On-the-Fly Jumping With Soft Landing
Leveraging Trajectory Optimization and Behavior Cloning

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

DOI
10.1109/TMECH.2025.3572176

Publication date
2025

Document Version
Final published version

Published in
IEEE/ASME Transactions on Mechatronics

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

Edoardo Panichi, Jiatao Ding, Vassil Atanassov, Peiyu Yang, Jens Kober, *Senior Member, IEEE*, Wei Pan, *Member, IEEE*, and Cosimo Della Santina, *Senior Member, IEEE*

***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*—Behavior cloning (BC), compliant control, neural network (NN), quadrupedal robots, trajectory optimization (TO).**

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, Chignoli and Kim[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, Yue et al.[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.

A soft landing is characterized by two main features: compliance with movement and reduction in motor stress. In this context, Jeon et al.[14] developed an optimal landing controller to regulate touchdown postures and forces. Although impressive, this work focused on falling rather than jumping. Roscia et al.[15] solved the problem of landing control with aggressive horizontal velocities, which, however, is limited by the assumption of landing on flat ground. Ding et al.[16] utilized an online TO to generate the Cartesian space landing motion but ignored the feasibility of the joint movement. Lu et al.[17]

Received 19 January 2025; revised 31 March 2025; accepted 14 May 2025. Date of publication 6 June 2025; date of current version 18 August 2025. The work was supported in part by EU project INVERSE under Grant GA 101136067. Recommended by Technical Editor X. Yu and Senior Editor Z. Sun. *(Edoardo Panichi and Jiatao Ding contributed equally to this work). (Corresponding author: Jiatao Ding.)*

Edoardo Panichi, Peiyu Yang, and Jens Kober are with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: edoardo.panichi99@gmail.com; p.yang-5@student.tudelft.nl; J.Kober@tudelft.nl).

Jiatao Ding is with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands, and also with the Department of Industrial Engineering, University of Trento, 38123 Trento, Italy (e-mail: J.Ding-2@tudelft.nl).

Vassil Atanassov is with the Oxford Robotics Institute, Department of Engineering Science, University of Oxford, OX1 2JD Oxford, U.K. (e-mail: vassilatanassov@robots.ox.ac.uk).

Wei Pan is with the Department of Computer Science, The University of Manchester, M13 9PL Manchester, U.K. (e-mail: wei.pan@manchester.ac.uk).

Cosimo Della Santina is with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands, and also with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany (e-mail: C.DellaSantina@tudelft.nl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMECH.2025.3572176.

Digital Object Identifier 10.1109/TMECH.2025.3572176

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. Deep RL has facilitated significant advancements in performing jumping tasks [18], [19], among which the imitation learning (IL) framework has been well investigated [20], [21], [22]. However, the RL framework requires careful design and reward tuning, 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 [24], [25], very limited trials in quadrupedal jumping are reported. Kurtz et al. [26] appeared 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.

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.

1) 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.
2) We develop a compliant controller to minimize landing impact and motor efforts after touch-down.
3) 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. Finally, Section VI concludes this article and discusses 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 3-D center of mass (CoM) position, while $[\phi, \theta, \psi]$ denote roll, pitch, and yaw angles. $[\dot{x}, \dot{y}, \dot{z}]$ and $[\omega_r, \omega_p, \omega_p]$ represent the linear velocity and the angular velocity, respectively. Joint positions and velocities are indicated as $\boldsymbol{q} \in \mathbb{R}^{12}$ and $\dot{\boldsymbol{q}} \in \mathbb{R}^{12}$, respectively.

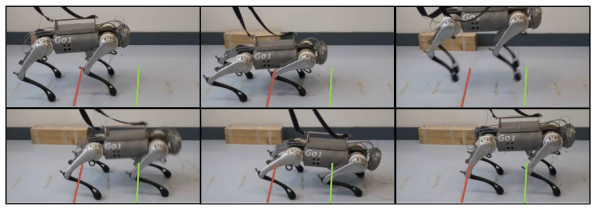

Fig. 1. Go1 jumps with a soft landing, using the learned policy from the BC. The distance between the red and green lines is 40 cm.

