# On-the-Fly Jumping with Soft Landing: Leveraging Trajectory Optimization and Behavior Cloning

Edoardo Panichi†, Jiatao Ding†⋆, Vassil Atanassov, Peiyu Yang, Jens Kober, Wei Pan and Cosimo Della Santina

***Abstract*—Quadrupedal jumping has been intensively investigated in recent years. Still, realizing controlled jumping with soft landings remains an open challenge due to the complexity of the jump dynamics and the need to perform complex computations during the short time. This work tackles this challenge by leveraging trajectory optimization and behavior cloning. We generate an optimal jumping motion by utilizing dual-layered coarse-to-refine trajectory optimization. We combine this with a variable impedance control approach to achieve soft landing. Finally, we distill this computationally heavy jumping and landing policy into an efficient neural network via behavior cloning. Extensive simulation experiments demonstrate that, compared to classic model predictive control, the variable impedance control ensures compliance and reduces the stress on the motors during the landing phase. Furthermore, the neural network can reproduce jumping and landing behavior, achieving at least a 97.4% success rate. Hardware experiments confirm the findings, showcasing explosive jumping with soft landings and on-the-fly evaluation of the control actions.**

***Index Terms*—compliant control, behavior cloning, neural network, trajectory optimization, and quadrupedal robots.**

## I. Introduction

Quadrupedal robots excel at navigating complex terrain, making them invaluable for exploring uncharted areas, such as challenging stairs, rocky terrain, and confined spaces. Among various locomotion modalities, quadrupedal jumping enhances mobility and adaptability [1], [2]. In particular, achieving a soft landing after touch-down reduces mechanical stress, extending robotic durability [3]. We aim to address this challenge, especially for the quadruped without precise contact force sensors [4] or without compliant mechanical design [5].

To address the computationally intense jumping planning and control task, the pipeline is often split into two stages: a motion planning stage and a motion tracking stage. A prevalent approach in the planning phase is trajectory optimization (TO), aiming for the optimal trajectory passing through a set of waypoints. This method has enabled various achievements, including jumping through window-shaped obstacles [6], robust jumping [7], and continuous jumping [8]. To achieve online optimization, [9] ignored joint dynamics and kinematics during the stance phase, assuming that the feet are located within a confined workspace that does not violate the limits of kinematics. Alternatively, [10] exploited analytical solutions for fast computation. After obtaining a reference, model-based controllers such as virtual model control [11], model predictive control (MPC) [12], and whole-body control [13] can be integrated for motion tracking. However, none of the above work emphasizes soft landing.

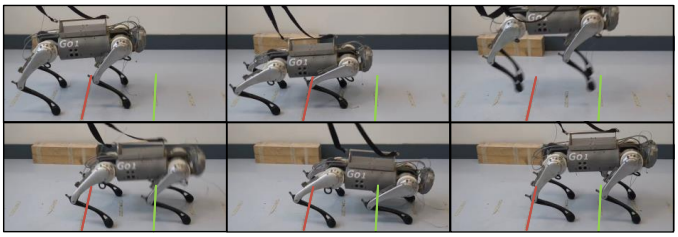

Fig. 1. The Go1 jumps with a soft landing, using the learned policy from the behavior cloning. The distance between the red and green lines is 40cm.

A soft landing is characterized by two main features: compliance with movement and reduction in motor stress. In this context, [14] developed an optimal landing controller to regulate touchdown postures and force profiles. Although impressive, this work focused on falling rather than jumping. [15] solved the problem of landing control with aggressive horizontal velocities, which, however, is limited by the assumption of landing on flat ground. [16] utilized an online TO to generate the Cartesian space landing motion but ignored the feasibility of the joint movement. [17] adopted virtue force control for soft landing, however, without considering constraints.

In contrast to model-based approaches, model-free strategies, such as reinforcement learning (RL), enable learning control policies through data sampling. In particular, deep RL has facilitated significant advancements in performing jumping tasks [18], [19], among which the imitation learning (IL) framework has been well investigated [20]–[22]. However, the RL framework requires careful design and tuning of the reward function and needs to bridge the sim2real gap. In contrast, behavior cloning (BC), a variant of IL [23], allows the agent to learn an effective policy matching the behavior of the expert while avoiding the computational overhead of TO. Although impressive results have been achieved recently [24], [25], very limited trials in quadrupedal jumping are reported. Kurtz et al. [26] appears to be the closest study, where a synthetic dataset was used to train a model on robot reorientation and landing during falls. However, comprehensive jumping with soft landing is not investigated.

Edoardo Panichi, Jiatao Ding, Peiyu Yang, Jens Kober and Cosimo Della Santina are with the Department of Cognitive Robotics, Delft University of Technology, Building 34, Mekelweg 2, 2628 CD Delft, Netherlands (e-mail:{edoardo.panichi99@gmail.com, J.Ding-2@tudelft.nl, p.yang-5@student.tudelft.nl, J.Kober@tudelft.nl, C.DellaSantina@tudelft.nl}). Vassil Atanassov is with the Oxford Robotics Institute, Department of Engineering Science, University of Oxford, UK (email: vassilatanassov@robots.ox.ac.uk). Wei Pan is with the Department of Computer Science, The University of Manchester, UK (wei.pan@manchester.ac.uk). Cosimo Della Santina is also with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany (e-mail: cosimodellasantina@gmail.com).

† These authors contribute equally.

⋆ Corresponding author.

In this work, we exploit the best of both fields, i.e., TO and BC, to achieve quadrupedal jumping with a soft landing. In particular, our solution involves a deep supervised learning framework that replaces the model-based planner and controller, enabling on-the-fly execution. First, we utilize model-based TO to generate optimal reference conditioned by jumping goals. Then, integrated with a variable impedance controller (VIC) for compliant landing, we generate a synthetic dataset of $11,000$ jumps with soft landings. This dataset is then used to train a neural network (NN), achieving performance comparable to the model-based method, but releasing the computational burden.

The main contributions are as follows:

- We formulate a dual-layer TO for jumping motion generation, leveraging the actuated spring-loaded inverted pendulum (aSLIP) dynamics and the single rigid body (SRB) dynamics.
- We develop a compliant controller to minimize landing impact and motor efforts after touch-down.
- We propose a BC scheme for on-the-fly control. Experiments demonstrate that the trained NN successfully replicates the soft landing behaviors and bypasses computational inefficiencies associated with the planner.

The rest of this article is organized as follows. In Section II, we state the problem formulation. Section III details the optimization-based jumping and landing control strategy. Section IV explores the supervised learning-based BC approach. Section V presents the experimental results, and Section VI concludes this work and discuss the future work.

## II. Problem Statement

The primary objective is to perform a quadrupedal jump with a soft landing. In particular, we aim for on-the-fly execution without necessitating extensive computation.

For a jumping motion, the full state of the robot is

$$\boldsymbol{X}^{+} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_r, \omega_p, \omega_y, \boldsymbol{q}^{\mathrm{T}}, \dot{\boldsymbol{q}}^{\mathrm{T}}], \quad (1)$$

where $[x, y, z]$ represent the 3D center of mass (CoM) position, while $[\phi, \theta, \psi]$ denote roll, pitch, and yaw (RPY) angles. $[\dot{x}, \dot{y}, \dot{z}]$ and $[\omega_r, \omega_p, \omega_p]$ represent the linear velocity and the angular velocity. Joint positions and velocities are indicated as $\boldsymbol{q} \in \mathbb{R}^{12}$ and $\dot{\boldsymbol{q}} \in \mathbb{R}^{12}$, respectively.

