have introduced a tail, which is an additional link that rotates about an axis that does not pass through the robot Center of Mass (CoM) [6]–[8]. However, a tail can only perform limited corrections due to its restricted range of motion and it adds weight and complexity to the robot. In [9] Mini Cheetah was provided with heavy boots to improve the influence on torso rotation. Purely vertical falls with large rotations on the sagittal plane are handled by a combination of off-line optimization and supervised machine learning. 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 mass-less legs. There is copious literature on approaches based on Trajectory Optimization (TO) to perform airborne maneuvers. Many of these techniques neglect the landing phase and assume perfect tracking of the optimized trajectory. Minor errors in timing can complicate the execution of the landing task. Using optimized joint torques as feed-forward commands and Proportional Derivative (PD) joint feedback for tracking the optimized joint trajectories, flips were accomplished by Mini Cheetah in [10]. [11] and [12] introduced a framework for performing highly dynamic jumps by combining off-line contact timings, off-line whole-body

The authors are with:
[1] Dynamic Legged Systems (DLS), Istituto Italiano di Tecnologia (IIT), Genoa, Italy,
[2] Department of Information Engineering and Computer Science (DISI), University of Trento, Trento, Italy
[3] Industrial Engineering Department (DII), University of Trento, Trento, Italy,
[4] Advanced Robotics (ADVR), Istituto Italiano di Tecnologia (IIT), Genoa, Italy.
Emails: {name.surname}@{iit, unitn}.it
The publication was created with the co-financing of the European Union FSE-REACT-EU, PON Research and Innovation 2014-2020 DM1062 / 2021.

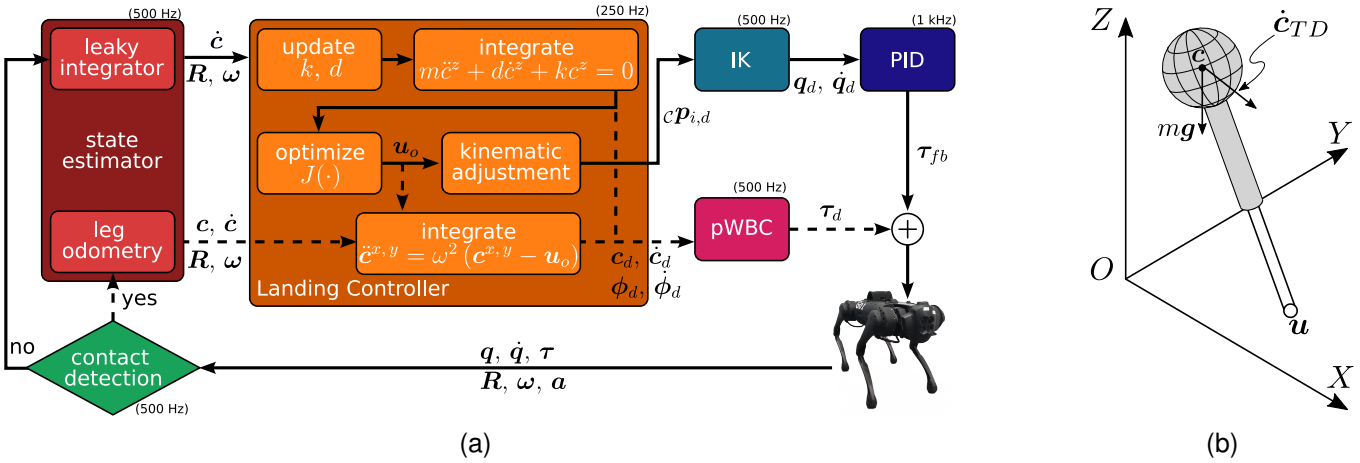


Fig. 2. (a) Overview of the LC framework. When the robot is in the flying phase, the virtual foot location is continuously re-optimized using the template model VHSIP, and the feet are shifted accordingly. Once landing is detected, the last computed trajectory for the CoM and trunk orientation is tracked. The dashed branches are active only after TD is detected. (b) The 3D VHSIP model used to describe the robot dynamics after TD.

