IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

OmniNet: Omnidirectional Jumping Neural Network with Height-awareness for Quadrupedal Robots

Yimin Han*, Jiahui Zhang*, Zeren Luo*, Yingzhao Dong, Jinghan Lin, Liu Zhao, Shihao Dong, and Peng Lu†

***Abstract*—In the robotics community, it has been a long-standing challenge for quadrupeds to achieve highly explosive movements similar to their biological counterparts. In this work, we introduce a novel training framework that achieves height-aware and omnidirectional jumping for quadrupedal robots. To facilitate the precise tracking of the user-specified jumping height, our pipeline concurrently trains an estimator that infers the robot and its end-effector states in an online fashion. Besides, a novel reward is involved by solving the analytical inverse kinematics with pre-defined end-effector positions. Guided by this term, the robot is empowered to regulate its gestures during the aerial phase. In the comparative studies, we verify that this controller can not only achieve the longest relative forward jump distance, but also exhibit the most comprehensive jumping capabilities among all the existing jumping controllers. A video summarizing the methodology and the validation in both simulation and real hardware is submitted along with this paper.**

***Index Terms*—Legged Robots, Reinforcement Learning, Machine Learning for Robot Control**

I. INTRODUCTION

IN natural settings, legged animals exhibit exceptional agility characterized by dynamic movements, such as jumping. However, replicating these behaviors in quadrupedal robots presents significant challenges. When operating at high speeds, the robot's dynamics are impacted by several physical factors, including the motor limits, the adjustment of contact forces, and the robot's maneuverability during its airborne phases. Proposing a universal framework that enables agile and versatile jumping is significant for enhancing the mobility potential of quadruped robots.

A. Model-based Method

Trajectory optimization (TO) is a widely used technique for enabling legged robots to execute aerial movements. For static jumping, TO is normally solved with the less approximated dynamic models [1], [2], [3] to produce the omnidirectional motions in 3-dimensional spaces. The resulting optimization is time-consuming or even infeasible, restricting these approaches to offline implementation. To fully utilize the robot's maneuverability, more demanding running jumps are explored which require real-time optimal trajectory. Through simplified models, such as the two-legged system in the sagittal plane [4], [5], and single-rigid body dynamics [6] speed up convergence, their performance could still be undermined by the violation of physical constraints. Moreover, these approaches require a specialized and complex design of the corresponding landing controller to ensure a smooth and safe recovery from the aerial phase.

Fig. 1: Versatile jumping experiments conducted on Unitree-Go2, including omnidirectional jumping and landing on different terrains.

B. Learning-based Method

Learning-based methods have been considered as an alternative line for generating highly dynamic behaviors [7], [8]. Early studies utilize reinforcement learning (RL) to generate directly the optimal jumping trajectory [9] or the compensatory adjustments [10]. Nevertheless, they are constrained by the predefined gait, exhibiting quite limited mobility capabilities.

Learning from demonstration is also made possible with the recent advances in computer graphics and robotics [11], [12]. It is now possible to mimic life-like agile jumping from dataset via imitation learning [13], [14], [15], [16]. Besides, with the hierarchical RL, the controllers can switch between various sub-policies and enable the end-effectors to perform highly

Manuscript received: February, 10, 2025; Revised May, 12, 2025; Accepted June, 8, 2025.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the General Research Fund under Grant 17204222, and in part by the Seed Fund for Collaborative Research.

The authors are with the Adaptive Robotic Controls Lab (ArcLab), Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR, China. ymhan2023@connect.hku.hk, holmesz@connect.hku.hk, zerluo@connect.hku.hk, dongyz@connect.hku.hk, linjh@connect.hku.hk, zhaol@connect.hku.hk, dongsh24@connect.hku.hk.

* Equal Contribution. †Corresponding author: lupeng@hku.hk.

The supplementary video is available at https://youtu.be/1Weu46sxc78

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

dynamic motions, guided by the reference that are generated from the motion planners [17], [18]. However, most imitation-based approaches demonstrate restricted generalization abilities beyond the imitated datasets or are constrained by the motion planner, allowing only specific jumping patterns.

More recent studies of reference-free domains pose more challenges while normally requiring a delicate curriculum design. Atanassov et al [19] involves designing a task-based curriculum and complex reward shaping heading from vertical to horizontal jumps. Similarly, recent studies have a well-designed curriculum focusing on terrains [7] and contact [8] to facilitate the generation of jumping movements. Moreover, most jumping controllers still rely on separate estimation modules or perfect state assumptions [19], [20], while joint training of policies and state estimators has shown promise for robust locomotion [21]. Unlike these approaches, our pipeline not only eliminates the need for complex curricula by using fewer reward terms in a single training stage, but also integrates state awareness directly into the policy to enhance robustness in real-world deployment.

In summary, we propose an **Omni**directional jumping neural **Net**work (OmniNet) for quadrupedal robots to achieve omni-directional and height-tracking jumping, as shown in Fig. 1. The key contributions of this work are listed as follows:

- We introduce a universal deep reinforcement learning (DRL) framework for quadruped omnidirectional jumping that enables agile jumping in multiple directions and robust landing without a predefined trajectory or reference dataset.
- The proposed controller possesses the capability to infer the CoM height and feet height of the robot during the jumping process, which implicitly helps to realize the height-variant jumping. The accurate height estimation also plays a vital role in autonomous seamless switching between single and continuous jumping.
- The training process enhances the dynamic robot gesture controlling in the air through a specific reward term. To control the joint angle effectively, the analytical inverse kinematic algorithm is implemented so that each angle can be regulated based on the desired joint angle computed according to the predefined feet end-effector height under the body frame.
- The proposed framework demonstrates promising potential for accommodating different gaits. Utilizing the estimated height, the jumping controller and walking controller can be switched to each other smoothly.