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 is defined as follows.

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 kth column, and the subscript $(\cdot)_{(i,j)}$ notes the element in the ith row and jth column. For the vector, $(\cdot)(k)$ refers to the kth 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, respectively.

III. Jumping Planning and Control

This section details the model-based motion planning and control. For motion planning, a dual-layer TO is proposed. For the soft landing, we resort to the VIC.

A. Jumping Motion Generation: Coarse-to-Refine TO

As illustrated in Fig. 1, 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

$$\arg\min_{\mathbf{c},\mathbf{a},\boldsymbol{dt}} \quad J_{\text{cost}} \quad (3a)$$

$$\text{s.t.} \quad \textbf{Kinematic constraints}$$

$$\mathbf{c}(0) = \boldsymbol{c}_0, \quad \dot{\mathbf{c}}(0) = \mathbf{0} \quad (3b)$$

$$\mathbf{c}_{(1,N-1)} = z_f^r \tag{3c}$$

$$\mathbf{c}_{(1,N_s-1)} = z^r_{\text{takeoff}} \tag{3d}$$

$$\boldsymbol{c}^{\min} \le \mathbf{c}(k) \le \boldsymbol{c}^{\max} \quad \forall k \le N_s \tag{3e}$$

$$\boldsymbol{dt}^{\min} \le \boldsymbol{dt} \le \boldsymbol{dt}^{\max} \tag{3f}$$

Dynamics constraints

$\forall k \in [0, 1, \ldots, N-1]$

$$\mathbf{c}(k+1) = \text{Taylor}(\mathbf{c}(k), \dot{\mathbf{c}}(k), \ddot{\mathbf{c}}(k)) \tag{3g}$$

$\forall k \le N_s$:

$$\ddot{\mathbf{c}}(k) = \mathbf{F}_s(\mathbf{c}(k))/m + \boldsymbol{g} + \mathbf{a}(k) \tag{3h}$$

$$\mathbf{a}_{(1,k)} + \boldsymbol{F}_{s,z}(\mathbf{c}(k))/m \ge 0 \tag{3i}$$

$$|\ddot{\mathbf{c}}_{(0,k)}/(\ddot{\mathbf{c}}_{(1,k)} + g)| \le \mu \tag{3j}$$

$\forall k > N_s$

$$\ddot{\mathbf{c}}(k) = \boldsymbol{g} \tag{3k}$$

where $\mathbf{c} \in \mathbb{R}^{2\times N}$ is the sagittal CoM position where the first and second row separately denote the forward and vertical position. $\mathbf{a} \in \mathbb{R}^{2\times N}$ is the sagittal acceleration resulting from the actuation force, and $\boldsymbol{dt} \in \mathbb{R}^2$ is the time steps for the stance and flight phase. $\mathbf{F}_s(\mathbf{c}(k)) \in \mathbb{R}^2$ is spring force 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

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

with x_f^r and z^r_{takeoff} being the desired landing distance and takeoff height, respectively, and, β_a, β_{jerk}, β_f, and β_x, being the coefficients.

Constraints: Equation (3b) ensures that the trajectory starts from a particular initial state (with $\boldsymbol{c}_0 \in \mathbb{R}^2$ being the initial CoM position). Equation (3c) ensures that the robot lands at the desired height z_f^r (as illustrated in Fig. 1). Equation (3d) regulates the height of the takeoff (z^r_{takeoff}) so that the robot can jump. Furthermore, (3e) obeys the kinematic reachability. Equation (3f) limits the timestep.

The dynamics constraints ensure that the robot follows the aSLIP dynamics [with (3h) in stance and (3k) in flight]. Equation (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 μ being the friction coefficient.