TO, and high-frequency tracking controller. Despite the relevant results shown, those approaches may struggle in different occasions. E.g., if the controller does not track exactly the references before the take off, the actual aerial stage may last more/less than planned, resulting in a landing trajectory that is not feasible. Moreover, these methods require to compute in advance the whole references, that depend on the take off conditions. For on-line optimization, such data may not be available if the robot is running and it is requested to jump without stopping. The MIT Mini Cheetah can perform a rich set of aerial movements, as shown in [13]. Motions are planned via centroidal momentum-based nonlinear optimization and then tracked using a variational-based optimal controller [14]. Landing is accomplished using a simple joint PD controller stabilizing a fixed landing position. Therefore, only small heights are considered. Several works based on Reinforcement Learning (RL) and Model Predictive Control (MPC) present interesting landing strategies. 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 [15]. Even if the method is tailored to uneven terrains, tests are conducted on flat and rigid ground. A Vicon system 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. Also [16] addresses the problem of landing on celestial bodies. 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 [17], 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 [18] 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 [19], a complete control framework can select optimal contact locations and timings in real-time. The controller consists of a Neural Network (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 time span represents an extensive flying phase: it corresponds to a height of about 0.5 m, starting with zero velocity. Therefore, falls from reduced heights may not be addressed with this approach.

A. Proposed Approach

This paper 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 touch down as a Variable Height Springy Inverted Pendulum (VHSIP). The use of a spring model is not novel in the literature [20]–[22]. 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.

B. Contributions and Outline

The main contributions of this work are as follows.

- We introduce a real-time *omni-directional* LC framework (Fig. 2a), 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 $3\ \mathrm{m/s}$ dropping the robot from a height of $1\ \mathrm{m}$. 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 advantages and limitations 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 remainder of this manuscript is organized as follows. In Section II, we discuss the VHSIP template model and formally state the landing problem. The structure of the landing framework is detailed in Section III, and the motion control in Section III-G. Finally, we present implementation details and results in Section IV and we draw the conclusions in Section V.

II. 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 Ground Reaction Forces (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.

A. Derivation of the VHSIP Model

Let us introduce an inertial coordinate frame $\mathcal{W}$: the Z–axis is orthogonal to a flat ground and the $XY-$plane lies 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 $\boldsymbol{c} = \begin{bmatrix} c^x & c^y & c^z \end{bmatrix}^T \in \mathbb{R}^3$, having inertia ${}_{\mathcal{C}}\boldsymbol{\mathcal{I}}_{\mathcal{C}} \in \mathbb{R}^{3\times3}$. When the robot is falling, the balance of moments constrains the linear dynamics to follow the ballistic trajectory while conserving the angular momentum $\boldsymbol{L} \in \mathbb{R}^3$. If the robot has n_c contact points with the ground, the balance of linear and angular moments can be written as

$$m\left(\ddot{\boldsymbol{c}} + \boldsymbol{g}\right) = \sum_{i=1}^{n_c} \boldsymbol{f}_i \qquad \dot{\boldsymbol{L}} = \sum_{i=1}^{n_c} \left(\boldsymbol{p}_i - \boldsymbol{c}\right) \times \boldsymbol{f}_i \tag{1}$$

being $\boldsymbol{g} \in \mathbb{R}^3$ the (constant) gravity acceleration vector, and $\boldsymbol{p}_i \in \mathbb{R}^3$ is the position of the i-th contact point on which the environment exerts the force $\boldsymbol{f}_i \in \mathbb{R}^3$. Under the assumptions of horizontal ground ($p_i^z = 0,\ \forall i = 1, \ldots, n_c$) and negligible variation of the angular momentum ($\dot{\boldsymbol{L}} \approx 0$), we obtain the relationship (for further details, see [23])

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

where $\boldsymbol{u} \in \mathbb{R}^2$ is the Center of Pressure (CoP), defined as

$$\boldsymbol{u} \triangleq \frac{\sum_{i=1}^{n_c} f_i^z \boldsymbol{p}_i^{x,y}}{\sum_{i=1}^{n_c} f_i^z}.$$

The dynamics in the horizontal directions of the CoM are decoupled, but they depend on the vertical motion through

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

Let us assume the CoP to be constant, 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 Fig. 2b). If the CoM height is constant, (2) simplifies to the well-known Linear Inverted Pendulum (LIP) model [24]. Nevertheless, such 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:

$$m\ddot{c}^z + d\dot{c}^z + k(c^z - l_0) = F^z, \tag{3}$$

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_{TD}^z and $\dot{c}_{TD}^z$. We will refer to (2)–(3) as the VHSIP model. 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 system is associated with a conserved quantity, the so-called Orbital Energy $E(c^x - u^x,\, \dot{c}^x)$ [25]:

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

with $r^x = c^x - u^x$, 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 to pass over $\boldsymbol{u}$ 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 $\boldsymbol{u}$, [26]. Therefore, robot configurations in equilibrium can be reached by selecting $\boldsymbol{u}$ so that at TD the following condition holds

$$E(c_{TD}^x - u^x,\, \dot{c}_{TD}^x) = 0. \tag{4}$$

In the capturability framework, such $\boldsymbol{u}$ is named Capture Point (CP). Whereas for the LIP model it is possible to explicitly compute it, 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 which 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 u^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.