II. METHOD

A. Problem Formulation

Our goal is to develop a locomotion control policy for jumping skills that can perform omnidirectional targeted jumps. We formulate the jumping control problem as a Markov Decision Process (MDP) defined by $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma\}$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. For state $\boldsymbol{s}_t$ ($\boldsymbol{s}_t \in \mathcal{S}$) at time step t, a reinforcement learning (RL) policy π_θ performs an action $\boldsymbol{a}_t$ ($\boldsymbol{a}_t \in \mathcal{A}$) to forward the environment to the next state $\boldsymbol{s}_{t+1}$ with transition probability $\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ meanwhile receiving reward $\mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t)(r_t$ for abbreviation). RL aims to find the optimal parameter θ that maximizes the accumulated rewards $J(\theta)$ of this MDP with a discount ratio $\gamma \in [0, 1]$, i.e.: $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}(\gamma^t r_t)\right]$.

B. Height-Aware Controller

1) Observation space: The proprioceptive observation $\boldsymbol{o}_t = (\boldsymbol{\omega}_t, \boldsymbol{g}_t, \boldsymbol{v}_t^{xy-cmd}, \omega_t^{yaw-cmd}, h_t^{cmd}, \boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{a}_{t-1})$ in our task consists of base angular velocity in the robot's frame $\boldsymbol{\omega}_t \in \mathbb{R}^3$, projected gravity $\boldsymbol{g}_t \in \mathbb{R}^3$, commanded linear velocity $\boldsymbol{v}_t^{xy-cmd} \in \mathbb{R}^2$, commanded angular velocity $\omega_t^{yaw-cmd} \in \mathbb{R}^1$, and commanded height $h_t^{cmd} \in \mathbb{R}^1$, joint angles $\boldsymbol{q}_t \in \mathbb{R}^{12}$, joint velocities $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{12}$, and the action of the last step $\boldsymbol{a}_{t-1} \in \mathbb{R}^{12}$. Meanwhile, we define a temporal observation $\boldsymbol{o}_t^H = [\boldsymbol{o}_t, \boldsymbol{o}_{t-1}, ..., \boldsymbol{o}_{t-H}]$ to store the proprioceptive observations over the past H time steps ($H = 20$ in our task).

2) Action space: The action of locomotion policy $\boldsymbol{a}_t \in \mathbb{R}^{12}$ represents the desired increment of the joint angle w.r.t the initial pose $\mathring{\boldsymbol{q}}$, i.e. $\boldsymbol{a}_t = \boldsymbol{q}_t^* - \mathring{\boldsymbol{q}}$. The final desired angle $\boldsymbol{q}_t^*$ is tracked by the torque generated by the joint-level PD controller, i.e. $\boldsymbol{\tau} = \boldsymbol{K}_p \cdot (\boldsymbol{q}_t^* - \boldsymbol{q}_t) + \boldsymbol{K}_d \cdot (-\dot{\boldsymbol{q}}_t)$.

3) Neural Network Design: As shown in Fig. 2, the jumping framework consists of three sub-networks: the policy net, the value net, and the height-estimator net, which work together to achieve versatile and height-aware jumping tasks. These three networks are trained together via PPO [22] in simulation, and the parameters of the height estimator are also updated via a regression algorithm. Since certain privileged information can significantly enhance the stability and agility of the robot's jumping, our controller adopts the Asymmetric Actor-Critic [23] structure for robot learning, which enables more effective exploration under high-dimensional spaces of legged robots. The actor and critic operate with different goals: the actor seeks to maximize expected rewards, while the critic focuses on minimizing the discrepancy between predicted and actual values. Each sub-network of the jumping controller will be discussed in the following parts.

Actor Net: The actor net is applied to infer the action $a_t \in \mathbb{R}^{12}$ by taking the combined input $\boldsymbol{p}_t$ which includes the estimated CoM position $\hat{\boldsymbol{P}}_{CoM} \in \mathbb{R}^3$, the estimated feet positions $\hat{\boldsymbol{H}}_{feet} \in \mathbb{R}^4$, the estimated linear velocity $\hat{\boldsymbol{v}}_t \in \mathbb{R}^3$, and the proprioception observation $\boldsymbol{o}_t$. The estimated values are the outputs of the height-estimator net. $\boldsymbol{p}_t$ and $\hat{\boldsymbol{P}}_{CoM}$ are defined as below, respectively:

$$\boldsymbol{p}_t = [\hat{\boldsymbol{P}}_{CoM}, \hat{\boldsymbol{H}}_{feet}, \hat{\boldsymbol{v}}_t, \boldsymbol{o}_t], \tag{1}$$

$$\hat{\boldsymbol{P}}_{CoM} = \{z, x, y\}. \tag{2}$$

Value Net: Apart from $\boldsymbol{o}_t$, another portion of the input of the value net are those that are expensive to obtain or readily deviate induced by the inaccurate measurement. In order to assess the value of the current states more accurately and provide useful feedback to the actor net, the critic net integrates more ground-truth physical knowledge, such as the exteroceptive information about the terrain, which is the ego-centric height map scan dots from the robot-centered area of the size $1.1m * 1.6m$: $\boldsymbol{h}_t \in \mathbb{R}^{187}$, the physical states including CoM positions $\tilde{\boldsymbol{P}}_{CoM} \in \mathbb{R}^3$, feet height $\tilde{\boldsymbol{H}}_{feet} \in \mathbb{R}^4$, and

