Robotica (2026), **44**, pp. 1630–1651
doi:10.1017/S0263574726103634

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Continuous multi-skill motion generation for quadruped robots based on imitation–reinforcement learning

Chong Pi[1,†], Senwei Huang[1,†], Wei Li[1], Haoyu Zhao[2], Jianing Wu[1], Tianqi Qiu[3], Peng Xu[3], Lei Jiang[3] and Xiuli Zhang[1,†]

[1]School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, China
[2]Department of Bionic Robotics Technology, China North Artificial Intelligence & Innovation Research Institute, China
[3]Unmanned Center, China North Vehicle Research Institute, China
Corresponding author: Xiuli Zhang; Email: zhangxl@bjtu.edu.cn

Received: 30 July 2025 UTC; **Revised:** 30 April 2026 UTC; **Accepted:** 15 May 2026 UTC;
First published online: 13 July 2026 UTC

Keywords: quadruped robot; reinforcement learning; imitation learning; multi-skill learning

Abstract

The appeal of quadruped robots lies not only in their ability to mimic the diverse and agile locomotion of animals but also in their potential to integrate with human artistic expression to achieve complex multi-skill movements. Continuous multi-skill motion in quadruped robots requires the realization of diverse, continuous, and long-horizon behaviors, involving a broader state space and more complex motion generation and transitions. This presents significant challenges, including sparse rewards, incomplete data, long-horizon motion planning, and the design of fine-grained motion transitions. In this work, we categorize quadruped robot skills into three types: rhythmic motions, expressive motions, and high-dynamic motions and generate reference trajectories for each category using central pattern generators, animation design, and motion capture, respectively. We then design an asymmetric neural network architecture and employ an imitation–reinforcement learning algorithm to train policies for generating these three types of motions. By composing multiple motion skill trajectories, we avoid long-horizon motion planning; by leveraging reinforcement learning, we enable smooth and continuous skill transitions; and by introducing a two-stage reference state initialization curriculum, the robot is able to switch from arbitrary states to the target motion skill. Moreover, during training, the policy imitates only key characteristics of the reference motions rather than strictly tracking fixed trajectories, making it more robust. Finally, we achieve robust motion skill generation and seamless transitions on a quadruped robot equipped with a 6-DoF manipulator, validating the effectiveness and feasibility of the proposed multi-skill generation and transition method.

1. Introduction

1.1. Motivation

At present, quadruped robots have achieved the capability to adapt to complex outdoor terrains and accomplish diverse locomotion control tasks, providing strong technical support for their further real-world applications [1–4]. Among the various application scenarios of quadruped robots, skill performance represents a commercially valuable direction. However, existing demonstrations are mostly limited to the presentation of a single motion skill or to multiple similar skills, such as different gaits, resulting in performances lacking diversity. In contrast, heterogeneous multi-skill motions require the realization of continuous, diverse, and long-horizon behaviors, involving a broader state space and more complex processes for motion generation and transition. These characteristics pose significant challenges to existing methods.

[†]These authors contributed equally to this work.

© The Author(s), 2026. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

In addition, expert data containing multiple motion skills and complete motion characteristics are crucial for multi-skill imitation learning (IL) in quadruped robots. Incomplete expert data and insufficient motion details make it difficult for existing approaches to effectively learn and model the transitions between skills. As a result, skill transitions often lack continuity and responsiveness, making it difficult to produce visually appealing multi-skill performances. However, high-quality expert data that contain multiple skills are typically difficult to obtain. Generating long-horizon trajectories with multiple skills through dynamic modeling and trajectory optimization is often tedious, time-consuming, and extremely challenging, significantly increasing the data preparation cost in the early stage. Alternatively, combining trajectories of different skills lacks natural transition processes and requires carefully designed motion transitions. A more practical and cost-effective approach is to collect data by manually controlling robot animation in simulation or real environments. However, the resulting data often lack dynamical information, making them unsuitable for IL. Another approach is to obtain motion data of animals or humans through motion capture or video extraction and then map them to quadruped robots. Nevertheless, this approach suffers from heterogeneous mapping problem between animals and robots and also makes it difficult to design artistic motion skills.