B. Landing Problem

Now, we can state the problem more formally.

Problem Statement. *Consider a robot that is free falling on 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.*

III. Methodology

In this section, the structure of our LC is introduced and discussed. At any sampling instant, we suppose that the TD is about to occur and use the template VHSIP model to compute a new CoM reference trajectory for landing. In this way, 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 to be able to place the virtual foot $\boldsymbol{u}$ (i.e., the CoP) on the CP at TD. Notice that (2)–(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 Fig. 3). The terrain frame is a horizontal frame [27], 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 III-D) is performed, in order to keep the feet aligned with the ground, 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 of 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.

A. Vertical Motion Reference

In the context of the VHSIP model, the requirement **(R1)** is equivalent to having a critically-damped oscillator in (3). This can be achieved 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

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

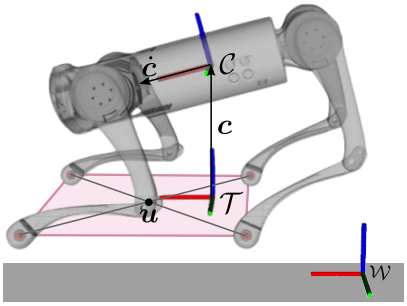


Fig. 3. The CoM frame $\mathcal{C}$ has the origin at the robot CoM and encapsulates the trunk orientation. The terrain frame $\mathcal{T}$ is the horizontal reference frame for which the CoM has coordinates $\boldsymbol{c} = \begin{bmatrix} 0 & 0 & l_0 \end{bmatrix}^T$ when the robot is in air. After 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.

that when the TD occurs, the CoM is always at 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 computed in closed form:

$$\dot{c}^z(t) = e^{\lambda(t-t_{TD})}\ \left(\lambda\left(t-t_{TD}\right)+1\right)\ \dot{c}^z_{TD} \tag{5a}$$

$$c^z(t) = e^{\lambda(t-t_{TD})}\ \left(t-t_{TD}\right)\ \dot{c}^z_{TD} + l_0 \tag{5b}$$

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)**). Having a critically-damped system, this minimum is unique for $t > t_{TD}$. Therefore, denoting with Δ^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:

$$k \geq k_1 = \frac{m}{\left(e\left(\Delta^z - l_0\right)\right)^2}\left(\dot{c}^z_{TD}\right)^2 .$$

We set a maximum for the convergence time to prevent long dynamic evolution when TD velocity is low. Being (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:

$$k \leq k_2 = m\left(t_c/7\right)^2 .$$

At every control instant, we fix the virtual stiffness to $k = \max\{k_1, k_2\}$ and update the virtual damping accordingly. Then, a new CoM reference trajectory is computed.

B. 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 u^x that drives the CoM above it with zero linear velocity when the system reaches steady-state, i.e., that attains zero orbital energy (4). Equation (2) can be rewritten in state-space form:

$$\underbrace{\begin{bmatrix} \dot{c}^x(t) \\ \ddot{c}^x(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} 0 & 1 \\ \omega^2(t) & 0 \end{bmatrix}}_{\boldsymbol{A}(t)} \underbrace{\begin{bmatrix} c^x(t) \\ \dot{c}^x(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} 0 \\ -\omega^2(t) \end{bmatrix}}_{\boldsymbol{B}(t)} u^x \tag{6}$$

showing its nature of Linear Time Varying (LTV) system, with $\omega(t)$ evolving in time. Even if the dynamics is linear, its

integration does not admit a closed-form solution because the matrix $\boldsymbol{A}(t)$ does not commute with its integral $\int_{t_{TD}}^{t} \boldsymbol{A}(\sigma)d\sigma$, [28]. Then, we set up an Optimal Control Problem (OCP) to find the optimal CoP 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 as 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