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

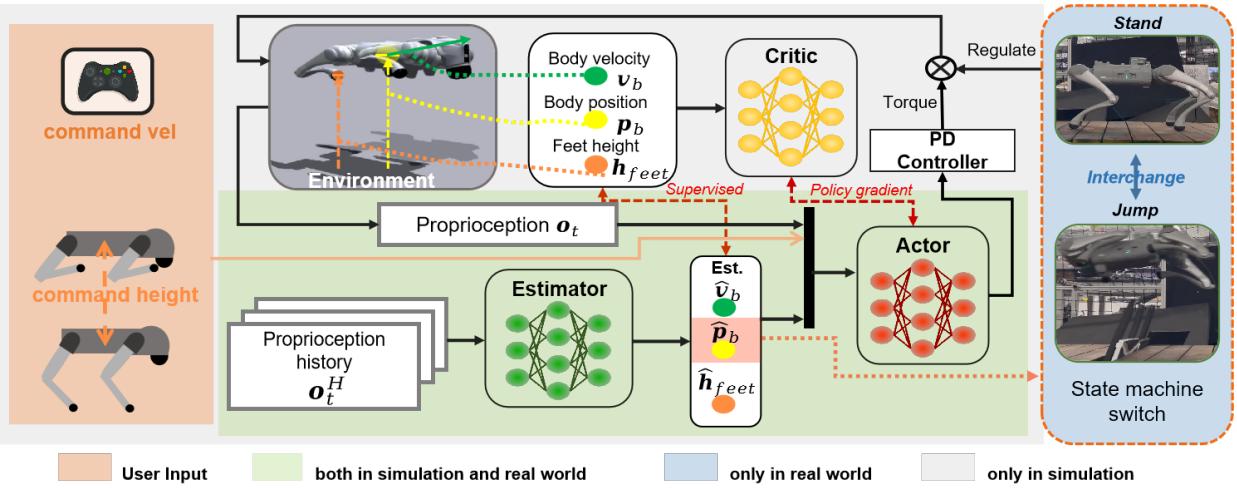

Fig. 2: The schematic of the proposed jumping framework. The orange block represents the user input module consisting of the commanded velocity $\boldsymbol{v}_t^{cmd}$, and the commanded height $\boldsymbol{h}^{cmd}$, which exist in both simulation and real-world deployment. The green blocks represent the modules that exist in both simulation training and real-world deployment. The value net is in the grey block, which exists only in simulation training. The value net and policy net in PPO algorithm, together with the Estimator net that encodes the physical information, are updated concurrently. The blue block is the state machines switch part, which occurs only in the low-level control loop. The $\hat{\boldsymbol{p}}_b$ represents the estimated CoM positions of the robot which will be needed in both actor net and low-level control loop.

body velocity $\tilde{\boldsymbol{v}}_t \in \mathbb{R}^3$. The input vector $\boldsymbol{s}_t$ of the value net can be defined as follows:

$$\boldsymbol{s}_t = [\tilde{\boldsymbol{P}}_{CoM}, \tilde{\boldsymbol{H}}_{feet}, \tilde{\boldsymbol{v}}_t, \boldsymbol{o}_t, \boldsymbol{h}_t], \tag{3}$$

Height-estimator Net: Since the accurate body position and velocity cannot be directly acquired through the robot's IMU, the height-estimator net aims to process the history of observations $\boldsymbol{o}_t^H$ and predict some critical features, including the estimated robot CoM positions $\hat{\boldsymbol{P}}_{CoM}$, the estimated feet height $\hat{\boldsymbol{H}}_{feet}$, and the estimated velocity $\hat{\boldsymbol{v}}_t$). The height-estimator net is not only optimized by the PPO algorithm through policy gradient $loss_{policy}$, but also trained with supervised learning to reduce the Mean Squared Error (MSE), a regression loss $loss_{reg}$, between the estimation and the corresponding ground-truth values, as follows:

$$\begin{aligned} loss_{reg} &= MSE(\hat{\boldsymbol{P}}_{CoM}, \tilde{\boldsymbol{P}}_{CoM}) \\ &\quad + MSE(\hat{\boldsymbol{H}}_{feet}, \tilde{\boldsymbol{H}}_{feet}) + MSE(\hat{\boldsymbol{v}}_t, \tilde{\boldsymbol{v}}_t) \\ Loss &= \beta \cdot loss_{reg} + (1-\beta) \cdot loss_{policy} \end{aligned} \tag{4}$$

where β is a hyperparameter with a range [0,1] ($\beta = 0.5$ in our case).

4) Reward Design: The rewards are designed to track the commanded height, horizontal linear velocity, and angular velocity, while regulating the robot gesture, penalizing large changes in actions and avoiding collisions as shown in Table I. The versatile jump (forward, side, and yaw-turning jump) is realized by the commanded velocity inputs. To encourage the robot to jump according to a certain height, two reward terms are designed, one sparse reward and one dense reward. The dense term tracks the commanded heights. The sparse term counts the agents that successfully jump to the acceptable height range (command height $\pm\, 0.04\ m$), as follows:

$$n_j = \begin{cases} 1, \text{ if agent j jump successfully} \\ 0, \text{ else} \end{cases} \tag{5}$$

TABLE I: REWARD TERMS

Term	Reward	Equation	Weight
Task	Height tracking	$e^{-20(z^*-z)^2}$	1.0
	Successful jump	n_{jump}	20
	Lin. Vel tracking	$e^{-4\|\|v_{xy}^*-v_{xy}\|\|^2}$	1.5
	Ang. Vel tracking	$e^{-4(\omega_{yaw}^*-\omega_{yaw})^2}$	0.6
Pose	Orientation	$\|\|q_{xy}\|\|^2$	-0.8
	Joint angle (aerial)	$\sum_{j=0}^{12}\|\|q_j-\hat{q}_j^{air}\|\|$	-0.4
	Joint angle (prelanding)	$\sum_{j=0}^{12}\|\|q_j-\hat{q}_j^{pre}\|\|$	-0.6
	Joint angle (landing)	$\sum_{j=0}^{12}\|\|q_j-\hat{q}_j^{ground}\|\|$	-0.12
Safety	Collisions	$-n_{collision}$	-1.0
Smoothness	Joint torque	$\|\|\tau\|\|^2$	$-e^{-5}$
	Action rate	$\|\|a_t-a_{t-1}\|\|^2$	-0.01
	Joint accerlerations	$\|\|\ddot{q}\|\|^2$	$-2.5e^{-7}$