In this work, we focus on the sagittal jump. Given the desired jumping distance $d$, the state of the robot at any given moment is then encapsulated by

$$\boldsymbol{X} = [z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_r, \omega_p, \omega_y, \boldsymbol{q}^{\mathrm{T}}, \dot{\boldsymbol{q}}^{\mathrm{T}}, d]. \quad (2)$$

Denoting the control input $\mathbf{u}$, the problem we seek to solve is then defined as:

**Problem:** *Starting from an initial state denoted by $\boldsymbol{X}_0$, identify the control action sequence $\mathbf{u}$ that enables the robot to achieve a jump of the desired length $d$. This sequence should be optimized to minimize motor effort and ensure smooth deceleration during the landing phase for a soft landing. Moreover, real-time computation of $\mathbf{u}$ must be ensured.*

*Notations:* Matrices and vectors are separately highlighted in bold normal font and bold italic font. The superscript $(\cdot)^{\mathrm{T}}$ represents the transpose operation. For the matrix with multiple rows and columns, the index $(\cdot)(k)$ means the $k$-th column and the subscript $(\cdot)_{(i,j)}$ notes the element in the $i$-th row and $j$-th column. For the vector, $(\cdot)(k)$ refers to the $k$-th element. Variables accompanied by $(\cdot)^r$ denote the reference values. Besides, variables with the superscript $(\cdot)^{\max}$ and $(\cdot)^{\min}$ separately denote the upper and lower boundaries.

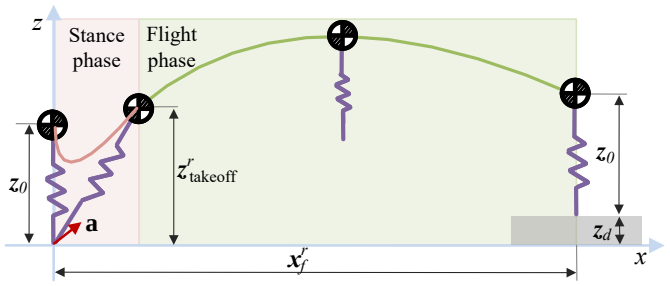


Fig. 2. An example jumping trajectory generated with the aSLIP model. The red zone and green zone separately mark the stance and the flight phase.

## III. Jumping Planning and Control

This section details the model-based jumping motion planning and control pipeline. In particular,for motion planning, dual-layer TO is proposed. For the soft landing, we resort to the variable impedance control.

### A. Jumping Motion Generation: Coarse-to-refine TO

As illustrated in Fig. 2, we divide the jumping motion into a stance phase and a flight phase, with all and none of the feet in contact, respectively. Assuming $N_s$ knots for stance (each knot lasts $t_s$) and $N_f$ knots for flight (each knot lasts $t_f$), we generate the jumping trajectory over $N$ knots ($N = N_s + N_f$).

*1) First-layer: TO with aSLIP dynamics:* Given the desired landing position, the first layer quickly generates a raw jumping trajectory, providing an initial guess for the second layer. Modeling the quadruped as an aSLIP [16], we solve

$$\begin{aligned}
&\arg\min_{\mathbf{c},\mathbf{a},\boldsymbol{dt}} \quad J_{\text{cost}}, && \text{(3a)}\\
&\text{s.t.} \quad \textbf{Kinematic constraints:}\\
&\quad \mathbf{c}(0) = \boldsymbol{c}_0, \quad \dot{\mathbf{c}}(0) = \mathbf{0}, && \text{(3b)}\\
&\quad \mathbf{c}_{(1,N-1)} = z^r_f, && \text{(3c)}\\
&\quad \mathbf{c}_{(1,N_s-1)} = z^r_{\text{takeoff}}, && \text{(3d)}\\
&\quad \boldsymbol{c}^{\min} \le \mathbf{c}(k) \le \boldsymbol{c}^{\max}, \quad \forall k \le N_s, && \text{(3e)}\\
&\quad \boldsymbol{dt}^{\min} \le \boldsymbol{dt} \le \boldsymbol{dt}^{\max}, && \text{(3f)}\\
&\quad \textbf{Dynamics constraints:}\\
&\quad \forall k \in [0, 1, \ldots, N-1]:\\
&\quad \mathbf{c}(k+1) = \text{Taylor}(\mathbf{c}(k), \dot{\mathbf{c}}(k), \ddot{\mathbf{c}}(k)), && \text{(3g)}\\
&\quad \forall k \le N_s:\\
&\quad \ddot{\mathbf{c}}(k) = \mathbf{F}_s(\mathbf{c}(k))/m + \boldsymbol{g} + \mathbf{a}(k), && \text{(3h)}\\
&\quad \mathbf{a}_{(1,k)} + \boldsymbol{F}_{s,z}(\mathbf{c}(k))/m \ge 0, && \text{(3i)}\\
&\quad |\ddot{\mathbf{c}}_{(0,k)}/(\ddot{\mathbf{c}}_{(1,k)} + g)| \le \mu, && \text{(3j)}\\
&\quad \forall k > N_s:\\
&\quad \ddot{\mathbf{c}}(k) = \boldsymbol{g}, && \text{(3k)}
\end{aligned}$$

In the above formulation, $\mathbf{c} \in \mathbb{R}^{2\times N}$ denote the sagittal CoM position where the first and second row separately denote the forward and vertical position. $\mathbf{a} \in \mathbb{R}^{2\times N}$ denote the sagittal acceleration resulting from the actuation force, and

$\boldsymbol{dt} \in \mathbb{R}^2$ denote the time steps for the stance and flight phase. $\mathbf{F}_s(\mathbf{c}(k)) \in \mathbb{R}^2$ is spring force (regarding the current body position) and $\boldsymbol{F}_{s,z}(\mathbf{c}(k))$ is the vertical component. $m$ is the total mass and $\boldsymbol{g} = [0, g]^{\mathrm{T}}$ is the gravitational acceleration.

*Cost function:* The cost function in (3a) penalizes control inputs (i.e., $\mathbf{a}$) and CoM jerk ( $\dddot{\mathbf{c}}$) during stance, penalizes the jumping height during the flight and penalizes the tracking error of landing position. In short, it is defined as

$$J_{\text{cost}} = \sum_{k=1}^{N_s} (\beta_a \|\mathbf{a}(k)\|_2 + \beta_{\text{jerk}} \|\dddot{\mathbf{c}}(k)\|_2) + \sum_{k=N_s}^{N} \beta_f \|\boldsymbol{c}_z(k) - z^r_{\text{takeoff}}\|_2 + \beta_x \|\boldsymbol{c}_x(N-1) - x^r_f\|_2, \tag{4}$$

with $x^r_f$ and $z^r_{\text{takeoff}}$ being the desired landing distance and take-off height, $\beta_a$, $\beta_{\text{jerk}}$, $\beta_f$ and $\beta_x$ being the coefficients.

*Constraints:* The kinematic constraints in (3b) ensure that the trajectory starts from a particular initial state (with $\boldsymbol{c}_0 \in \mathbb{R}^2$ being the initial CoM position). (3c) ensures that the robot lands at the desired height $z^r_f$ ($z^r_f = z_0 + z_d$ as described in Fig. 2). The constraint in (3d) regulates the height of the take-off ($z^r_{\text{takeoff}}$) so that the robot can jump. Furthermore, (3e) obeys the kinematic reachability. (3f) limits the timestep.

The dynamics constraints ensure that the robot follows the aSLIP dynamics (with (3h) in stance and (3k) in flight). (3g) is realized through second-order Taylor integration. Furthermore, the constraint in (3i) avoids free fall, and (3j) ensures that there is no slippage in stance with $\mu$ being the friction coefficient.