$$\begin{aligned} \min_{\mathbf{x}_k,\ u^x}\ & w_p|\boldsymbol{C}_p\mathbf{x}_N - u^x|^2 + w_v|\boldsymbol{C}_v\mathbf{x}_N|^2 + w_u|u^x|^2 \\ \text{s.t.}\quad & \mathbf{x}_{k+1} = \bar{\boldsymbol{A}}_k\mathbf{x}_k + \bar{\boldsymbol{B}}_k u^x \qquad k = 0,\, 1,\, \ldots,\, N-1 \end{aligned} \tag{7}$$

having set $\bar{\boldsymbol{A}}_k = \boldsymbol{I}_{2\times 2} + T_s\boldsymbol{A}_k$ and $\bar{\boldsymbol{B}}_k = T_s\boldsymbol{B}_k$. $\boldsymbol{I}_{2\times 2}$ is the 2×2 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 at time NT_s the system is at steady-state. We solve (7) with a single shooting approach. For doing so, the knowledge of $\omega^2(t)$, $0 \le t \le NT_s$, is needed. Thus, (3) must be forward integrated up to NT_s. Recursively applying (2), we express the state trajectory as a function of the virtual foot and the initial conditions. This results in an unconstrained Quadratic Program (QP) in the scalar optimization variable u^x:

$$\min_{u^x}\ J(\mathbf{x}_0, u^x) \triangleq \mathbf{x}_0^T\boldsymbol{Q}^x\mathbf{x}_0 + u^{xT}\boldsymbol{Q}^u u^x + 2\mathbf{x}_0^T\boldsymbol{Q}^{xu}u^x \tag{8}$$

$\boldsymbol{Q}^x$, $\boldsymbol{Q}^u$ and $\boldsymbol{Q}^{xu}$ are real matrices of dimensions 2×2, 1×1 and 2×1, respectively. For the sake of conciseness, we do not report their expression here. 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 (2) and forward integrating along the horizon to get the horizontal components, while keeping the vertical component previously computed as described in (3). Conversely, if a TD does not occur, the kinematic adjustment described in Section III-D is performed, shifting the feet according to the optimal virtual foot location. Additionally, the terrain frame is updated and a new reference is computed.

C. 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. 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 (3) attains the steady state:

$$\boldsymbol{\phi}_d(t) = e^{\lambda(t-t_{TD})}\ (t - t_{TD})\ \dot{\boldsymbol{\phi}}_{TD}. \tag{9}$$

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

D. Kinematic Adjustment

During the flying phase, the robot limbs must prepare to realize the optimal CoP at TD. We call such motion *kinematic adjustment* [27]. Fig. 3 shows the robot while performing the kinematic adjustment. It consists in 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 optimal CoP $\boldsymbol{u}_o$:

$$\boldsymbol{p}_{i,\,d}(t) = \boldsymbol{p}_{i,\,0} + \alpha(t)\,\boldsymbol{u}_o, \tag{10}$$

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 ${}_{\mathcal{C}}\boldsymbol{p}_{i,\,d} = {}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}}\big(\boldsymbol{p}_{i,\,d} - \begin{bmatrix} 0 & 0 & l_0 \end{bmatrix}^T\big)$. Feet motions can be realized with joint-space control. Desired joint configurations can be computed solving an Inverse Kinematic (IK) problem for each foot assuming mass-less legs: $\boldsymbol{q}_{i,\,d} = \mathrm{IK}({}_{\mathcal{C}}\boldsymbol{p}_{i,\,d})$, where $\boldsymbol{q}_{i,\,d}$ contains the desired angles of the joints on the $i-$th leg. The mechanical structure of our robot allows us to employ closed form IK at the position level and $\boldsymbol{q}_{i,\,d} \in \mathbb{R}^3$. Desired joint velocities are computed considering the Cartesian-space error ${}_{\mathcal{C}}\boldsymbol{e}_i = {}_{\mathcal{C}}\boldsymbol{p}_{i,\,d} - {}_{\mathcal{C}}\boldsymbol{p}_i$:

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

being $\boldsymbol{J}_i\,(\boldsymbol{q}_d) \in \mathbb{R}^{3\times 3}$ the Jacobian matrix associated to the i-th foot, $\boldsymbol{I}_{3\times 3}$ is the 3×3 identity matrix, $\varepsilon \in \mathbb{R}^+$ a regularization parameter, and $k_v \in \mathbb{R}^+$ a scaling factor. The joint trajectory is tracked with a joint-space PD controller

$$\boldsymbol{\tau}_{fb} = \boldsymbol{K}_P^j\,(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K}_D^j\,(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{\tau}_g\,(\boldsymbol{q})\,. \tag{11}$$

with the gravity compensation $\boldsymbol{\tau}_g\,(\boldsymbol{q})$ 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.