Due to the challenges of long-horizon motion planning or delicate motion transition design, incomplete data, and sparse rewards, multi-skill motion generation remains a highly challenging problem in quadruped robot locomotion control. Methods that combine IL with reinforcement learning (RL) have demonstrated significant advantages, as they allow the policy to imitate reference motions while exploring the limits of control capabilities, thereby achieving more robust motion control. Therefore, this paper proposes a motion skill generation and continuous transition method based on imitation–reinforcement learning (IRL). Multiple independent policies are trained as motion controllers to enable long-horizon multi-skill motion in quadruped robots. Compared to a single unified policy, adopting multiple independent policies with policy switching offers significant advantages: each policy can focus on learning a single skill, thereby improving motion fidelity and stability; it avoids mode collapse and catastrophic forgetting caused by multi-task coupling; it reduces the complexity of policy learning; and it enables more flexible composition among dissimilar skills. Specifically, we learn motion skills by imitating reference motions to avoid complex reward function design; we combine motion trajectories of multiple skills to eliminate the need for long-horizon trajectory planning and optimization; we employ RL to improve the continuity and robustness of skill transitions; and we adopt a two-stage reference state initialization (RSI) curriculum to address the learning difficulties caused by incomplete expert data.

1.2. Contributions

We propose an IRL framework to enable continuous multi-skill motions for a quadruped robot equipped with a 6-DoF manipulator. A variety of skills – including rhythmic trotting, dynamic jumping, double-leg standing, torso swaying, single-leg swinging, double-leg swinging, and leg–arm dancing – are generated and seamlessly transitioned in both simulation and real-world environments, demonstrating robust and coordinated whole-body motion control capabilities. The main contributions of this work are summarized as follows:

- We propose three methods for skill trajectory generation based on central pattern generators (CPG), animation design, and motion capture, as well as a trajectory retargeting and optimization method for mapping quadruped animal motions to quadruped robots;
- We develop an IRL algorithm for multi-skill motion generation in quadruped robots, which learns diverse motion skills through imitation of reference motions, achieves continuous multi-skill transitions via trajectory composition and RL, and incorporates a self-recovery mechanism during training to enhance robustness;
- We design a two-stage curriculum based on RSI to enable continuous skill transitions from arbitrary states and propose an asymmetric actor–critic neural network architecture that allows the robot to generate similar and robust motions without requiring reference motion inputs during deployment.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

2. Related work

2.1. Skill control methods based on dynamic models

Traditional control methods, represented by model predictive control and whole-body control, generally follow a modeling–planning–control pipeline, in which motion control is achieved by optimizing and tracking trajectories based on dynamic models [5–8]. For generating motion skills in quadruped robots, the typical approach first specifies an initial state, then performs offline optimization of skill trajectories based on simplified or full-body dynamic models, and finally tracks these trajectories online in real time. This enables the execution of individual motion skills such as jumping, backflips, and handstands [9–16]. However, such methods are limited by the initial conditions assumed during optimization, making it difficult to achieve continuous multi-skill motions.

To enable continuous long-horizon multi-skill motions, one approach is to concatenate offline trajectories of multiple skills and introduce transition phases and intermediate gaits to ensure smooth switching. By tracking these trajectories online, robots can perform sequences such as stepping up and down stairs, lying down, turning, and crawling under obstacles [17]. However, this approach is generally limited to slow and low-dynamic motions and is not suitable for highly dynamic behaviors such as jumping. To address this limitation, online trajectory optimization methods have been proposed, which generate motion trajectories in real time based on the robot's current state, enabling continuous execution of dynamic motions such as jumping, backflips, and running jumps [18–24]. By predefining contact sequences, these methods can also achieve multi-gait locomotion [25–27], and by switching between wheeled and legged modes, multimodal locomotion can be realized [28]. Nevertheless, such approaches are typically limited to single or similar types of motion skills and cannot handle heterogeneous multi-skill motions.