2) SRB-Based Kino-Dynamics Optimization: Using the first layer as the reference, we refine the motion with kino-dynamics TO, leveraging the SRB dynamics,[1] in the following:

$$\arg\min_{\mathbf{X}^+, \mathbf{F}, \mathbf{r}} \quad \|\tilde{\mathbf{X}} - \tilde{\mathbf{X}}^r\|^2_{\mathbf{Q}} \tag{5a}$$

s.t. **Kinematic constraints**

$$\mathbf{X}^+(0) = \boldsymbol{X}_0^+, \quad \mathbf{r}(0) = \boldsymbol{r}_0 \tag{5b}$$

$$\boldsymbol{c}_d - \boldsymbol{\epsilon} \le \mathbf{X}_{(0:2,N-1)} \le \boldsymbol{c}_d + \boldsymbol{\epsilon} \tag{5c}$$

$\forall k \in [1, 2, \ldots, N]$

$$\mathbf{r}^i(k) = \text{FK}(\mathbf{q}^i(k)), \quad i \in \{1, 2, 3, 4\} \tag{5d}$$

$$\|\mathbf{h}^i(k) - \mathbf{r}^i(k)\| \le L^{\max}_{\text{leg}} \tag{5e}$$

$$\mathbf{X}^+(k) \in \mathcal{B} \tag{5f}$$

Dynamics constraints

$\forall k \le N-1$

$$\dot{\mathbf{X}}^+(k+1) = f(\mathbf{X}^+(k), \mathbf{r}(k), \mathbf{F}(k)) \tag{5g}$$

$\forall k \le N_s$

$$CCC(\mathbf{F}(k), \mathbf{r}(k)) \tag{5h}$$

$$|[\mathbf{J}(\mathbf{q}^i(k))]^{\mathrm{T}} \mathbf{F}^i(k)| \le \tau_{\max} \tag{5i}$$

$$-\mu \mathbf{F}^i_{(2,k)} \le \mathbf{F}^i_{(0,k)} \le \mu \mathbf{F}^i_{(2,k)} \tag{5j}$$

$$-\mu \mathbf{F}^i_{(2,k)} \le \mathbf{F}^i_{(1,k)} \le \mu \mathbf{F}^i_{(2,k)}. \tag{5k}$$

With this formulation, we optimize the full state ($\mathbf{X}^+ \in \mathbb{R}^{36\times N}$), 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 ith leg, respectively.

Cost functions: Equation (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 $\boldsymbol{r}_0$. Equations (5c) ensure the terminal CoM position near the 3-D 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. Equation (5e) restricts leg length within kinematic reachability, defined by $L^{\max}_{\text{leg}}$.

The constraint (5f) sets boundaries for the state variables. Equation (5g) imposes the SRB dynamics. At each step, we have that

$$\ddot{\mathbf{X}}^+_{(0:2,k)} = \sum_{i=1}^{n_c} \mathbf{F}^i(k)/m - \boldsymbol{f}_g$$

$$\frac{d}{dt}(\mathbf{I}\mathbf{X}^+_{(9:11,k)}) = \sum_{i=1}^{n_c} (\mathbf{r}^i(k) - \mathbf{X}^+_{(0:2,k)}) \times \mathbf{F}^i(k). \tag{6}$$

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

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, and $\mathbf{I}$ 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, maintains contact in the stance phase, and improves the convergence of the TO algorithm. Check [14] for details.

The inequality constraint in (5i) guarantees a reasonable joint torque, where $\mathbf{J}(\mathbf{q}^i(k)) \in \mathbb{R}^{3\times3}$ 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 VIC [27] could minimize accelerations while keeping tracking errors within an acceptable range by regulating the impedance online. Here, 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] :

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

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

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 ith 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 ($\tilde{\boldsymbol{x}}_0 \in \mathbb{R}^6$ and $\dot{\tilde{\boldsymbol{x}}}_0 \in \mathbb{R}^6$).