E. Touch Down Detection

We define the transition between flying and 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 Ground Reaction Forces (GRFs) $\hat{\boldsymbol{f}}_i$ exerted on it:

$$\hat{\boldsymbol{f}}_i = \boldsymbol{J}_i(\boldsymbol{q})^{-T}\boldsymbol{S}_i\big(\boldsymbol{C}(\boldsymbol{q})\,\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{\tau}\big) \tag{12}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\boldsymbol{\tau}$ are the measured joint positions, velocities and torques, respectively, $\boldsymbol{C}(\boldsymbol{q})\,\dot{\boldsymbol{q}}$ is the centrifugal and Coriolis term, $\boldsymbol{g}(\boldsymbol{q})$ is the vector of gravitational effects, $\boldsymbol{S}_i$ is the matrix picking out torques associated to the i-th leg. Because of the flat horizontal ground assumption, a contact is detected if $\hat{f}_i^z \ge f_{th}^z$, with f_{th}^z a (small) robot-dependent threshold.

F. 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 base frame). The linear acceleration in the inertial frame is retrieved considering the accelerometer bias ${}_{\mathcal{C}}\boldsymbol{b} \in \mathbb{R}^3$ as: $\boldsymbol{a} = {}_{\mathcal{W}}\boldsymbol{R}_{\mathcal{C}} \left({}_{\mathcal{C}}\boldsymbol{a} - {}_{\mathcal{C}}\, \boldsymbol{b} \right) - \boldsymbol{g}$. During the flying phase, we reconstruct the base linear velocity $\boldsymbol{v} \in \mathbb{R}^3$ making use of a leaky integrator, equally to [29]:

$$\hat{\boldsymbol{v}}_{k+1} = \left(\boldsymbol{I}_{3\times 3} - \boldsymbol{\Gamma} T_s \right) \hat{\boldsymbol{v}}_k + \boldsymbol{I}_{3\times 3}\, T_s \boldsymbol{a}_k \tag{13}$$

where $\boldsymbol{\Gamma} = \operatorname{diag}\left(\gamma^x, \gamma^y, \gamma^z\right)$ 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 [30].

G. Motion Control During Landing Phase

In this section, we discuss the projection-based Whole Body Control (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 [30].

1) Feedback Wrench: A PD feedback term is computed to track the CoM reference and the base orientation:

$$\begin{aligned} \boldsymbol{w}_{fb}^{lin} &= \boldsymbol{K}_P^{lin} \left(\boldsymbol{c}_d - \boldsymbol{c} \right) + \boldsymbol{K}_D^{lin} \left(\dot{\boldsymbol{c}}_d - \dot{\boldsymbol{c}} \right), \\ \boldsymbol{w}_{fb}^{ang} &= \boldsymbol{K}_P^{ang}\, \boldsymbol{e} \left({}_{\mathcal{W}}\boldsymbol{R}_{\mathcal{C},\,d}\ {}_{\mathcal{W}}\boldsymbol{R}_{\mathcal{C}}{}^T \right) + \boldsymbol{K}_D^{ang} \left(\boldsymbol{\omega}_d - \boldsymbol{\omega} \right) \end{aligned}$$

where ${}_{\mathcal{W}}\boldsymbol{R}_{\mathcal{C},\,d}$ is the rotation matrix associated to $\boldsymbol{\phi}_d$, $\boldsymbol{e} : SO(3) \to \mathbb{R}^3$ maps a rotation matrix to the associated rotation vector and $\boldsymbol{\omega}$ is the actual angular velocity. The desired angular velocity are obtained with $\boldsymbol{\omega}_d = \boldsymbol{T}\left(\boldsymbol{\phi}_d\right) \dot{\boldsymbol{\phi}}_d$.