In summary, traditional control methods rely on accurate motion planning and control to achieve individual motion skills, multi-gait locomotion, and similar multi-skill behaviors. Their performance depends heavily on model accuracy, trajectory quality, and real-time control capability. In real-world scenarios, model mismatches and invalid assumptions often lead to system failures, and these methods generally exhibit limited generalization and robustness when facing changes in tasks or environments. Therefore, multi-skill motion generation remains a highly challenging problem.

2.2. Skill learning methods based on neural networks

In recent years, intelligent learning methods such as RL and IL have been widely applied to generate robust motion skills for quadruped robots, leveraging the strong nonlinear approximation capabilities of neural networks [29–33]. End-to-end neural network policy controllers trained purely with RL do not require trajectory planning, optimization, or tracking and have enabled quadruped robots to perform tasks such as terrain adaptation, obstacle avoidance, and parkour [34–38]. However, learning complex motion skills with RL often suffers from sparse rewards or requires carefully designed reward functions and extensive parameter tuning. Poorly designed reward functions can prevent the agent from making meaningful learning progress for extended periods. IL offers an alternative approach by learning control policies directly from expert datasets, enabling quadruped robots to acquire motion skills with biomimetic or artistic characteristics [39–41]. Although IL avoids complex reward design, its performance heavily depends on the quality and completeness of expert data, and it generally lacks sufficient exploration and generalization capabilities.

Another line of research combines RL with IL to mitigate the need for complex reward design while improving robustness. For example, reference motion trajectories can be generated using animation tools, and imitation-based reward terms are incorporated into RL frameworks to enable the learning of various skills such as jumping, backflips, acrobatics, and martial arts [42–44]. In addition, the well-known adversarial motion priors approach imitates motion styles rather than strictly tracking fixed trajectories, resulting in more robust motion skills [45–47]. However, when expert datasets contain skills with varying levels of complexity, AMP often struggles to learn a unified policy that captures all skills.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

It may learn only a subset of skills or suffer from catastrophic forgetting during long training, tending to favor simpler or blended behaviors that may not meet practical requirements. To address these limitations, one potential approach is to manually separate different types of skills within the expert dataset and train independent networks for each, although this requires significant manual annotation and computational resources. In addition, some studies combine the strengths of traditional control methods and learning-based approaches to achieve complex motion skills for quadruped robots [48–52].

2.3. Continuous multi-skill motion and transitions

Although existing methods are capable of learning and generating complex motion skills, they often struggle to achieve continuous switching between multiple skills and lack smooth transition capabilities. Tu et al. [53,54] proposed a progressive adversarial self-imitation skill-switching method, which leverages high-quality historical trajectories to autonomously learn both skills and their transitions, enabling natural switching between walking and crawling, as well as between walking and bipedal locomotion in quadruped robots. Yu et al. [55] employed a high-level policy to generate weights for different gait policies, which are then combined to form multi-skill motions, achieving multiple gaits and smooth transitions on the Unitree A1 quadruped robot. Yang et al. [56] designed a network architecture that integrates CPG with multilayer perceptrons, and jointly trained the model using gradient-based learning, enabling strong multi-skill capabilities, including fall recovery and various quadrupedal gaits. However, this approach can only switch motions between gaits that are close in the action space. Watanabe et al. [57] trained a neural network via RL to encode reference motions into structured latent variables, enabling smooth transitions among multiple dance motions. However, their method requires interpolation of the latent variables during deployment to ensure smooth transitions at arbitrary time points. Moreover, strong periodicity assumptions are imposed in both motion representation and policy learning, limiting its applicability to more complex motion skills. Most approaches adopt the idea of adversarial IL, guiding the policy to spontaneously generate transitional motions between different expert datasets. However, due to the use of a single-policy multi-skill learning framework, this often leads to mode collapse and catastrophic forgetting. Consequently, although these works have made considerable progress, achieving long-horizon multi-skill motions for quadruped robots remains a challenging problem.