Analytical IK Algorithm To accurately control the robot's gestures, especially thigh and calf angles in the air, we design a pose reward that regulates the joint angle directly according to the predefined foot height. Unlike numerical IK, analytical IK is chosen to accelerate the computation process. Based on the theory in [24], the algorithm will compute the joints' angles $\boldsymbol{q} \in \mathbb{R}^{12}$ given the four feet position $\boldsymbol{p}_f \in \mathbb{R}^{12}$ under the body frame, and then calculate the invertible Jacobian matrix that maps the feet velocity $\boldsymbol{v}_f \in \mathbb{R}^{12}$ to its joints' velocities $\dot{\boldsymbol{q}}$, according to the following equation,

$$\boldsymbol{q}, \dot{\boldsymbol{q}} = IK(\boldsymbol{p}_f, \boldsymbol{v}_f). \tag{6}$$

The detailed meaning of the geometrical variations and deduction process is shown in the supplementary materials [1]. In our task, three joints' angles $\boldsymbol{q}^{air}$, $\boldsymbol{q}^{pre}$, and $\boldsymbol{q}^{ground}$ will be calculated using the foot position in each phase, as shown in Table I.

[1]Please refer to the supplementary materials here: https://anonymous.4open.science/r/Supplementary_Materials_OMNINet/Supplementary_Materials_OmniNET.pdf

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

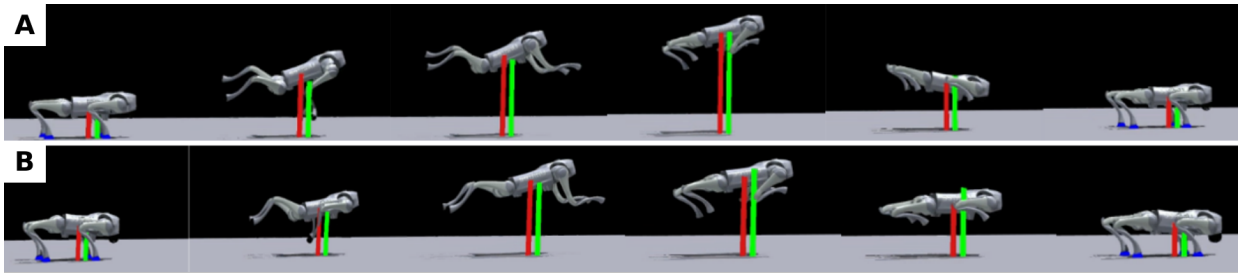

Fig. 3: Robot jump with the different height commands to track in the simulation environments, while the forward commands are the same. The red pillar represents the real height of the robot's COM height, while the green pillar represents the estimated height generated by the height estimator. Forward-jump with command height of 0.66 m (A) and 0.52 m (B)

C. Controllers Switch

In the low-level control loop, the estimated height h^{est} is used to select the switch point after the robot has landed. Each jumping process for the robot lasts approximately 0.2 seconds (about 40 control steps with the control frequency of 200 Hz). A step counter is designed to track control steps. And a threshold is predefined, which is visualized as the switch zone in Fig. 4. Additionally, if the estimated heights reach the inflection point during this switch zone, the system determines this point as the switch point. A state machine switch will occur at that point, from the jumping state to the standing state. Then, the state machine can be changed back to the jumping controller according to the user input.

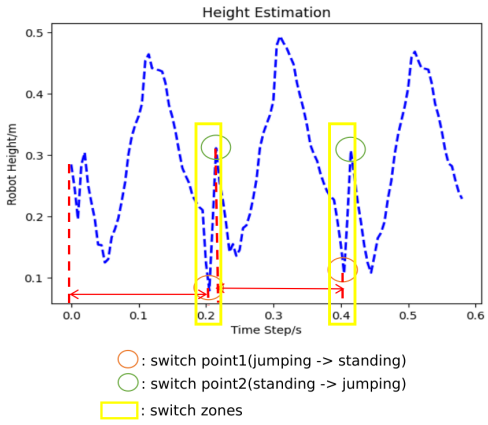

Fig. 4: The switch points shown in terms of the estimated robot height

III. Experimental Results and Discussion

A. Implementation details

1) Simulation: We train 4096 agents in parallel on the Isaac Gym simulator [25] for 5000 episodes. An episode is terminated and reset under specific termination criteria, which include the robot's trunk, thigh, calf, or hip collisions with the ground. We use both Unitree Go2 and Aliengo quadruped models to test our algorithm, which shows the generality of our algorithm. The actor network and the value net are both 4-layer multi-layer perception (MLP) and the architecture of these two MLPs are [512, 256, 128, 12] and [512, 256, 128, 1], respectively. Both of them use Exponential Linear Units (ELU) as the activation between hidden layers. The input dimensions of the actor net and the value net is 56 and 243, respectively. The architecture of the height-estimator is [258, 128, 10], and it uses the Rectified Linear Unit (RELU) as the activation. The input dimension of the height-estimator net is 920. Some parameters of the environment are randomized at the initialization stage of training, as shown in Table II. The entire training is performed on a PC with a single NVIDIA RTX 4070 GPU for approximately an hour and a half.

TABLE II: The Randomization Range of Parameters

Parameters	Range	Unit
Payload	[-5.0, 5.0]	kg
Frictions coefficient	[0.2, 1.25)	-
Center of mass shift	[-0.05, 0.05)	m
System delay	[0, 4]	ms
Motor strength factor	[0.9, 1.1]	N·m

2) Hardware Deployment: The real hardware experiments are conducted on both Unitree Go2 and Aliengo quadruped robots. We choose Go2 to analyze. The experiments carried out on Aliengo will be performed in the video attached to this paper. Go2 is the robot with 12 actuated degrees of freedom and 15 kg weight. In our work, the nominal K_p is set as 40.0, and K_d as 1.2 for all joints on each leg.