2) Feed-forward Wrench: We annihilate the effects of the gravity force on the CoM through the feed-forward component

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

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

$$\boldsymbol{w}_{ff}^{lin} = m\ddot{\boldsymbol{c}}_d, \qquad \boldsymbol{w}_{ff}^{ang} = {}_{\mathcal{T}}\boldsymbol{R}_{\mathcal{C}}\ {}_{\mathcal{C}}\boldsymbol{\mathcal{I}}_{\mathcal{C}}\ {}_{\mathcal{C}}\boldsymbol{R}_{\mathcal{T}} \dot{\boldsymbol{\omega}}_d.$$

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

3) 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

$$\begin{aligned} \boldsymbol{f}_d = \underset{\boldsymbol{f}_d}{\operatorname{argmin}}\ & \frac{1}{2} \| \boldsymbol{G}\boldsymbol{f}_d - \boldsymbol{w}_d \|^2 \\ \text{s.t.}\quad & |f_{i,\,d}^{x,\,y}| \le \mu f_{i,\,d}^z, \quad f_{i,\,d}^z > 0 \quad i = 1, \ldots, n_c \end{aligned} \tag{14}$$

that enforces **(R4)** by means of the linearized friction cone constraints. 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

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

where $\left[\,\cdot\,\right]_\times$ is the skew-symmetric operator associated to the vector product. Finally, the desired torque to be exerted by the joints of the i-th leg is $\boldsymbol{\tau}_{i,\,d} = \boldsymbol{S}_i\, \boldsymbol{C}(\boldsymbol{q})\, \dot{\boldsymbol{q}} - \boldsymbol{J}_i\left(\boldsymbol{q}\right) \boldsymbol{f}_{i,\,d}$.

IV. Results

A. Implementation details

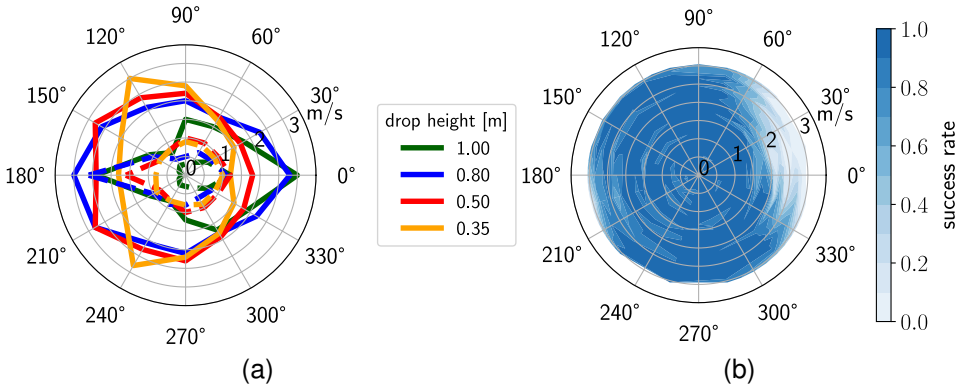
The robot we use to demonstrate the validity of our approach is the torque-controlled quadruped Unitree Go1 Edu [31]. To visualize, simulate and interact with the robot, we use Locosim [32], a platform-independent software framework designed for fast code prototyping[3]. We implemented both the pWBC and the LC in a Python ROS node, which relies on Pinocchio [33] 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. To increase computational efficiency, we coded the VHSIP dynamics in C++ 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 accompanying video available at youtu.be/wiuedeHfSEY. For all the cases, we set the parameters of Sec. III-A as follow: the nominal height $l_0 = 0.27$ m (robot height in home configuration), the clearance $\Delta^z = 0.10$ m, the maximum settling time $t_c = 1.2$ s.

B. 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 it is present in our LC. 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 achieve a successful landing. By executing thousands of automated drops with varying the initial conditions of (2)-(3), we detected 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 Fig. 4a. We tested dropping horizontal velocities in the form $\dot{\boldsymbol{c}}_0^{x,\,y} = \nu \begin{bmatrix} \cos\psi & \sin\psi \end{bmatrix}^T$, with $\nu = 0.0,\, 0.1,\, \ldots,\, 4.0\,\mathrm{m/s}$

[3]The code to replicate the results is open source and can be downloaded at github.com/iit-DLSLab/reactive_landing_controller/.



(a) (b)

	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]

(c)

Fig. 4. Simulation results. (a) 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., $\mathrm{atan2}\left(\dot{c}_0^{y},\, \dot{c}_0^{x}\right)$. (b) 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. (c) Angular perturbations limits dropping Go1 from 0.6 m height and with 1.0 m/s forward velocity. The asymmetry of the robot inertia causes the roll angle and rate bounds to be not symmetric.

and $\psi = 0, \pi/6, \ldots, 2\pi$ rad. For all the tested dropping heights, 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 for 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.