3. Methods

3.1. Framework overview

We construct an IRL framework, as illustrated in Figure 1, to enable the generation and transition of continuous and diverse motion skills for quadruped robots. The framework consists of three main components: skill planning, skill learning, and policy transfer. In the skill planning stage, a reference trajectory library is built, including rhythmic motions, stylistic motions, and high-dynamic motions. In the skill learning stage, an IRL algorithm is employed to train neural network policies that control the robot to imitate reference motions in a simulation environment. In the policy transfer stage, the learned policies are deployed to a more accurate dynamics simulation environment and real robot hardware, enabling the robot to execute the learned skills and achieve continuous transitions in real-world settings.

Most existing approaches tend to train a single neural network policy to generate multiple motion skills. However, for heterogeneous skills with significantly different state spaces, learning a unified policy is highly challenging. Therefore, we train independent neural network policies for each motion skill and achieve continuous transitions between skills through RL. This approach not only avoids long-horizon motion planning and training processes but also allows more flexible composition of multiple skills to realize diverse long-horizon behaviors. During training, an imitation-based reward mechanism is introduced to address the issue of sparse rewards and to maximize the similarity between generated motions and reference trajectories. In addition, a two-stage curriculum based on RSI is adopted to ensure

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

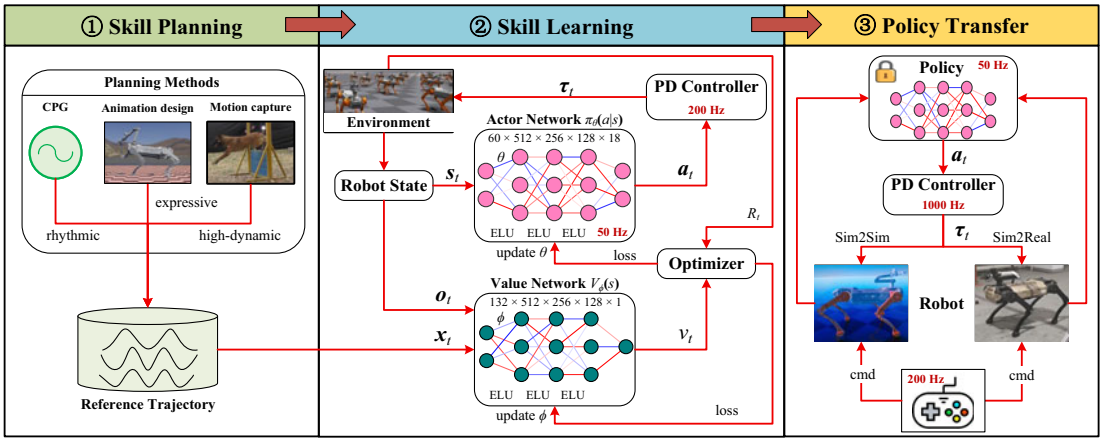


Figure 1. *An IRL framework for multi-skill learning in quadruped robots. The system consists of three key components: skill planning, skill learning, and policy transfer. We employ RL to train neural network policies, where the designed reference motion trajectories are used as imitation rewards and observation inputs to maximize the similarity between the generated motions and the reference motions. After deployment on the real robot, arbitrary and smooth transitions between motion skills can be achieved via policy switching.*

that the initial state distribution of the agent covers the entire state space of the reference motions, thereby improving learning efficiency.

3.2. Multi-skill motion planning method

To enable efficient learning of motion generation by imitating the characteristics of reference trajectories, a large amount of diverse trajectory data is required. We first categorize quadruped robot motions into three types based on trajectory characteristics: rhythmic motions, expressive motions, and high-dynamic motions and propose corresponding methods for acquiring these trajectories:

1. **Rhythmic motions:** These exhibit clear periodicity and consist of simple, repetitive trajectories, such as standing and trotting. These trajectories are generated using CPG.
2. **Expressive motions:** These are designed to be visually expressive and stylistically rich, characterized by complex and diverse trajectories, such as double-leg swinging and leg–arm dancing. These trajectories are generated using animation software.
3. **High-dynamic motions:** These are characterized by explosive and instantaneous movements with highly nonlinear trajectories, such as jumping and flipping. These trajectories are captured from quadrupedal animal motions via motion capture.