B. Analysis of the Height Estimator

To validate the estimator's capability to approximate the CoM height, a series of up-jump experiments with different commanded heights are conducted, both in simulation and real-world deployment. Fig. 3 compares the predicted heights with the real CoM height online in the simulation. The red and green pillars always stay at similar heights during the whole jumping process.

To further illustrate the prediction accuracy quantitatively, we conduct two different-height real-robot up-jump experiments under VICON and record the estimator output values and VICON's localization information respectively. The results are shown in Fig. 5. Through these figures, the results indicate that the height estimator is able to predict the robot's CoM height h^{est} accurately by encoding the proprioception history $\boldsymbol{o}_t^H$. More analysis of about the height estimator can be found in the supplementary materials.

C. Jumping Analysis

1) Gesture Adjustment in the air: To better demonstrate the robot's ability to adjust its gesture in the air, we conducted a

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

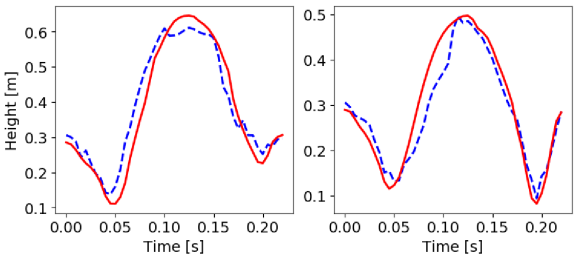

Fig. 5: Comparisons between the estimated and the real height of two experiments using the same jumping policy but different command heights. The red solid line represents the real height and the blue dotted line represents the output of the height estimator. Up-jump with command height of 0.66 m (Left) and 0.52 m (Right).

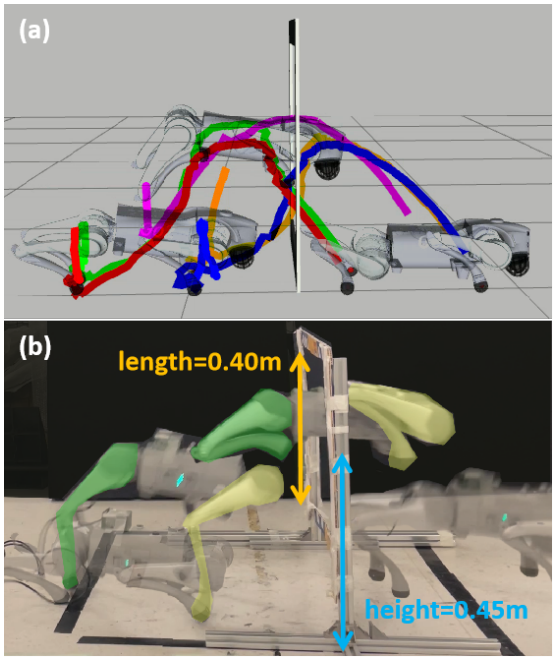

Fig. 6: (a) The simulation experiment is conducted using the data collected from the real-robot experiment. Using RViz, the whole frame traversing process is replayed, and visualized. Five trajectories are drawn in the picture, indicating the trajectories of CoM, and four feet respectively. (b) The center of the frame is 0.45 m in height. The length of the frame is 0.4 m. The front and rear legs are highlighted with yellow and green colors respectively.

frame-traversing experiment with the robot. The robot needs to pull its legs close to the trunk in the air, in order to jump through the frame without collision. Because the frame length is designed according to the length of Go2 leg's length which is 0.4 m according to the Unitree official document [26]. As described in Sec. II-B4, the joint angles are regulated in the flight phase using a feet gesture reward that is the joint angle reward in the air shown in Table I. Fig. 6(a) and Fig. 6(b) effectively demonstrate the influence of this reward on the robot's gesture control.

In the take-off phase, the robot needs to stretch its legs to push its body into the air. After four feet take off from the ground, the controller is trained to pull back its leg to the trunk fast. Fig. 6(a) reveals that the planned trajectory of the feet's end-effectors will circumvent the frame edge successfully. The system coordinates all joints swiftly in real time to generate feasible whole-body trajectories, as visualized in Fig. 6(b).

2) Versatile Jump Performance and Landing on challenging terrains: In order to further demonstrate our method's omnidirectional jumping ability, several experiments are conducted, including side jumps, rotation jumps, as well as compound directional jumps. The combinations of directional movements show that the robot is capable of simultaneously tracking the linear, and angular commands.

To realize a safe and stable landing, we designed a pre-landing reward, as shown in Table I, to regulate the robot's gesture before landing. The robot learns to extend its legs before making contact with the ground. After the feet touch the ground, the robot gradually reduces the length of its legs to gradually absorb the impact of landing, which is a soft landing behavior. And then the robot switches to the standing gesture stably. We have conducted the experiments on the rough terrains to thoroughly showcase that this landing mechanism provides a stable and safe landing behavior. Our framework ensures that the robot maintains balance when landing on the uneven surfaces, while also taking off well from uneven terrain and maintaining a good aerial posture.

The results of both omnidirectional jumping and complex-terrain landing experiments can be found in the supplementary videos and supplementary materials.

3) Torque Analysis: We evaluated the policy's safety and stability on a highly challenging task which is a 270-degree yaw-turning jump as depicted in Fig. 7(a), and recorded the quantitative results of motor output, both angles and torques of 12 joints. The robot is able to finish this challenging task in a well-coordinated manner with proper torque and satisfying the joint angle limits, as depicted in Fig. 7. In Fig. 7(b), the plots are separated into several color zones indicating different phases. The torque and angle of each leg joint change following the same pattern of variation. One noticeable phenomenon is that the peak torque occurs at the take-off phase. The process of the torque increasing and decreasing is quite steep. In the flight phase, the torque remains higher than zero mainly because the robot needs to adjust its gesture in the air.