C. 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. Within each batch, we compute the success rate as the percentage of achieved landing tasks, see the colour map in Fig. 4b. Despite the noise, the overall success rate is 0.875.

D. Angular perturbations

Here, we want to evaluate how tolerant our method is to angular perturbations, despite the fact that our template model does not describe the angular motion (see Section III-C). In simulation, 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 4c.

E. Experiments

We performed an extensive experimental study to assess the performance of our LC. All the tests are visible in the accompanying video. We dropped the 12 kg Go1 robot from various heights with different manually induced horizontal velocities in all the directions (forward, backwards, left, right). Fig. 1 shows a drop from about 0.8 m with non-zero horizontal velocity in the forward direction. Plots in Fig. 5 are associated to this experiment. 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 in all the tests joint velocities and torques never reach their limits.

V. Conclusions

In this work, we presented a model-based approach for quadruped robot landing after unexpected falls or planned drops. A successful landing entails dissipating all the kinetic energy, stopping the robot without hitting the ground with the trunk and keeping the feet in contact. Tracking an impedance model turned out to be a suitable candidate to dissipate the excess of kinetic energy avoiding rebounds. The approach is reactive enough (500 Hz) to cope with heights lower than 0.5 m as in [19] and it makes use only of proprioceptive measurements, being therefore independent of an external motion capture system. Furthermore, it can achieve *omni-directional* landing with significant horizontal velocity, up to 3.0 m/s. Our landing controller was extensively bench-marked in simulation and demonstrated to dramatically outperform a naive landing strategy that tracks the vertical impedance but does not adjust the feet locations during the fall. Despite employing a simplified model that assumes horizontal orientation, the approach was demonstrated to tolerate a large range of orientation errors. An extensive experimental evaluation of omni-directional landing on the real robot Go1 was also presented, randomly dropping the robot in different ways.

Future research directions could increase the model descriptiveness, relaxing the assumption of negligible variation of the angular momentum, extending the region of feasible horizontal velocities and tolerable angular perturbations. We also want to introduce a backup plan for situations in which a large horizontal velocity causes the optimal CoP to lie outside of either the kinematic region or the convex hull of the feet.

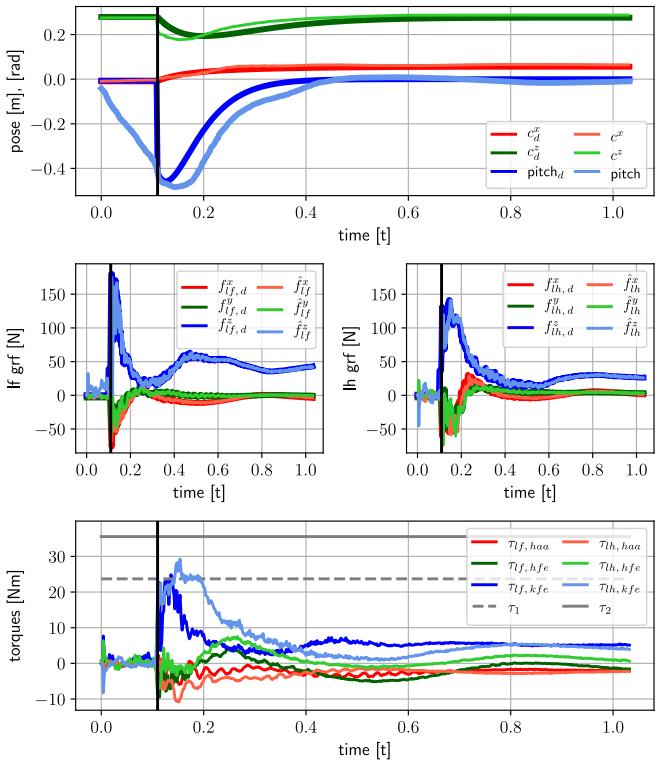


Fig. 5. Experimental results. Desired and actual CoM and pitch trajectories expressed in $\mathcal{T}$-frame (top); desired and estimated GRFs exerted on the left front foot (left center) and on the left hind foot (right center); measured torques for the left legs (bottom). The upper actuation limits for both hip abduction/adduction (haa) and hip flexion/extension (hfe) mechanisms is τ_1, for knee flexion/extension (kfe) one is τ_2. Lower limits are mirrored. The vertical lines denote the detection of the TD.

Finally, we want to extend the approach for landing onto non-horizontal surfaces and soft terrains.