*2) SRB-based kino-dynamics optimization:* Using the first layer as the reference, we then refine the motion with kino-dynamics optimization, leveraging the SRB dynamics[1], following

$$\begin{aligned}
\arg\min_{\mathbf{X}^+, \mathbf{F}, \mathbf{r}} \quad & \|\tilde{\mathbf{X}} - \tilde{\mathbf{X}}^r\|^2_{\mathbf{Q}}, && \text{(5a)}\\
\text{s.t.} \quad & \textbf{Kinematic constraints:} \\
& \mathbf{X}^+(0) = \boldsymbol{X}^+_0, \qquad \mathbf{r}(0) = \boldsymbol{r}_0, && \text{(5b)}\\
& \boldsymbol{c}_d - \boldsymbol{\epsilon} \le \mathbf{X}_{(0:2,N-1)} \le \boldsymbol{c}_d + \boldsymbol{\epsilon}, && \text{(5c)}\\
& \forall k \in [1, 2, \dots, N]: \\
& \quad \mathbf{r}^i(k) = \mathrm{FK}(\mathbf{q}^i(k)), \quad i \in \{1,2,3,4\} && \text{(5d)}\\
& \quad \|\mathbf{h}^i(k) - \mathbf{r}^i(k)\| \le L^{\max}_{\text{leg}}, && \text{(5e)}\\
& \quad \mathbf{X}^+(k) \in \mathcal{B}, && \text{(5f)}\\
& \textbf{Dynamics constraints:} \\
& \forall k \le N-1: \\
& \quad \dot{\mathbf{X}}^+(k+1) = f(\mathbf{X}^+(k), \mathbf{r}(k), \mathbf{F}(k)), && \text{(5g)}\\
& \forall k \le N_s: \\
& \quad CCC(\mathbf{F}(k), \mathbf{r}(k)), && \text{(5h)}\\
& \quad |[\mathbf{J}(\mathbf{q}^i(k))]^{\mathrm{T}} \mathbf{F}^i(k)| \le \tau_{\max}, && \text{(5i)}\\
& \quad -\mu \mathbf{F}^i_{(2,k)} \le \mathbf{F}^i_{(0,k)} \le \mu \mathbf{F}^i_{(2,k)}, && \text{(5j)}\\
& \quad -\mu \mathbf{F}^i_{(2,k)} \le \mathbf{F}^i_{(1,k)} \le \mu \mathbf{F}^i_{(2,k)}. && \text{(5k)}
\end{aligned}$$

With the above formulation, we optimize the full state (defined in (1)) with $\mathbf{X}^+ \in \mathbb{R}^{36\times N}$, contact force ($\mathbf{F} \in \mathbb{R}^{12\times N}$) and foot location ($\mathbf{r} \in \mathbb{R}^{12\times N}$). $\mathbf{F}^i$, $\mathbf{h}^i$, $\mathbf{q}^i$ and $\mathbf{r}^i$ ($\in \mathbb{R}^{3\times N}$) are the contact force, hip position, joint angle and foot location of the $i$-th leg ($i \in \{1,2,3,4\}$).

[1]The formulation draws inspiration from the work in [14]. However, while [14] mainly addresses falls, we here define the TO for explosive jumping.

*Cost functions:* (5a) penalizes the tracking errors of the reference CoM motion. $\tilde{\mathbf{X}} \in \mathbb{R}^{6\times N}$ contains the first six states of $\mathbf{X}^+$ and $\tilde{\mathbf{X}}^r \in \mathbb{R}^{6\times N}$ contains the reference CoM position and body inclination, among which the reference lateral CoM position and body inclination angles are zeros by default. The state errors are weighted by the matrix $\mathbf{Q}$.

*Constraints:* Constraints (5b) define the initial conditions, including the starting CoM state $\boldsymbol{X}^+_0$ and the initial foot position $\boldsymbol{r}_0$. (5c) ensure the terminal CoM position near the 3D target position ($\boldsymbol{c}_d \in \mathbb{R}^3$), with $\boldsymbol{\epsilon}$ serving as slack parameters.

The constraint in (5d) transforms the joint angles into the feet' positions with forward kinematics. (5e) restricts leg length within kinematic reachability, defined by $L^{\max}_{\text{leg}}$.

The constraint (5f) sets boundaries for the state variables.

(5g) imposes the SRB dynamics. At each step, we

$$\begin{aligned}
\ddot{\mathbf{X}}^+_{(0:2,k)} &= \sum_{i=1}^{n_c} \mathbf{F}^i(k)/m - \boldsymbol{f}_g, \\
\frac{d}{dt}(\mathbf{I}\mathbf{X}^+_{(9:11,k)}) &= \sum_{i=1}^{n_c} (\mathbf{r}^i(k) - \mathbf{X}^+_{(0:2,k)}) \times \mathbf{F}^i(k).
\end{aligned} \tag{6}$$

In (6), linear dynamics is defined as the sum of contact forces acting on each foot, minus the gravitational force ($\boldsymbol{f}_g \in \mathbb{R}^3$). The rotational dynamics is determined by the torque generated by these forces around the CoM. The variable $n_c$ represents the number of feet in contact with the ground, $\mathbf{I}$ is the inertia tensor. Note that in the flight phase ($k \in (N_s, N]$), the robot follows a parabolic trajectory.

Equation (5h) enforces contact complementary constraints, helping to maintain contact in the stance phase and improving the convergence of the TO algorithm. Please check [14] for more details.

Inequality constraint in (5i) guarantees a reasonable joint torque, where $\mathbf{J}(\mathbf{q}^i(k)) \in \mathbb{R}^{3\times 3}$ is the contact Jacobian. The constraints in (5j) and (5k) prevent slippage.

### *B. Soft Landing Control*

To realize soft landing, we need to mitigate impact perturbations. The variable impedance control proposed in [27] could minimize accelerations while keeping tracking errors within an acceptable range by regulating the impedance online. In this work, we first introduce the basic idea and then apply it to landing control.

*1) VIC basis:* VIC works by solving the following optimization problem [27]

$$\begin{aligned}
\arg\min_{\mathbf{D}, \mathbf{K}} \quad & J(\mathbf{D}, \mathbf{K}), && \text{(7)}\\
\text{s.t.} \quad & \mathbf{D}^{\min}_{(i,j)} \le \mathbf{D}_{(i,j)} \le \mathbf{D}^{\max}_{(i,j)}, \\
& \mathbf{K}^{\min}_{(i,j)} \le \mathbf{K}_{(i,j)} \le \mathbf{K}^{\max}_{(i,j)}, \\
& \max_{\tilde{\boldsymbol{x}}_0, \dot{\tilde{\boldsymbol{x}}}_0, \boldsymbol{F}_{\text{ext}}} |\tilde{x}_i(t)| \le b_i \quad \forall t \in [0, +\infty), \\
& \quad \text{s.t.} \quad \tilde{\boldsymbol{x}}^{\max}_0 \le \tilde{\boldsymbol{x}}_0 \le \tilde{\boldsymbol{x}}^{\max}_0, \\
& \qquad \dot{\tilde{\boldsymbol{x}}}^{\min}_0 \le \dot{\tilde{\boldsymbol{x}}}_0 \le \dot{\tilde{\boldsymbol{x}}}^{\max}_0, \\
& \qquad \boldsymbol{F}^{\min}_{\text{ext}} \le \boldsymbol{F}_{\text{ext}} \le \boldsymbol{F}^{\max}_{\text{ext}}, \\
& \qquad \boldsymbol{\Lambda}(q)\ddot{\tilde{\boldsymbol{x}}} + \mathbf{D}\dot{\tilde{\boldsymbol{x}}} + \mathbf{K}\tilde{\boldsymbol{x}} = \boldsymbol{F}_{\text{ext}}.
\end{aligned}$$

with $i \in \{0, \dots, 5\}$, $j \in \{0, \dots, 5\}$.