The gesture change of the robot can be seen in Fig. 7(a). When the robot enters the landing phase, it needs to first go through a buffering phase during which the joint angle and torque remain the same as that in the flight phase. When the CoM reaches the switch point, the joints begin to generate a significant torque, prompting the robot to return to the standing phase and prepare for the next jumping phase. Therefore, the torque peak of the landing phase occurs simultaneously with the joint angles' change. According to the official documents of Unitree Robotics, the torque limits for hip, thigh, and calf joints are 23.7 $N \cdot m$, 23.7 $N \cdot m$, and 45.43 $N \cdot m$ respectively. The torque remains within a reasonable range during the jumping process.

D. Scalability Analysis

Since velocity commands are used to trigger the directional movement in our framework, we explored our framework's enhanced transferability across different gaits and the realization

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

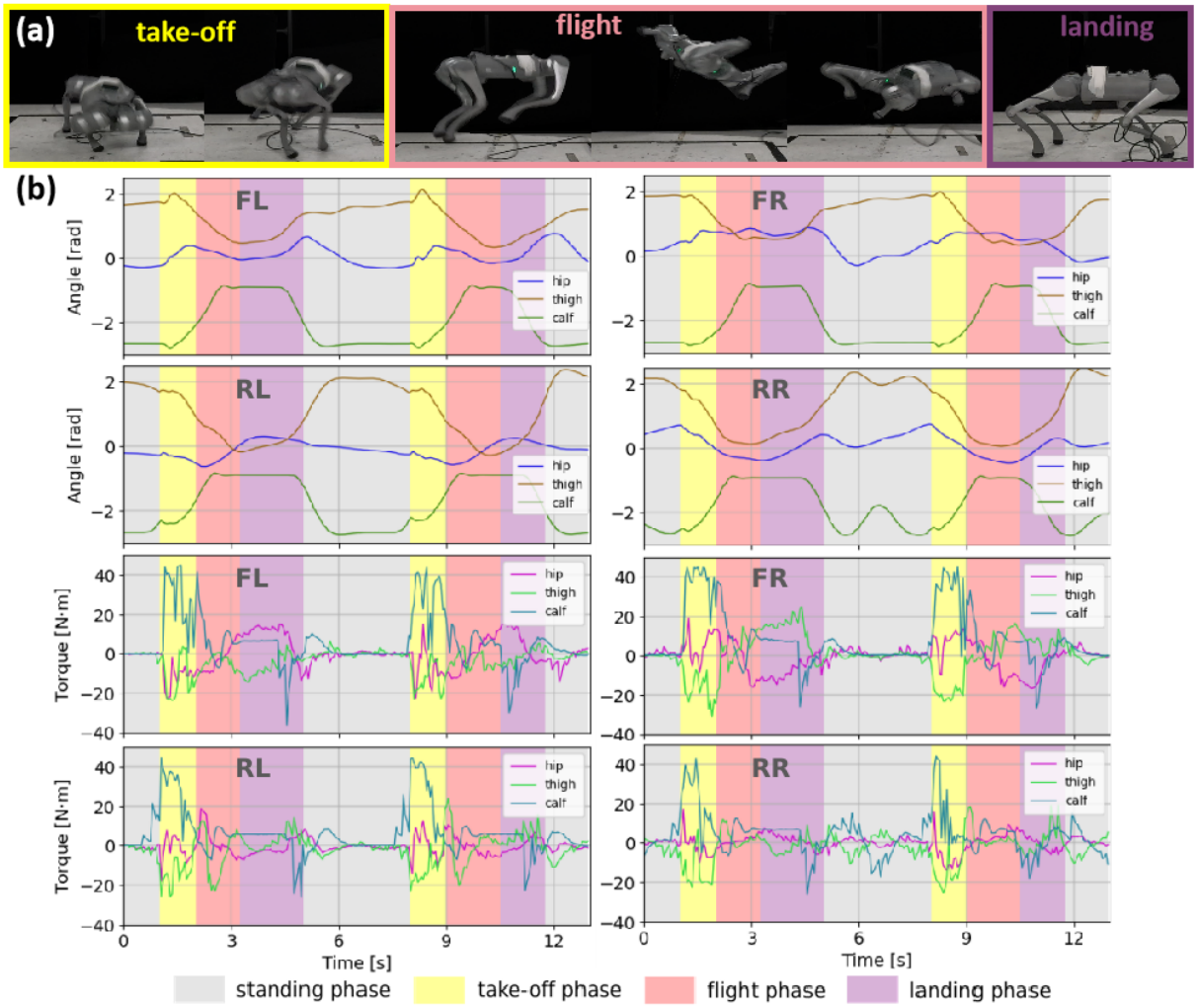

Fig. 7: (a) The image depicts the robot making one 270-degree yaw-turning jump, starting from the pre-jumping phase to the landing phase. (b) Hardware quantitative result for the 270-degree yaw-turning jump. We show the angles and torque changes of 4 legs respectively during two continuous jumps. The "FL", "FR", "RL", and "RR" in each sub-figures represent the front left, front right, rear left, and rear right legs, respectively. The standing, take-off, flight, and landing phases are indicated by the grey, yellow, red, and purple-shaded regions, respectively.

of more dynamic movements. Using the same reinforcement learning framework with different reward terms (eliminating the jump reward and the height tracking reward), we are able to train trotting policies that share the same command inputs as the jumping policy. To explore the potential of our framework in accommodating different gaits, we combined one trotting policy and jumping policy together. The estimated height h_{est} is the key for the controller switch in the low-level control loop. When $h_{est} < 0.4$, a trotting policy will be loaded. When $h_{est} \geq 0.4$, the jumping controller will be loaded. We have conducted the walking-jump experiments in both simulation environment and real-world deployment, as shown in Fig. 8. The experiments showcase that our framework can empower the robot's movement to maintain continuity during gait transitions.