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

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

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

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

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

$$
k_i = d_i^2/(4m_i). \qquad (10)
$$

2) Landing Controller: Upon landing, we calculate the torque command for soft landing in the 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, then 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:

$$
\mathbf{W}_{\text{com}} = \mathbf{K}_{\text{final}}\tilde{\boldsymbol{x}} + \mathbf{D}_{\text{final}}\dot{\tilde{\boldsymbol{x}}}. \qquad (11)
$$

4) Computing GRFs via quadratic programming. Details follow the work in [11].
5) Generating $\boldsymbol{\tau}_{\text{VIC}} \in \mathbb{R}^{12}$ with Jacobian transformation.

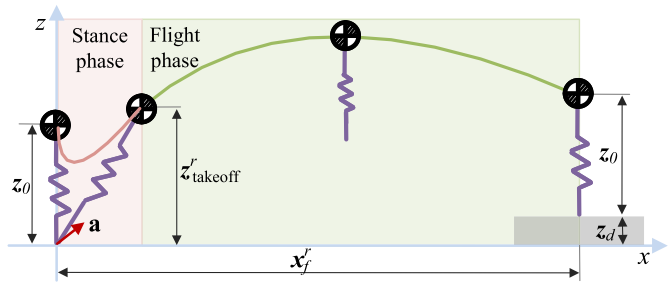

Fig. 2. Example of jumping trajectory generated with the aSLIP model. The red zone and green zone separately mark the stance and the flight phase.

IV. Supervised Learning-Based BC

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. 2 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

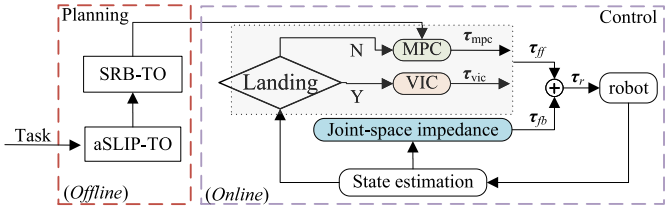

Fig. 3. Overall pipeline for model-based control.

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]

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

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

where $\mathbf{K}_p$ and $\mathbf{K}_d$ ($\in \mathbb{R}^{12\times 12}$) are the proportional and derivative gains, respectively. 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 NN architectures, named *feedforward NN* and *feedback NN*.

Feedforward NN: The core concept of the "*feedforward*" approach involves a single prejump prediction by the NN, forecasting the trajectory over the whole jumping process. To enhance the robustness, we take the initial state and desired jumping length (d_{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. 3(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* ($\mathcal{O}_{\text{FF}}$) comprises

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

with p indicating predicted trajectories.

As depicted in Fig. 3(a), in real applications, $\boldsymbol{\tau}_p$ replaces $\boldsymbol{\tau}_{\text{ff}}$ in Fig. 2, 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.

Feedback NN: The *feedback NN* works like a traditional feedback control system, where the NN computes torque commands $\boldsymbol{\tau}_p$ 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, as follows:

$$\mathcal{O}_{\text{FB}} = \tau_p.$$

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. 3(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. 2. Specifically, the robot was programmed to jump forward, ranging from 0.10 to 0.55 m, in increments of 0.01 m.[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 the following three types of noise when collecting data.

1) Gaussian noise in the initial configuration, facilitating a broad range of starting positions.
2) Gaussian noise in the state, exploring the neighborhood of the trajectory and promoting resilience to sensory errors.
3) 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 online.[5]

[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 Supplementary Material.

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

[5][Online]. Available: https://youtu.be/EEsEgtZr62s

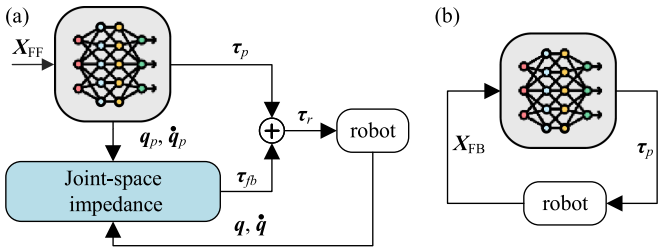

Fig. 4. Behavior clone schemes. (a) *Feedforward NN*. (b) *Feedback NN*.