The above problem seeks to find the optimal damping matrix ($\mathbf{D} \in \mathbb{R}^{6\times6}$) and stiffness matrix ($\mathbf{K} \in \mathbb{R}^{6\times6}$) that minimize the cost function $J(\mathbf{D}, \mathbf{K})$. The solution ensures bounded tracking errors ($\tilde{\boldsymbol{x}} \in \mathbb{R}^6$) over time. In particular, the peak value of the $i$-th component over time ($\tilde{x}_i(t)$) is constrained within $b_i$, assuming bounded external forces ($\boldsymbol{F}_{\text{ext}} \in \mathbb{R}^6$) and initial conditions ($\tilde{\boldsymbol{x}}_0 \in \mathbb{R}^6$, $\dot{\tilde{\boldsymbol{x}}}_0 \in \mathbb{R}^6$).

In the optimization problem, $\tilde{\boldsymbol{x}}$ denotes the tracking error of the CoM position and rotation. $\boldsymbol{\Lambda}(q) \in \mathbb{R}^{6\times6}$ represents the positive definite Cartesian inertia, and the vector $\boldsymbol{F}_{\text{ext}}$ encapsulates the external force/torque acting on the CoM. Bilateral constraints are applied to the variables $\mathbf{D}_{(i,j)}$, $\mathbf{K}_{(i,j)}$, $\tilde{\boldsymbol{x}}_0$, $\dot{\tilde{\boldsymbol{x}}}_0$, and $\boldsymbol{F}_{\text{ext}}$, of which the boundary values are chosen to satisfy the stability, motion tracking and feasibility requirements.

Assuming $\boldsymbol{\Lambda}(q)$, $\mathbf{D}$ and $\mathbf{K}$ *diagonally dominant*, [27] demonstrates that a closed-form solution of the above optimization problem exists. The $i$-th diagonal element of the matrix $\mathbf{D}$ (denoted as $d_i$, bounded by $d_i^{\min}$ and $d_i^{\max}$),

$$d_i = \min\left(\max\left(d_i^{\min}, \frac{2m_i\dot{\tilde{x}}_{0_i,\max}}{(b_i - \tilde{x}_{0_i,\max})\mathrm{e}}\right), d_i^{\max}\right). \tag{8}$$

where, $m_i$ represents the $i$-th diagonal element of the matrix $\boldsymbol{\Lambda}$, e is the natural constant. $\tilde{x}_{0_i,\max}$ and $\dot{\tilde{x}}_{0_i,\max}$ (with $i \in \{0,\ldots,5\}$) are defined by

$$\tilde{x}_{0_i,\max} \triangleq \max\left(|\tilde{\boldsymbol{x}}_0^{\min}(i)|, \tilde{\boldsymbol{x}}_0^{\max}(i)\right),$$
$$\dot{\tilde{x}}_{0_i,\max} \triangleq \max\left(|\dot{\tilde{\boldsymbol{x}}}_0^{\min}(i)|, \dot{\tilde{\boldsymbol{x}}}_0^{\max}(i)\right).$$

To achieve a soft landing, we demand a *critically damped* behavior in the system. Then, the $i$-th diagonal element of the matrix $\mathbf{K}$ (denoted as $k_i$) is determined as

$$k_i = d_i^2/(4m_i). \tag{10}$$

*2) Landing controller:* Upon landing, we calculate the torque command for soft landing, following

1) Calculate damping $\mathbf{D}$ and stiffness $\mathbf{K}$ with VIC.
2) Validate the gains against the stability criteria. If the stability criteria are satisfied, assign $\mathbf{D}$ and $\mathbf{K}$ as the final gains, i.e., $\mathbf{D}_{\text{final}}$ and $\mathbf{K}_{\text{final}}$. Otherwise, assign to $\mathbf{D}_{\text{final}}$ the stability bound (see [27]) plus a small increment, and $\mathbf{K}_{\text{final}}$ can be recalculated using (10).
3) Calculate the desired wrench $\mathbf{W}_{\text{com}} \in \mathbb{R}^6$ as follows
$$\mathbf{W}_{\text{com}} = \mathbf{K}_{\text{final}}\tilde{\boldsymbol{x}} + \mathbf{D}_{\text{final}}\dot{\tilde{\boldsymbol{x}}}. \tag{11}$$
4) Computing GRFs via quadratic programming (QP). Details follow the work in [11].
5) Generating VIC torque $\boldsymbol{\tau}_{\text{VIC}} \in \mathbb{R}^{12}$ with Jacobian transformation.

## IV. Supervised Learning-based Behavior Cloning

This section introduces the BC scheme for on-the-fly jumping and landing control. First, we summarize the model-based jumping control pipeline. Then, we introduce two methods to learn jump and landing behavior.

### A. Jumping Control Pipeline

Fig. 3 describes the pipeline for model-based jumping planning and soft landing control. Given the task requirement, i.e. the desired jump distance, the dual-layer offline TO generates the optimal jump trajectory. For online motion control, MPC [16] is used to generate the torque command ($\boldsymbol{\tau}_{\text{mpc}} \in \mathbb{R}^{12}$) in the stance phase. Once landing, the VIC is activated[2].

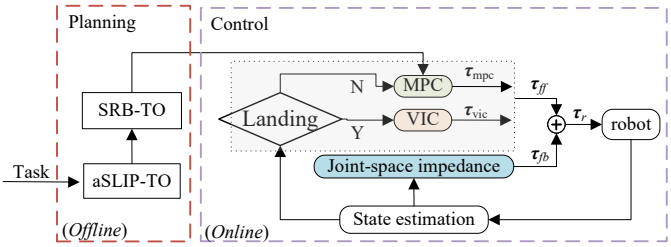

Fig. 3. Overall pipeline for model-based control.

In addition to feedforward torque ($\boldsymbol{\tau}_{ff} \in \mathbb{R}^{12}$) above, the feedback torque ($\boldsymbol{\tau}_{fb} \in \mathbb{R}^{12}$) is also considered, following

$$\boldsymbol{\tau}_{fb} = \mathbf{K}_p(\boldsymbol{q}^r - \boldsymbol{q}) + \mathbf{K}_d(\dot{\boldsymbol{q}}^r - \dot{\boldsymbol{q}}). \tag{12}$$

where $\mathbf{K}_p$ and $\mathbf{K}_d$ ($\in \mathbb{R}^{12\times12}$) are the proportional and derivative gains. The reference joint angle $\boldsymbol{q}^r \in \mathbb{R}^{12}$ and angular velocity $\dot{\boldsymbol{q}}^r \in \mathbb{R}^{12}$ are calculated by inverse kinematics.

Note that $\mathbf{K}_p$ and $\mathbf{K}_d$ are set at low values to avoid interference with the compliance provided by the VIC.

### B. Deep Learning-based BC

Since the TO process above is computationally intensive, it is difficult to execute all processes on the fly, limiting the deployment of responsive quadruped robots in real-world environments. To overcome this issue, this section introduces the BC method, which directly learns a mapping from observations to actions by mimicking expert demonstrations.

*1) Network structure:* Inspired by the work in [26], we developed and compared two distinct neural network architectures, named *Feedforward NN* and *Feedback NN*.

**Feedforward NN:** The core concept of the '*feedforward*' approach involves a single pre-jump prediction by the neural network, forecasting the trajectory over the whole jumping process. To enhance the robustness, we take the initial state and desired jumping length ($d_{\text{des}}$) as the input[3]. That is,

$$\boldsymbol{X}_{\text{FF}} = [z_0, \phi_0, \theta_0, \psi_0, \dot{x}_0, \dot{y}_0, \dot{z}_0, \omega_{r,0}, \omega_{p,0}, \omega_{y,0}, q_0, \dot{q}_0, d_{\text{des}}]. \tag{13}$$

Following the control logic in Fig. 4(a), the *Feedforward NN* predicts the desired feedforward torque, joint angle and angular velocities for low-level control. For safety checking, we also output the sagittal CoM position. As a result, the output of the *feedforward NN* ($\mathcal{O}_{FF}$) comprise

$$\mathcal{O}_{FF} = [\boldsymbol{x}_p, \boldsymbol{z}_p, \boldsymbol{\tau}_p, \boldsymbol{q}_p, \dot{\boldsymbol{q}}_p], \tag{14}$$

with $p$ indicating predicted trajectories.