E. Ablation Studies

Fig. 9 provides a comprehensive analysis of the performance of four different observation spaces. We categorize observation spaces into four distinct types, each incorporating different sensor inputs, and evaluate these policies' performance through successful jump reward and linear velocity tracking reward.

As we can see in the Fig. 9, different observation spaces yielded distinct reward outputs. The estimation of CoM position, feet end-effector height, and the robot's linear velocity has a significantly positive effect on enabling the robot to successfully achieve specified jump heights and accurately track linear velocity commands. We can conclude that the robot can achieve better performance when provided with estimation of both height and velocity information, than only with estimation of either height or velocity.

F. Comparison Studies

We have compared the jumping capabilities across different methods, as shown in Table III. Our methods can not only achieve the longest relative forward jump distance, but also exhibit the most comprehensive capabilities among all the existing jumping controllers.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

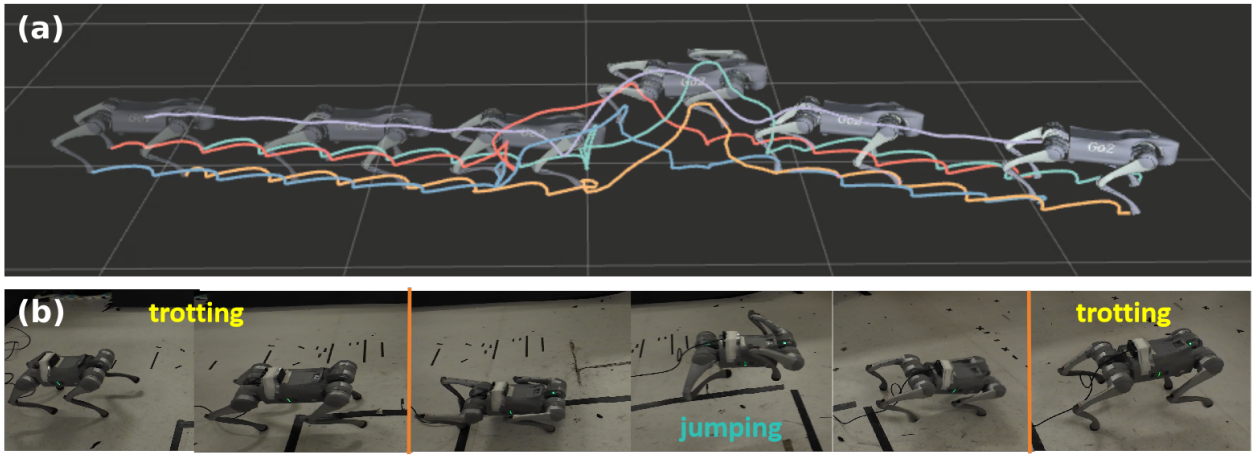

Fig. 8: (a) The trot-jump switch experiment in the simulation (Gazebo). (b) The trot-jump switch experiment in the real-world deployment.

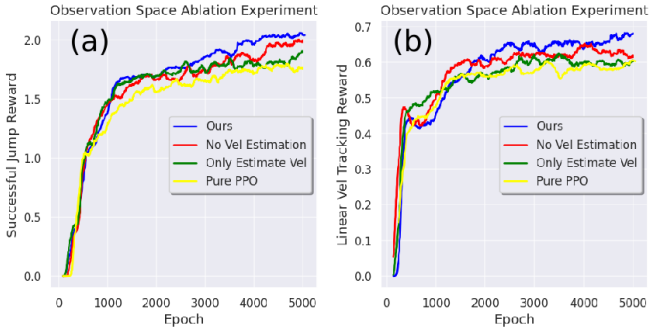

Fig. 9: All the policies are trained for a total of 5000 episodes similarly. Different observation spaces are designed as following: 1. **Ours**: $[\hat{\boldsymbol{P}}_{CoM}, \hat{\boldsymbol{H}}_{feet}, \hat{\boldsymbol{v}}_{\boldsymbol{t}}, \boldsymbol{o}_t]$; 2. **No Vel Estimation**: $[\hat{\boldsymbol{P}}_{CoM}, \hat{\boldsymbol{H}}_{feet}, \boldsymbol{o}_t]$; 3. **Only Estimate Vel**: $[\hat{v}_t, \boldsymbol{o}_t]$; 4. **Without Encoder (Pure PPO)**: $[\boldsymbol{o}_t,]$. **(a)**: The y-axis represents the value of the successful jump reward. **(b)**: The y-axis represents the value of the linear velocity tracking reward.

Methods (Robots)	Jump Length	Versatile Jump	Walking+Jump
[9] (Mini Cheetah)	0.7	✗	✓
[20] (Unitree-Go1)	1.2	✓	✗
[27] (Unitree-Go1)	1.4	✗	✓
[7] (Unitree-A1)	1.5	✓	✗
[19] (Unitree-Go1)	1.5	✗	✓
Ours (Unitree-Go2)	1.6	✓	✓

TABLE III: The first column indicates the different jumping controllers and the corresponding hardware platforms. The second column, Jump Length represents the normalized forward jump length with respect to quadruped's longest dimension which is its body length. The third column, Versatile Jump represents the controllers' ability in realizing omnidirectional jumps such as side jump, diagonal jump, etc. The last column indicates each framework's ability in accommodating to the different tasks.

To further demonstrate the effectiveness of our algorithm in terms of jumping control, we evaluate the robot performance quantitatively on three different DRL algorithms (including Extreme Parkour [7], Curriculum-based [19] and Ours) in a 0.65 m forward-jump task operating on Go2. Five indicators are chosen to determine the robot's performance: Maximum jumping height, maximum jumping velocity, minimum feet height under the body frame, aerial phase length, and change in torque. Five experiments for each algorithm are conducted. We recorded the average values and standard deviations (the value inside the parentheses) of four other metrics, excluding torque, in Table IV.