Each collected skill trajectory $x^{(i)}$ contains the following components: the position and linear velocity of the center of mass (CoM), the orientation and angular velocity of the base, the foot positions, the joint angles, and the joint angular velocities:

$$x^{(i)} = \left(\boldsymbol{p}_t, \boldsymbol{\theta}_t, \boldsymbol{v}_t, \boldsymbol{\omega}_t, \boldsymbol{p}_t^{\text{toe}}, \boldsymbol{q}_t, \dot{\boldsymbol{q}}_t\right)_{t=0}^{T} \tag{1}$$

where T is the trajectory length; $\boldsymbol{p}_t \in \mathbb{R}^{3\times1}$ and $\boldsymbol{v}_t \in \mathbb{R}^{3\times1}$ represent the CoM position and linear velocity; $\boldsymbol{\theta}_t \in \mathbb{R}^{4\times1}$ (quaternion) and $\boldsymbol{\omega}_t \in \mathbb{R}^{3\times1}$ represent the base orientation and angular velocity in the world frame; $\boldsymbol{p}_t^{\text{toe}} \in \mathbb{R}^{12\times1}$ is the foot positions in the body frame; and $\boldsymbol{q}_t \in \mathbb{R}^{18\times1}$, $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{18\times1}$ are the joint angles and velocities.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Each skill can be expanded into multiple variants with different frequency profiles through sampling and interpolation, thereby augmenting the motion data to form a multi-skill trajectory library containing N trajectories:

$$D = \{x^{(i)}\}_{i=1}^{N} \tag{2}$$

3.2.1. CPG-based planning of rhythmic skills

Given the coupled and periodic nature of rhythmic motions – both among limbs and between the limbs and torso – we adopt a CPG model inspired by the mechanisms of rhythmic motor control in animals to construct a flexible trajectory generator for rhythmic motions. CPGs are distributed oscillatory networks composed of multiple neurons that interact to generate self-sustained oscillations. These oscillations exhibit stable phase-locked relationships among neurons. Due to the plasticity of their synaptic connections, CPGs can produce a wide variety of connection patterns and output signals, enabling the generation of diverse rhythmic behaviors [58,59]. CPGs have been widely used in robotics to produce rhythmic gaits and have proven highly effective for designing rhythmic motion trajectories.

To construct a CPG network, it is crucial to select an appropriate oscillator that can generate periodic signals. The Hopf oscillator is characterized by a stable limit cycle in the state space, ensuring that, regardless of the non-zero initial condition, the oscillator converges to a periodic signal of consistent shape [60–62]. We adopt the Hopf oscillator to build the CPG network. The mathematical model of the Hopf oscillator is given by:

$$\begin{cases} \dot{x} = \alpha_x(\mu^2 - r^2)x - \omega y \\ \dot{y} = \alpha_y(\mu^2 - r^2)y + \omega x \\ \omega = \dfrac{\omega_{\mathbf{st}}}{e^{-by}+1} + \dfrac{\omega_{\mathbf{sw}}}{e^{by}+1} \\ r^2 = x^2 + y^2 \end{cases} \tag{3}$$

here, x and y are the output signals of the oscillator. α_x and α_y are the convergence coefficients for x and y, respectively, controlling how quickly the oscillator converges to the limit cycle. μ denotes the amplitude of the oscillator, and r is the radius of the limit cycle. ω represents the angular frequency of the oscillator, while ω_{st} and ω_{sw} are the angular frequencies for the stance and swing phases, corresponding to the proportion of stance time and swing time in a complete gait cycle. The constant b is the time integration factor, which determines the rate at which ω transitions between ω_{st} and ω_{sw}, ensuring a smooth variation of μ across different half-planes.