As depicted in Fig. 4(a), in real applications, $\boldsymbol{\tau}_p$ replaces $\boldsymbol{\tau}_{ff}$ in Fig. 3, and $\boldsymbol{q}_p$ and $\dot{\boldsymbol{q}}_p$ facilitate the computation of $\boldsymbol{\tau}_{fb}$ according to (12). The generated trajectory contains 150 timesteps. We interpolate the output to synchronize with the controller's frequency.

[2]VIC can also be used in the stance phase. However, considering that the MPC can achieve better tracking performance, we use it for stance control.

[3]A comparison study with the *Feedforward NN* without taking the initial state as the input is attached in the video.

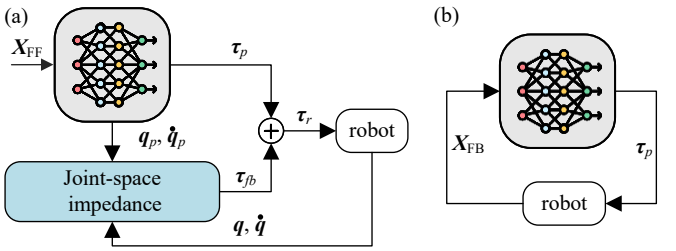


Fig. 4. Behavior clone schemes: (a) *Feedforward NN*, (b) *Feedback NN*.

**Feedback NN:** The *Feedback NN* works like a traditional feedback control system. With this structure, the neural network computes torque commands $\boldsymbol{\tau}_p$ for each loop, based on the real state of the robot. The input to this network ($\boldsymbol{X}_{\text{FB}}$) is a 36-element vector, defined as

$$\boldsymbol{X}_{\text{FB}} = [z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_r, \omega_p, \omega_y, \boldsymbol{q}, \dot{\boldsymbol{q}}, d, t]. \tag{15}$$

where $t$ is the elapsed time since the start of the motion.

Then *Feedback NN* outputs the torque command, follows

$$\mathcal{O}_{FB} = \tau_p,$$

Differing from the *Feedforward NN*, the *Feedback NN* predicts 12 joint torques at each timestep, which is applied directly to the robot, as illustrated in Fig. 4(b).

*2) Data collection:* As described in Section IV-B1, the BC methodology belongs to supervised learning. Since we adopt the deep learning framework, the quality and volume of training data are paramount. For this project, we used the simulation engine *PyBullet* [28] to generate a diverse dataset, comprising approximately 11,000 simulated jumps under varying conditions. These jumps were executed by the robot using the framework described in Fig. 3. Specifically, the robot was programmed to jump forward, ranging from 0.10m to 0.55m, in increments of 0.01m[4]. This approach ensured that each discrete distance was represented equally in the dataset, providing a comprehensive basis for training the algorithm.

To explore the whole state space while avoiding overfitting, we introduced three types of noise when collecting data:

- Gaussian noise in the initial configuration, facilitating a broad range of starting positions.
- Gaussian noise in the state, exploring the neighborhood of the trajectory and promoting resilience to sensory errors.
- An external disturbance force applied to the robot's CoM. The disturbance force, equivalent to $10\%$ of the robot's mass, was applied with a $30\%$ probability at each control iteration, with its direction randomized.

Of the 11,000 jump simulations, one third incorporated all three types of noise. The remaining two-thirds included only the first two types, ensuring a balanced and comprehensive dataset for training our BC network.

## V. Experimental Validation

This section verifies the proposed methodology. To start, we clarify the evaluation metrics for landing compliance. Then, we compare the SLIP-TO [16] and the dual-layer TO, MPC and VIC, and fully analyze the BC performance in simulation where the three types of noises described in Section IV-B2 are added in simulations when evaluating MPC, VIC and BC policies. Finally, we report the hardware experiments. Results can be seen: https://youtu.be/EEsEgtZr62s

[4]The dual-layer TO hardly generates a feasible jumping longer than 0.55m.

### A. Evaluation Metrics

As introduced in Section I, a soft landing is distinguished by two key characteristics: minimal stress on the motors and a compliant response from the robot. To evaluate motor effort, we devised two metrics: *rotational effort* and *peak torque*.

The *rotational effort* is defined as

$$\text{Rotational Effort} = \sum_{i=1}^{12} \int |\tau_i(t)| \, dt, \tag{16}$$

with $\tau_i$ being the sensory torque of the $i$-th motor, $dt$ is the time interval of each control loop.

The *peak effort* is defined as

$$\text{Peak Effort} = \max_j \left( \sqrt{\sum_{i=1}^{12} \tau_{ij}^2} \right), \tag{17}$$

with $j$ being the time instance and $i$ the motor number.

The low *rotational effort* and *peak effort* indicate a soft landing. However, although *rotational effort* and *peak effort* effectively measure motor stress, they do not directly capture the lading impact. To this end, we also examine the maximal CoM acceleration during landing, which can characterize the contact force when using the single rigid body dynamics, as listed in (6). Note that we do not evaluate the contact force directly since the measured force will drift a lot when landing occurs, especially when equipped with cheap sensors.

### B. Tracking Performance-SLIP-T0 [16] vs. dual-layer TO

We here compare the tracking performance with different reference trajectories generated by SLIP-TO in [16] and the dual-layer TO proposed in the work. For a fair comparison, we use the same controller, i.e., MPC, in both cases.

To quantify the result, we compute the mean square error (MSE) of each jumping trajectory. The robot jumped from 0.1m to 0.55m, in increments of 0.05m. Each distance is repeated 20 times (adding Gaussian noise in the state each time), and we report the mean values and standard deviations (Std.) of MSE. Fig. 5 reveals that the dual-layer TO consistently outperforms SLIP-TO (except at 0.35m-'CoMx MSE'), meaning that a refined model yields improved performance.

### C. Landing Performance in Simulation - MPC vs. VIC

To verify the soft landing on flat ground, we compare the performance with the explosive jump at different distances (ranging from 0.1m to 0.55m, increasing by 0.05m). For each distance, the robot completed 20 trials, and we report the mean values and standard deviations of each metric.

When evaluating the MPC, we adhered to the scheme outlined in Fig. 3, while continuing to use the MPC even after landing. Conversely, we switched to the VIC when the robot's four feet made contact with the ground.

Fig. 6 and Fig. 7 separately plot the *rotational effort* and *peak effort* during the landing phase. As depicted in Fig. 6, the

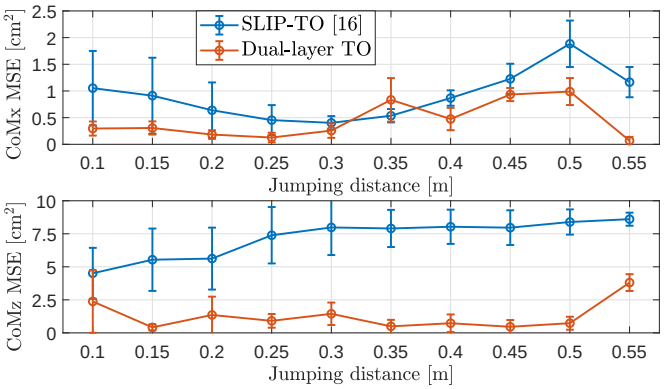


Fig. 5. CoM tracking error with SLIP-TO and dual-layer TO. 'CoMx' and 'CoMz' separately denote the forward and vertical CoM position.