References

[1] T. R. Kane and M. Scher, "A dynamical explanation of the falling cat phenomenon," *International journal of solids and structures*, vol. 5, no. 7, pp. 663–670, 1969.

[2] N. H. Hunt, J. Jinn, L. F. Jacobs, and R. J. Full, "Acrobatic squirrels learn to leap and land on tree branches without falling," *Science*, vol. 373, no. 6555, pp. 697–700, 2021.

[3] B. Shields, W. S. Robertson, N. Redmond, R. Jobson, R. Visser, Z. Prime, and B. Cazzolato, "Falling cat robot lands on its feet," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2013, pp. 74–82.

[4] H. Kolvenbach, E. Hampp, P. Barton, R. Zenkl, and M. Hutter, "Towards jumping locomotion for quadruped robots on the moon," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5459–5466.

[5] F. Roscia, A. Cumerlotti, A. Del Prete, C. Semini, and M. Focchi, "Orientation control system: Enhancing aerial maneuvers for quadruped robots," *Sensors*, vol. 23, no. 3, 2023.

[6] G. Wenger, A. De, and D. E. Koditschek, "Frontal plane stabilization and hopping with a 2dof tail," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 567–573.

[7] X. Chu, C. H. D. Lo, C. Ma, and K. W. S. Au, "Null-space-avoidance-based orientation control framework for underactuated, tail-inspired robotic systems in flight phase," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3916–3923, 2019.

[8] Y. Tang, J. An, X. Chu, S. Wang, C. Y. Wong, and K. S. Au, "Towards safe landing of falling quadruped robots using a 3-dof morphable inertial tail," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1141–1147.

[9] V. Kurtz, H. Li, P. M. Wensing, and H. Lin, "Mini cheetah, the falling cat: A case study in machine learning and trajectory optimization for robot acrobatics," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4635–4641.

[10] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6295–6301.

[11] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim, "Optimized jumping on the mit cheetah 3 robot," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7448–7454.

[12] C. Nguyen and Q. Nguyen, "Contact-timing and trajectory optimization for 3d jumping on quadruped robots," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 994–11 999.

[13] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7693–7699.

[14] M. Chignoli and P. M. Wensing, "Variational-based optimal control of underactuated balancing for dynamic quadrupeds," *IEEE Access*, vol. 8, pp. 49 785–49 797, 2020.

[15] N. Rudin, H. Kolvenbach, V. Tsounis, and M. Hutter, "Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning," *IEEE Transactions on Robotics*, vol. 38, pp. 317–328, 2021.

[16] J. Qi, H. Gao, H. Yu, M. Huo, W. Feng, and Z. Deng, "Integrated attitude and landing control for quadruped robots in asteroid landing mission scenarios using reinforcement learning," *Acta Astronautica*, 2022.

[17] G. Bledt, "Regularized predictive control framework for robust dynamic legged locomotion," Ph.D. dissertation, Massachusetts Institute of Technology, 2020.

[18] Y. L. Jingwen Zhang, Junjie Shen and D. Hong, "Design of a jumping control framework with heuristic landing for bipedal robots," *https://arxiv.org/pdf/2304.00536.pdf*, 2023.

[19] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the mit mini cheetah," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 178–184.

[20] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[21] R. Blickhan, "The spring-mass model for running and hopping," *Journal of biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.

[22] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3821–3828.

[23] P.-B. Wieber, R. Tedrake, and S. Kuindersma, "Modeling and control of legged robots," in *Springer handbook of robotics*. Springer, 2016, pp. 1203–1234.

[24] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 1991, pp. 1405–1406.

[25] J. E. Pratt and S. V. Drakunov, "Derivation and application of a conserved orbital energy for the inverted pendulum bipedal walking model," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2007, pp. 4653–4660.

[26] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*. IEEE, 2006, pp. 200–207.

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

[28] C.-T. Chen, *Linear system theory and design*, 3rd ed. Oxford University Press, 1999, ch. 4, pp. 106 – 117.

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

[30] M. Focchi, A. Del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, pp. 259–272, 2017.

[31] Unitree. Go1 - accompany you to the world. [Online]. Available: https://www.unitree.com/en/go1/

[32] M. Focchi, F. Roscia, and C. Semini, "Locosim: an open-source cross-platform robotics framework," *arXiv preprint arXiv:2305.02107v2*, 2023.

[33] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. 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)*. IEEE, 2019, pp. 614–619.