Our algorithm outperforms the other two algorithms in both maximum jumping velocity and maximum jump height, indicating more agile jumping behaviors with larger momentum. Moreover, our methods have smaller feet height under the body frame, which means the legs can be pulled closer to the trunk. The design of IK reward in Sec. II-B4 guarantees good control of the robot's gesture in the air. Since our methods allows the robot to achieve greater jump heights, it is understandable that the flight time is also longer.

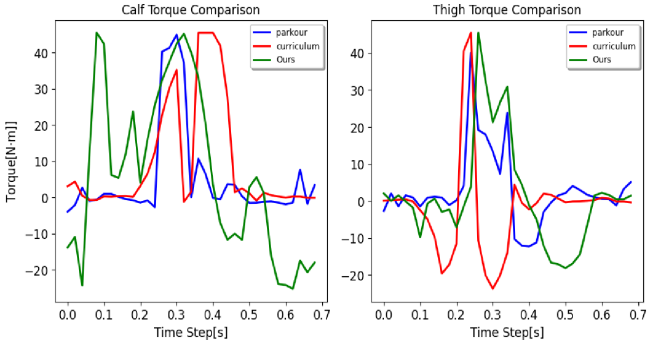

Fig. 10: Torques comparisons between Extreme Parkour (blue), Curriculum-based jump (red) and our work (green)

In Fig. 10, the calf and thigh torques of the front left leg are recorded, since calf and thigh play more important roles in the jumping process. Although the torque peaks of these three algorithms are similar, our method can produce a larger torque range, especially on the calf joint. This helps to explain the higher agility of our controller.

IV. CONCLUSIONS

Our work proposed a height-aware end-to-end DRL-based framework that can achieve omnidirectional and height-tracking jumps in quadruped robots. Although jumping is com-

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE IV: Evaluation of Jumping Performance: mean and standard deviation

Controller	Max Height[m]	Max Velocity[m/s]	Min Feet Height[m]	Aerial Time[s]
Parkour	0.3311(0.0110)	2.056(0.0366)	-0.1724(0.0003)	0.548(0.0299)
Curriculum	0.4347(0.0245)	1.940(0.2830)	-0.1023(0.0078)	0.413(0.0189)
Ours	**0.6597**(0.0039)	**3.083**(0.0471)	**-0.0501**(0.0005)	**0.585**(0.0087)

mon in nature, it's challenging to reproduce it on quadruped robots. The jumping task involves long aerial phases, and large impact impulse during the takeoff and landing phases, which places high demands on the robot's stability control and gesture adjustment. Our framework included a height estimation module and certain reward terms to assist jumping and landing. Thanks to these designs, our methods can achieve highly agile movements such as forward long jump, large yaw-turning jump, and omnidirectional jump. Moreover, our algorithm has fewer reward terms and only needs single-stage training, which simplifies and facilitates the training process. We further explore our framework's scalability to different gaits. We validate our method's performance by conducting several comparison experiments which indicate that our method not only has better forward jump ability, but it also has more comprehensive jumping abilities.

REFERENCES

[1] L. Yue, Z. Song, J. Dong, Z. Li, H. Zhang, L. Zhang, X. Zeng, K. Sreenath, and Y.-h. Liu, "Online omnidirectional jumping trajectory planning for quadrupedal robots on uneven terrains," *arXiv preprint arXiv:2411.04494*, 2024.

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

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

[4] H. W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *2015 Robotics: Science and Systems Conference, RSS 2015*. MIT Press Journals, 2015.

[5] Z. He, J. Wu, J. Zhang, S. Zhang, Y. Shi, H. Liu, L. Sun, Y. Su, and X. Leng, "Cdm-mpc: An integrated dynamic planning and control framework for bipedal robots jumping," *IEEE Robotics and Automation Letters*, 2024.

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

[7] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 443–11 450.

[8] Z. Zhuang, Z. Fu, J. Wang, C. G. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 73–92.

[9] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. bae Kim, and P. Agrawal, "Learning to jump from pixels," in *Conference on Robot Learning*. PMLR, 2022, pp. 1025–1034.

[10] G. Bellegarda, C. Nguyen, and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 182, p. 104799, 2024.

[11] R. Grandia, F. Farshidian, E. Knoop, C. Schumacher, M. Hutter, and M. Bächer, "Doc: Differentiable optimal control for retargeting motions onto legged robots," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 1–14, 2023.

[12] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.

[13] A. Klipfel, N. Sontakke, R. Liu, and S. Ha, "Learning a single policy for diverse behaviors on a quadrupedal robot using scalable motion imitation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2768–2775.

[14] C. Zhang, J. Sheng, T. Li, H. Zhang, C. Zhou, Q. Zhu, R. Zhao, Y. Zhang, and L. Han, "Learning highly dynamic behaviors for quadrupedal robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9183–9189.

[15] L. M. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Learning and adapting agile locomotion skills by transferring experience," in *Robotics: Science and Systems*, 2023.

[16] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Robust and versatile bipedal jumping control through reinforcement learning," in *Robotics science and systems*. RSS, 2023.

[17] X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, and K. Sreenath, "Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2715–2722.

[18] X. Huang, Q. Liao, Y. Ni, Z. Li, L. Smith, S. Levine, X. B. Peng, and K. Sreenath, "Hilma-res: A general hierarchical framework via residual rl for combining quadrupedal locomotion and manipulation," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 9050–9057.

[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 Robotics Automation Magazine*, pp. 2–15, 2024.

[20] Y. Yang, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots, "Continuous versatile jumping using learned action residuals," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 770–782.

[21] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[23] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[24] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance GPU based physics simulation for robot learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=fgFBtYgJQX_

[26] X. Wang, "Unitree robotics," 2018. [Online]. Available: https://www.unitree.com

[27] Y. Yang, G. Shi, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots, "Cajun: Continuous adaptive jumping using a learned centroidal controller," in *Conference on Robot Learning*. PMLR, 2023, pp. 2791–2806.