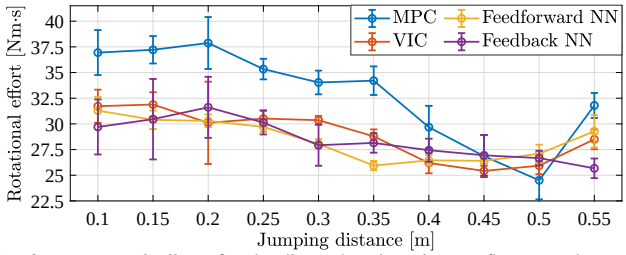


Fig. 6. *Rotational effort* after landing when jumping on flat ground.

VIC generally shows reduced *rotational effort* across various jump distances, except for the 0.50m forward jumping. In addition, Fig. 7 shows that the VIC reduces the peak effort across all jumping distances. In particular, the maximum torque is reduced by around 40% when jumping from 0.2m∼0.5m.

Besides, Fig. 8 show the maximal CoM acceleration along the $x$- (forward) and $z$- (vertical) axes with various jumping distances. It is clear that the VIC consistently outperforms the MPC in reducing vertical acceleration (see plots in 'CoMz max. acc') at all jump distances. A similar trend is also observed along the $x$- axis, except when jumping at 0.55 m.

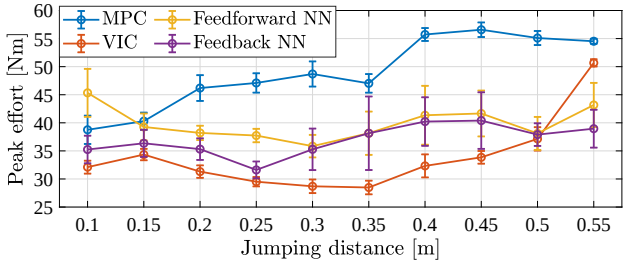


Fig. 7. *Peak effort* during the landing phase when jumping on flat ground.

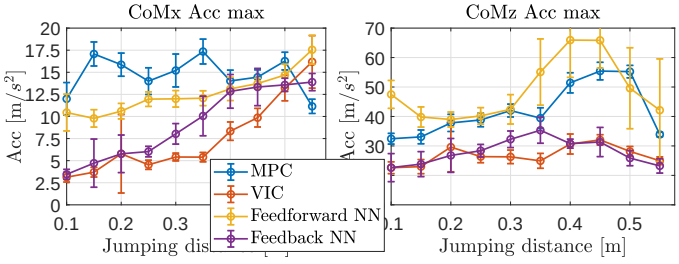


Fig. 8. Maximal CoM accelerations during the landing phase when jumping on flat ground. 'CoMx max. acc' and 'CoMz max. acc' separately denote the maximal CoM acceleration along the forward and vertical directions.

TABLE I
TRAINING SETUP FOR *Feedback* AND *Feedforward* NNS.

| | *Feedback NN* | *Feedforward NN* |
|---|---|---|
| Epochs | 400 | 1000 |
| Batch Size | 500 | 2 |
| Input Size | 36 | 35 |
| Output Size | 12 | 5700 |
| Normalization Layer | yes | yes |
| Learning Rate | 0.001 | 0.001 |
| Hidden Layer & Neuron number | $2 \times 1024$ | $2 \times 128$ |
| Activation Function | ReLU | ELU |
| Loss Function | MSE | MSE |
| Optimizer | Adam | Adam |

TABLE II
LANDING ERRORS WITH DIFFERENT METHODS.

| | *Feedback NN* | *Feedforward NN* | MPC |
|---|---|---|---|
| Error in forward CoM | 2.0±1.2cm | 5.7±3.5cm | 1.9±2.5cm |
| Error in vertical CoM | 1.7±0.7cm | 1.8±0.6cm | 1.7±0.5cm |

### D. BC Performance in Simulation

We train the neural network with the $Scikit-learn$ library [29]. The network model and training setup are detailed in Table I. In the training process, we adopted the input normalization and the early stop mechanism to avoid overfitting.

*1) Feedback vs. Feedforward NN:* Before diving into the details, we compare the landing precisions of different NNs, where the robot performs 100 jumps of random length. Table II reports the mean values and standard deviations of the forward and vertical landing errors. In addition, landing errors when using the MPC controller are also reported. As can be seen in Table II, the *Feedback NN* results in a smaller mean MSE, i.e. a higher landing precision, than *Feedforward NN*. Compared with MPC, both NNs achieve decent landing precision.

*2) NN Performance:* Firstly, we plot the *rotational effort*, *peak effort* and maximal CoM acceleration with the BC schemes against the model-based approaches above. Fig. 6, 7, and 8 demonstrate that the *Feedback NN* basically achieves a similar landing performance to that of VIC. However, the *Feedforward NN* performs a little worse than VIC in reducing CoM acceleration (see Fig. 8). We guess it is because of the lack of feedback regulation.

Secondly, we compare the solving time needed by different blocks, including the time to solve the SRB-based TO (the second layer), the time to solve the MPC, and the time for the NN to predict the result. To this end, the TO solver, MPC solver and NN prediction are all implemented in `C++`.

Table III compares the mean, standard deviation and maximum solving/prediction time calculated on 100 random jumps. We can see that the NN substantially outperforms the MPC by at least an order of magnitude in each metric. The fast computation with NN enables the quadruped robot to execute continuous jumps without the necessity of pre-computation. An example has been put in the attached video. In contrast, even with the warm start for the SLIP-based TO, the large time cost to solve the SRB-based TO makes it impractical to run the optimization online.

Lastly, we evaluate the success rate of the BC approach,

TABLE III
COMPUTING TIME NEEDED BY TO, MPC AND *Feedback NN*.

| Time cost | TO | MPC | *Feedback NN* |
|---|---|---|---|
| Mean ± Std. | 30.5±15.6s | 951±139$\mu$s | 78±3$\mu$s |
| Max | 80.1s | 5325$\mu$s | 389$\mu$s |

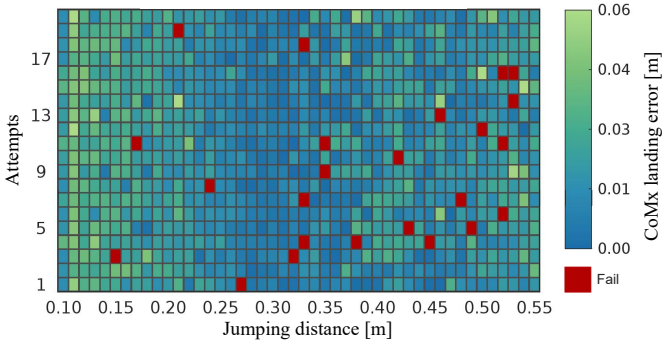


Fig. 9. The relationship between the jump length, the number of attempts per length, and the forward landing error. The red marks a failed jump.

where a 'successful' motion means that the robot lands without tipping over. Fig. 9 shows that the success rate of *Feedback NN* is approximately 97.4%. And the robot achieves a fairly small landing error in the middle of the distance range. A comparison with other policies, such as those trained with a smaller dataset and a clean dataset (without adding noise in data collection), is also conducted, see the attached video.

### *E. Jumping in Unknown Situations*

To demonstrate the scalability, we applied the proposed method to unknown situations, including jumping across uneven terrain and jumping onto an unknown slope. Here, we present the jumping across uneven terrain, where the robot jumps over the uneven ground with ±2mm height variation.

Fig. 10 shows that the model-based controller, i.e., MPC in the stance phase, obtained the smallest landing error, while *Feedback NN* realized decent tracking. In contrast, due to the lack of a feedback mechanism, *Feedforward NN* resulted in the highest landing error. In terms of the landing behavior (Fig. 11 and Fig. 12), MPC generated the largest *peak effort* and *rotational effort* while another three approaches achieved smaller values, meaning softer landings. It is worth mentioning that the *Feedback NN* resulted in the largest standard deviation in most jumping distances, meaning a fluctuated performance.