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 τ_i being the sensory torque of the ith motor, and 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 landing impact. To this end, we also examine the maximal CoM acceleration during landing, which can characterize the contact force when using the SRB 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-TO [16] Versus 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 0.1 to 0.55 m, in increments of 0.05 m. 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. 4 reveals that the dual-layer TO consistently outperforms SLIP-TO (except at 0.35 m-"CoMx MSE"), meaning that a refined model yields improved performance.

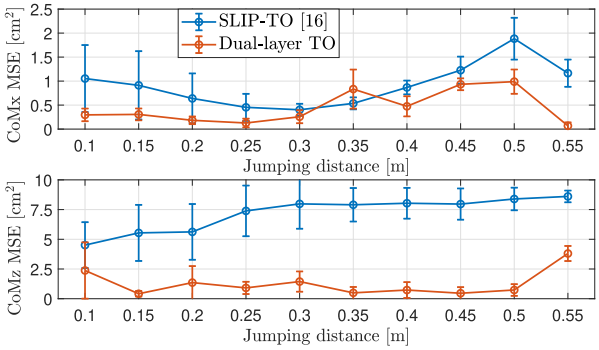

Fig. 5. CoM tracking error with SLIP-TO and dual-layer TO. "CoMx" and "CoMz" separately denote the forward and vertical CoM position, respectively.

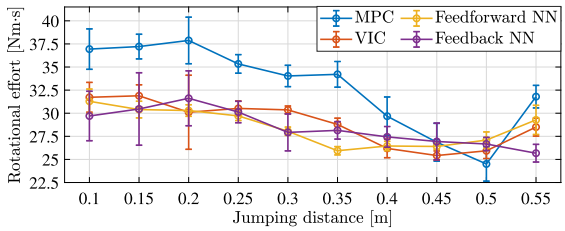

Fig. 6. *Rotational effort* after landing when jumping on flat ground.

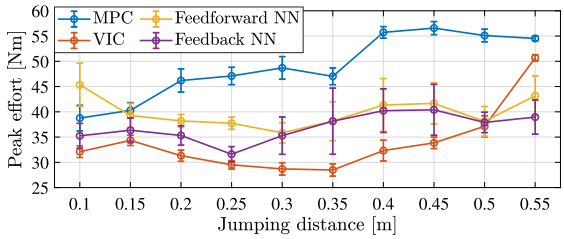

Fig. 7. *Peak effort* during the landing phase when jumping on flat ground.

C. Landing Performance in Simulation—MPC Versus VIC

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

When evaluating the MPC, we adhered to the scheme outlined in Fig. 2, 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.

Figs. 5 and 6 separately plot the *rotational effort* and *peak effort* during the landing phase. As depicted in Fig. 5, the VIC generally shows reduced *rotational effort* across various jump distances, except for the 0.50-m forward jumping. In addition, Fig. 6 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 0.2–0.5m.

Besides, Fig. 7 shows 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

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	2×1024	2×128
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

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

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

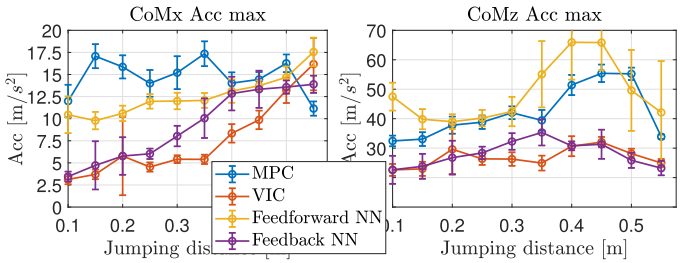

Fig. 8. Maximal CoM accelerations during the landing phase when jumping on flat ground. "CoMx max. acc" and "CoMz max. acc" separately denote the maximal CoM acceleration along the forward and vertical directions.

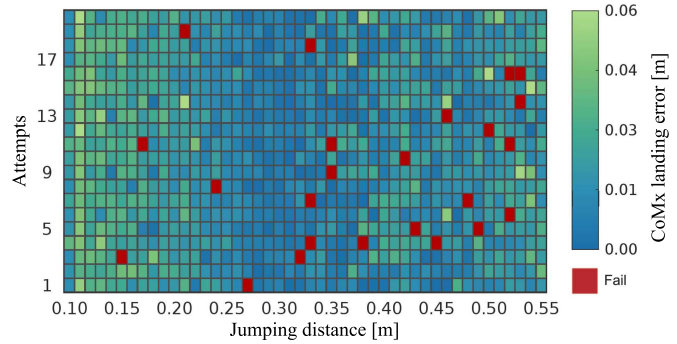

Fig. 9. Relationship between the jump length, the number of attempts per length, and the forward landing error. The red marks a failed jump.

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.

D. BC Performance in Simulation

We train the NN 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 Versus 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 Std. 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: First, we plot the *rotational effort*, *peak effort*, and maximal CoM acceleration with the BC schemes against the model-based approaches. Figs. 5–7 demonstrate that the *feedback NN* basically achieves a similar landing performance to VIC while the *feedforward NN* performs a little worse in reducing CoM acceleration (see Fig. 7). We guess it is because of the lack of feedback regulation.

Second, 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, Std., and maximum solving/prediction time for 100 random jumps. We see that the NN substantially outperforms the MPC by at least an order of magnitude in all metrics. The fast computation with NN enables the quadruped robot to execute continuous jumps without the necessity of precomputation, as can be seen in the Supplementary Material. 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.

Last, we evaluate the success rate of the BC approach, where a "successful" motion means that the robot lands without tipping over. Fig. 8 shows that the success rate of *feedback NN* is approximately 97.4%. 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 Supplementary Material.

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 ± 2 mm height variation.

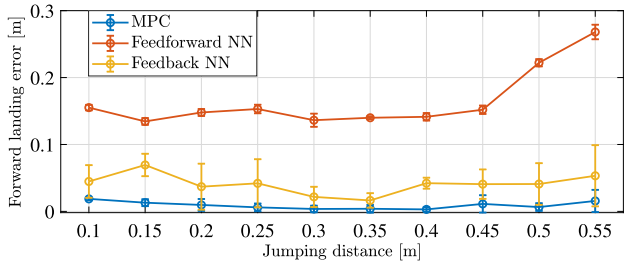

Fig. 10. Forward landing error when jumping over uneven ground.

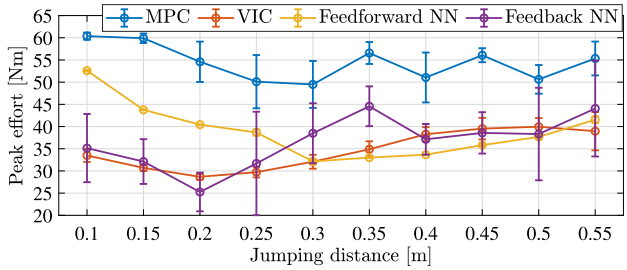

Fig. 11. *Peak effort* after landing when jumping over uneven ground.

Fig. 9 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 (see Figs. 10 and 11), 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 Std. in most jumping distances, meaning a fluctuated performance.

Similar results are found when jumping onto unknown slope terrain. Check the Supplementary Material for more details.

F. Hardware Results

Although the *feedback NN* was found to be superior in achieving higher landing precision in simulation (see 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 preassessment of joint movements and CoM trajectories, which is crucial for safe operations. Thus, this section proceeds with the *feedforward NN* for the hardware validation. To adapt *feedforward NN* for hardware execution, the following two modifications were implemented.