Similar results are found when jumping onto unknown slope terrain. Please check the attached video for more details.

### *F. Hardware Results*

Although the *Feedback NN* was found to be superior in achieving higher landing precision in simulation (Sections V-D1 and V-E), *Feedforward NN* offers essential benefits for hardware applications, particularly in terms of success rate and interpretability. Also, the *Feedforward NN* enables the pre-assessment of joint movements and CoM trajectories, which is crucial for safe operations. Thus, this section proceeds with the *Feedforward NN* for the hardware validation.

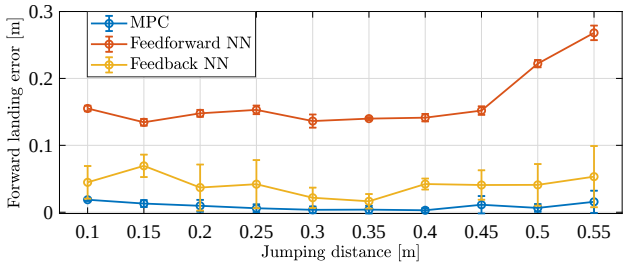


Fig. 10. Forward landing error when jumping over uneven ground.

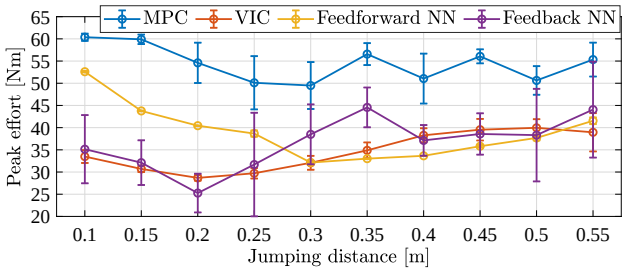


Fig. 11. *Peak effort* after landing when jumping over uneven ground.

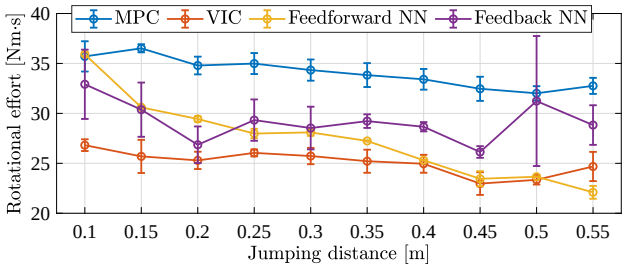


Fig. 12. *Rotational effort* after landing when jumping over uneven ground.

To adapt *Feedforward NN* for hardware execution, two modifications were implemented

- Introduction of a low-pass filter on the predicted torques.
- Refining network using an additional dataset comprising 3000 simulated jumps.

In addition, 25 actual starting positions were recorded and used to synthesize the additional dataset above for training, working as data argumentation. Following these adjustments, the robot was tested with jumps from 0.1m to 0.5m. In each trial, we judge the landing when there is a jerky joint velocity, without relying on the contact force measurement.

Each jump was repeated three times, and no failure occurred. Table IV shows that the forward BC scheme could suffer a larger landing error (e.g., when jumping at 0.1m and 0.5m). Nevertheless, Fig. 13 and Fig. 14 demonstrate that the BC method reduced the *peak effort*, *rotational effort*, and peak acceleration, achieving a softer landing.

One 0.4m forward jumping is depicted in Fig. 1. For other jumping motions, including the comparison with MPC, please check the attached video.

## VI. CONCLUSION & DISCUSSION

This work realizes explosive jumping with a soft landing by leveraging model-based and model-free approaches. We started with dual-layer optimization. In addition, we incorporated variable impedance control to achieve a soft landing

TABLE IV
AVERAGE RESULTS DURING THE LANDING PHASE (HARDWARE)

| | | 0.1m | 0.2m | 0.3m | 0.4m | 0.5m |
|---|---|---|---|---|---|---|
| Landing error [cm] | MPC [16] | 9.4 | 4.3 | 3.5 | 3.3 | 6.4 |
| | BC | 12.5 | 3.2 | 3.6 | 2.0 | 7.2 |

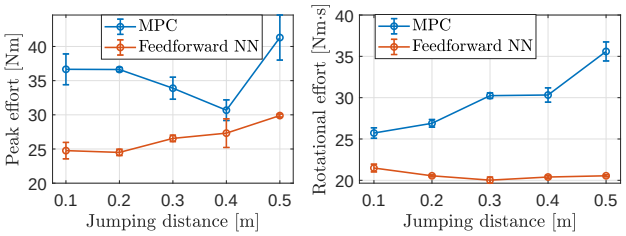


Fig. 13. *Peak effort* and *rotational effort* after landing with hardware tests.

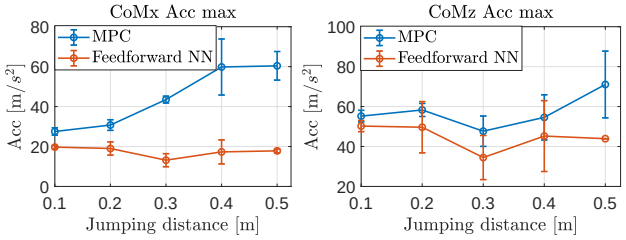


Fig. 14. Maximal CoM accelerations after landing with hardware tests.

behavior. Experiments have demonstrated that the behavior cloning approach could mitigate expert motions, realizing on-the-fly execution of jumping with a soft landing. It should be mentioned again that our approach does not rely on precise contact force measurement or compliant mechanical design.

We found that the failure with the *Feedback* network usually occurs in the flight or stance phase, as shown in the attached video. To improve the success rate, we may include also joint positions and velocities in output and add a low-gain feedback controller to track joint motion. In addition, we may improve the behavior cloning performance by using data argumentation or adopting more advanced learning mechanisms, such as transfer learning [30]. In the future, we are also keen to combine behavior cloning with deep reinforcement learning [31] to enhance jumping robustness across uneven terrain while maintaining the soft landing. Furthermore, we will extend it to 3D jumping. To this end, we will first generate the 3D jumping by reshaping the TO formulation following [16] and then retrain the policy by considering more state inputs.

## REFERENCES


[1] J. Ding, P. Posthoorn, V. Atanassov, F. Boekel, J. Kober, and C. Della Santina, "Quadrupedal locomotion with parallel compliance: E-go design, modeling, and control," *IEEE/ASME Trans. Mechatron.*, vol. 29, no. 4, pp. 2839–2848, 2024.

[2] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *Proc. IEEE In. Conf. Robot. Autom.*, 2024, pp. 11 443–11 450.

[3] J. Zhang, M. Li, J. Cao, Y. Dou, and X. Xiong, "Research on bionic jumping and soft landing of single leg system in quadruped robot," *J. Bionic Eng.*, vol. 20, no. 5, pp. 2088–2107, 2023.

[4] D. Liu, J. Wang, and S. Wang, "Force-sensorless active compliance control for environment interactive robotic systems," *IEEE/ASME Trans. Mechatron.*, 2024.

[5] M. Hutter, C. D. Remy, M. A. Hoepflinger, and R. Siegwart, "Efficient and versatile locomotion with highly compliant legs," *IEEE/ASME Trans. Mechatron.*, vol. 18, no. 2, pp. 449–458, 2012.

[6] Z. Song, L. Yue, G. Sun, Y. Ling, H. Wei, L. Gui, and Y.-H. Liu, "An optimal motion planning framework for quadruped jumping," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2022, pp. 11 366–11 373.

[7] J. Ding, M. A. van Löben Sels, F. Angelini, J. Kober, and C. Della Santina, "Robust jumping with an articulated soft quadruped via trajectory optimization and iterative learning," *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 255–262, 2023.