1) Introduction of a low-pass filter on the predicted torques.
2) 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 0.1 to 0.5 m. In each trial, we judge the landing when there is a jerky joint velocity, without relying on the contact force measurement.

TABLE IV
AVERAGE RESULTS DURING THE LANDING PHASE (HARDWARE)

		0.1 m	0.2 m	0.3 m	0.4 m	0.5 m
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

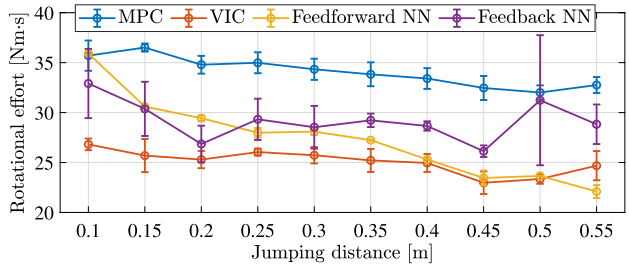

Fig. 12. *Rotational effort* after landing when jumping over uneven ground.

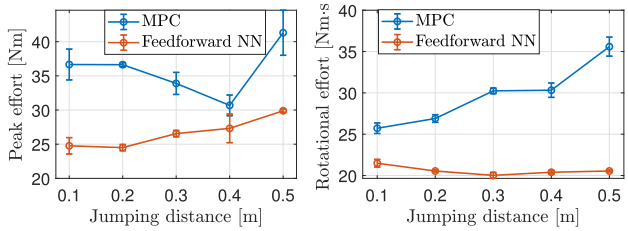

Fig. 13. *Peak effort* and *rotational effort* after landing with hardware tests.

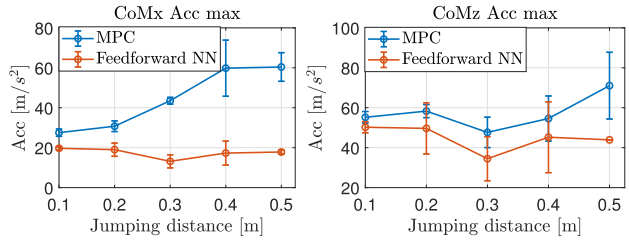

Fig. 14. Maximal CoM accelerations after landing with hardware tests.

Each jump was repeated three times, and no failure occurred. Table IV gives that the forward BC scheme could suffer a larger landing error (e.g., when jumping at 0.1 and 0.5 m). Nevertheless, Figs. 12 and 13 demonstrate that the BC method reduced the *peak effort*, *rotational effort*, and peak acceleration, achieving a softer landing.

One 0.4 m forward jumping is depicted in Fig. 14. For other jumping motions, including the comparison with MPC, check the Supplementary Material.

VI. Conclusion and 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 VIC to achieve a soft landing behavior. Experiments have demonstrated that the BC 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 Supplementary Material. To improve the success rate, we may also include joint positions and velocities in output and add a low-gain feedback controller to track joint motion. In addition, we may improve the BC 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 BC with deep RL [31] to enhance jumping robustness across uneven terrain while maintaining the soft landing. Furthermore, we will extend it to 3-D jumping. To this end, we will first generate the 3-D 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, Aug. 2024.

[2] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 11443–11450.

[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.*, vol. 30, no. 2, pp. 1481–1491, Apr. 2025.

[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, Apr. 2013.

[6] Z. Song et al., "An optimal motion planning framework for quadruped jumping," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2022, pp. 11366–11373.

[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, Jan. 2024.

[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 Int. Conf. Robot. Autom.*, 2021, pp. 7693–7699.

[10] L. Yue et al., "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. 11993–11999.

[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 Int. 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," 2019, *arXiv:1909.06586*.

[14] S. H. Jeon, S. Kim, and D. Kim, "Online optimal landing control of the MIT mini Cheetah," in *Proc. IEEE Int. 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.*, vol. 8, no. 11, pp. 7210–7217, Nov. 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. Informat.*, vol. 11, no. 20, pp. 13406–13415, Nov. 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 Int. 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.*, early access, Nov. 20, 2024, doi: 10.1109/MRA.2024.3487325.

[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, OR, USA, Jul. 2020, pp. 12–16.

[21] Y. Fuchioka, Z. Xie, and M. Van de Panne, "OPT-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *Proc. IEEE Int. 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, 2024, Art. no. 104799.

[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, Jun. 2021.

[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, 2023, Art. no. 678.

[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, Dec. 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 Int. Conf. Robot. Autom.*, 2022, pp. 4635–4641.

[27] M. J. Pollayil et al., "Choosing stiffness and damping for optimal impedance planning," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 1281–1300, Apr. 2023.

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

[29] F. Pedregosa 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. Int. Jt. Conf. Auton. Agents Multiagent Syst.*, 2020, pp. 465–473.

Edoardo Panichi received the B.Sc. degree (cum laude) in automation engineering from the University of Bologna, Bologna, Italy, in 2021, and the M.Sc. degree (cum laude) in robotics from the Delft University of Technology, Delft, The Netherlands, in 2024.

He is a Robotics Engineer and is currently with 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 the B.Eng. and Ph.D. degrees in engineering from Wuhan University, Wuhan, China, in 2014 and 2020, respectively.

From 2018 to 2020, he was a visiting Ph.D. student with the Italian Institute of Technology, Genoa, Italy. From 2020 to 2025, he was an Assistant Research Scientist with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen, China, and a Postdoctoral Researcher with the Department of Cognitive Robotics, Delft University of Technology, Delft, The Netherlands. He is currently a Senior Postdoctoral Researcher with the Department of Industrial Engineering, University of Trento, Trento, Italy. His research interests include optimal control and robot learning on complex robotic systems.

Vassil Atanassov received the B.Eng. degree (First Class Hons.) in mechanical engineering from the University of Glasgow, Glasgow, U.K., in 2021, and the master's degree (cum laude) in robotics from the Delft University of Technology, Delft, The Netherlands, in 2023. He is currently working toward the Ph.D. degree in Robobics with the Department of Engineering Science, University of Oxford, Oxford, U.K.

His research interests include model-based and data-driven control of legged robots.

Peiyu Yang received the bachelor's degree in automation, working on the mechanical and system design of bipedal robots, from the Beijing Institute of Technology, Beijing, China, in 2023. He is currently working toward the master's degree in robotics with the Delft University of Technology, Delft, The Netherlands.

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

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

Wei Pan (Member, IEEE) received the Ph.D. degree in bioengineering from Imperial College London, London, U.K., in 2017.

He is currently an Associate Professor with The University of Manchester, Manchester, U.K. He was an Assistant Professor with TU Delft, Delft, The Netherlands, and a Project Leader with DJI, Shenzhen, China. His research interests include robot control using machine learning and the principles of dynamic control.

Cosimo Della Santina (Senior Member, IEEE) received the Ph.D. degree (cum laude) in robotics from the University of Pisa, Pisa, Italy, in 2019.

From 2017 to 2019, he was a visiting Ph.D. student and a Postdoc with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. In 2020 and 2021, respectively, he was a Senior Postdoc and a Guest Lecturer with the Department of Informatics, Technical University of Munich, Munich, Germany. He is currently an Associate Professor with TU Delft, Delft, The Netherlands, and a Research Scientist with German Aerospace Institute, Munich. His research focuses on providing motor intelligence to physical systems, focusing on elastic and soft robots.

Dr. Santina was the recipient of several awards, including the euRobotics Georges Giralt Ph.D. Award in 2020 and the IEEE RAS Early Academic Career Award in 2023, and also the recipient of an NWO VENI. He is a PI in a number of European and Dutch Projects and the Co-Director of Delft AI Lab SELF.