[8] C. Nguyen, L. Bao, and Q. Nguyen, "Continuous jumping for legged robots on stepping stones via trajectory optimization and model predictive control," in *Proc. IEEE Conf. Decis. Control.*, 2022, pp. 93–99.

[9] M. Chignoli and S. Kim, "Online trajectory optimization for dynamic aerial motions of a quadruped robot," in *Proc. IEEE In. Conf. Robot. Autom.*, 2021, pp. 7693–7699.

[10] L. Yue, L. Zhang, Z. Song, H. Zhang, J. Dong, X. Zeng, and Y.-H. Liu, "A fast online omnidirectional quadrupedal jumping framework via virtual-model control and minimum jerk trajectory generation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2024, pp. 11 993–11 999.

[11] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim, "Optimized jumping on the mit cheetah 3 robot," in *Proc. IEEE In. Conf. Robot. Autom.*, 2019, pp. 7448–7454.

[12] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[13] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

[14] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the mit mini cheetah," in *Proc. IEEE In. Conf. Robot. Autom.*, 2022, pp. 178–184.

[15] F. Roscia, M. Focchi, A. Del Prete, D. G. Caldwell, and C. Semini, "Reactive landing controller for quadruped robots," *IEEE Robot. Autom. Lett.*, pp. 7210–7217, 2023.

[16] J. Ding, V. Atanassov, E. Panichi, J. Kober, and C. Della Santina, "Robust quadrupedal jumping with impact-aware landing: Exploiting parallel elasticity," *IEEE Trans. Robot.*, vol. 40, pp. 3212–3231, 2024.

[17] Z. Lu, Y. Xiong, H. Liu, L. Yao, and Z. Wang, "Trajectory planning for jumping and soft landing with a new wheeled bipedal robot," *IEEE Trans. Ind. Inform.*, vol. 11, no. 20, pp. 13 406–13 415, 2024.

[18] F. Vezzi, J. Ding, A. Raffin, J. Kober, and C. Della Santina, "Two-stage learning of highly dynamic motions with rigid and articulated soft quadrupeds," in *Proc. IEEE In. Conf. Robot. Autom.*, 2024, pp. 9720–9726.

[19] V. Atanassov, J. Ding, J. Kober, I. Havoutis, and C. D. Santina, "Curriculum-based reinforcement learning for quadrupedal jumping: A reference-free design," *IEEE Robot. Autom. Mag.*, pp. 2–15, 2024.

[20] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning Agile Robotic Locomotion Skills by Imitating Animals," in *Proc. Robot.: Sci. Syst.*, Corvalis, Oregon, USA, July 2020.

[21] Y. Fuchioka, Z. Xie, and M. Van de Panne, "Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *Proc. IEEE In. Conf. Robot. Autom.*, 2023, pp. 5092–5098.

[22] G. Bellegarda, C. Nguyen, and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," *Robot. Auton. Syst.*, vol. 182, p. 104799, 2024.

[23] A. O. Ly and M. Akhloufi, "Learning to drive by imitation: An overview of deep behavior cloning methods," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 195–209, 2020.

[24] Q. Wang, Z. He, J. Zou, H. Shi, and K.-S. Hwang, "Behavior cloning and replay of humanoid robot via a depth camera," *Mathematics*, vol. 11, no. 3, p. 678, 2023.

[25] J. Ding, T. L. Lam, L. Ge, J. Pang, and Y. Huang, "Safe and adaptive 3-d locomotion via constrained task-space imitation learning," *IEEE/ASME Trans. Mechatron.*, vol. 28, no. 6, pp. 3029–3040, 2023.

[26] 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 *Proc. IEEE In. Conf. Robot. Autom.*, 2022, pp. 4635–4641.

[27] M. J. Pollayil, F. Angelini, G. Xin, M. Mistry, S. Vijayakumar, A. Bicchi, and M. Garabini, "Choosing stiffness and damping for optimal impedance planning," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 1281–1300, 2022.

[28] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[30] S. E. Ada, E. Ugur, and H. L. Akin, "Generalization in transfer learning: robust control of robot locomotion," *Robotica*, vol. 40, no. 11, pp. 3811–3836, 2022.

[31] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments," in *Proc. Proc. Int. Jt. Conf. Auton. Agents Multiagent Syst.*, 2020, pp. 465–473.

**Edoardo Panichi** is a Robotics Engineer with an MSc in Robotics from Delft University of Technology, graduating cum laude. He holds a BSc in Automation Engineering from the University of Bologna, where he also graduated cum laude. Edoardo currently works at X-Laboratory, developing robotic solutions for the offshore industry. His research interests include robust control systems and imitation learning approaches for quadruped robotics.

**Jiatao Ding** received his B.Eng. and Ph.D. degrees in Engineering from Wuhan University, China, in 2014 and 2020, respectively. From 2018-2020, he was a visiting Ph.D. student at the Italian Institute of Technology, Italy. From 2020-2022, he worked as an assistant research scientist at the Shenzhen Institute of Artificial Intelligence and Robotics for Society, China. From 2022-2024, he was a Postdoctoral Researcher with the Department of Cognitive Robotics, Delft University of Technology, The Netherlands

He is currently a Senior Postdoctoral Researcher with the Department of Industrial Engineering, University of Trento, Italy. His research interests include optimal control and robot learning on legged locomotion.

**Vassil Atanassov** is a PhD student at the Department of Engineering Science, University of Oxford. Previously, he obtained a BEng Mechanical Engineering degree from the University of Glasgow, graduating with First Class Honours, and obtained a Master's degree in Robotics at the Delft University of Technology, graduating with Cum Laude. His research interests are in model-based and data-driven control of legged robots.

**Peiyu Yang** is a Master's student in Robotics at the Delft University of Technology. He received his bachelor's degree in Automation from Beijing Institute of Technology in 2023, where he worked on the mechanical and system design of bipedal robots during his undergraduate studies. His current research interests include model-based and learning-based control of legged robots, with an emphasis on safe locomotion.

**Jens Kober** (Senior Member, IEEE) received the Ph.D. degree in engineering from TU Darmstadt, Darmstadt, Germany, in 2012.

He is currently an Associate Professor with TU Delft, Delft, The Netherlands. He was a Postdoctoral Scholar jointly with CoR-Lab, Bielefeld University, Bielefeld, Germany, and with Honda Research Institute Europe, Offenbach, Germany.

Dr. Kober was a recipient of the annually awarded Georges Giralt PhD Award for the best PhD thesis in robotics in Europe, the 2018 IEEE RAS Early Academic Career Award, the 2022 RSS Early Career Award, and was a recipient of an ERC Starting grant.

**Wei Pan** received the Ph.D. degree from Imperial College London, London, UK, in 2017. He is currently an Associate Professor with The University of Manchester, Manchester, UK. He was an Assistant Professor with TU Delft, The Netherlands, and Project Leader with DJI, Shenzhen, China. He has a broad interest in robot control using machine learning and the principles of dynamic control.

**Cosimo Della Santina** (Senior Member, IEEE) received his Ph.D. degree (cum laude) in robotics from the University of Pisa, Pisa, Italy, in 2019. He is currently an Associate Professor with TU Delft, Delft, The Netherlands, and a Research Scientist with German Aerospace Institute (DLR), Munich, Germany. He was a visiting Ph.D. student and a Postdoc with Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, from 2017 to 2019. He was a Senior Postdoc and a Guest Lecturer with the Department of Informatics, Technical University of Munich, in 2020 and 2021, respectively. His research interest is in providing motor intelligence to physical systems, focusing on elastic and soft robots. Cosimo is a recipient of several awards, including the euRobotics Georges Giralt Ph.D. Award in 2020 and the IEEE RAS Early Academic Career Award in 2023. He is involved as PI in a number of European and Dutch Projects, he is the co-Director of Delft AI Lab SELF, and he is a recipient of a NWO VENI.