A phase variable $\phi \in [-\pi, \pi]$ is defined to describe the contact state of a leg with the ground: when $\phi \in [0, \pi]$, the foot is in the stance phase; when $\phi \in [-\pi, 0]$, it is in the swing phase, as illustrated in Figure 2(a). The periodic phase ϕ is modeled using the output of the Hopf oscillator:

$$\phi = \arctan 2(x, y) \tag{4}$$

As shown in Figure 2(b), a CPG network is constructed by coupling four Hopf oscillators to model the phase dynamics of the four legs of the quadruped robot:

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \begin{bmatrix} \alpha_x(\mu^2 - r^2) & -\omega \\ \omega & \alpha_y(\mu^2 - r^2) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + k \sum_{j \neq i} \boldsymbol{R}(\varphi_{ij}) \begin{bmatrix} x_j \\ y_j \end{bmatrix} \tag{5}$$

here, k is the coupling gain, and $\varphi_i^j = \phi_i - \phi_j$ represents the phase offset of oscillator i with respect to oscillator j, where $i, j \in \{1, 2, 3, 4\}$. The coupling matrix $\boldsymbol{R}(\varphi_{ij})$, which controls the interactions among oscillators and determines the gait characteristics of the quadruped robot, is defined as follows:

$$\boldsymbol{R}(\varphi_{ij}) = \begin{bmatrix} \cos(\varphi_{ij}) & -\sin(\varphi_{ij}) \\ \sin(\varphi_{ij}) & \cos(\varphi_{ij}) \end{bmatrix} \tag{6}$$

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Table I. *Parameter settings of the Hopf oscillator.*

Parameter	Symbol	Value
Convergence rate of x	α_x	10.0
Convergence rate of y	α_y	10.0
Amplitude of the oscillator	μ	1.0
Time integration constant	b	50.0
Coupling gain	k	1.0

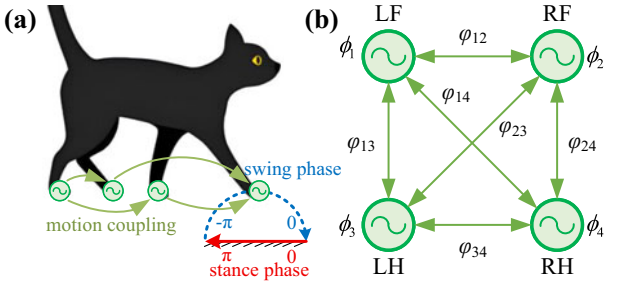


Figure 2. *The CPG network of a quadruped robot constructed by coupling four oscillators. (***a***) Foot phase. (***b***) Topology of the CPG network. LF, RF, LH, and RH denote the left front, right front, left hind, and right hind legs, respectively, with the oscillators numbered from 1 to 4.*

A phase parameter ρ is defined to represent the normalized phase progression of the foot during the stance or swing phase:

$$\rho = \begin{cases} -\dfrac{\phi}{\pi}, \phi \in [-\pi, 0) \\ \dfrac{\phi}{\pi}, \phi \in [0, \pi) \end{cases} \tag{7}$$

Following the polynomial-based foot trajectory generation model proposed in [52], the normalized phase progression parameter ρ is used as the input variable of a polynomial to generate foot trajectories. The gait period is set to 0.5 s, and the phase relationships among the four oscillators are configured as $[0, \pi, \pi, 0]$. Both stance and swing phase frequencies are set to 0.5. Other parameters for the Hopf oscillator are listed in Table I. Under these settings, the phase relationships and foot trajectories corresponding to the trot gait are illustrated in Figure 3. By modifying the coupling phase offsets between the four oscillators and adjusting the stance/swing frequencies, various rhythmic gaits such as walk, pace, and bound can be generated.

3.2.2. Animation-based planning of expressive skills

For expressive motions that require strong visual appeal, we leverage the intuitive 3D animation software Blender and the physics simulator Webots for trajectory design. Blender offers powerful animation authoring and rendering capabilities, allowing for fine-grained motion design and preview. Webots enables manual specification of robot poses and simulates the planned trajectories in a realistic physical environment. Our workflow for planning expressive motion trajectories is as follows: first, construct the robot's 3D model and kinematic structure; next, set the key postures of the motion as animation keyframes; then, generate the full motion sequence via linear or Bézier interpolation under kinematic constraints; finally, export the resulting trajectories for direct use in RL algorithm training. The key postures for the double-leg swinging and leg-arm dancing motions are illustrated in Figure 4.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

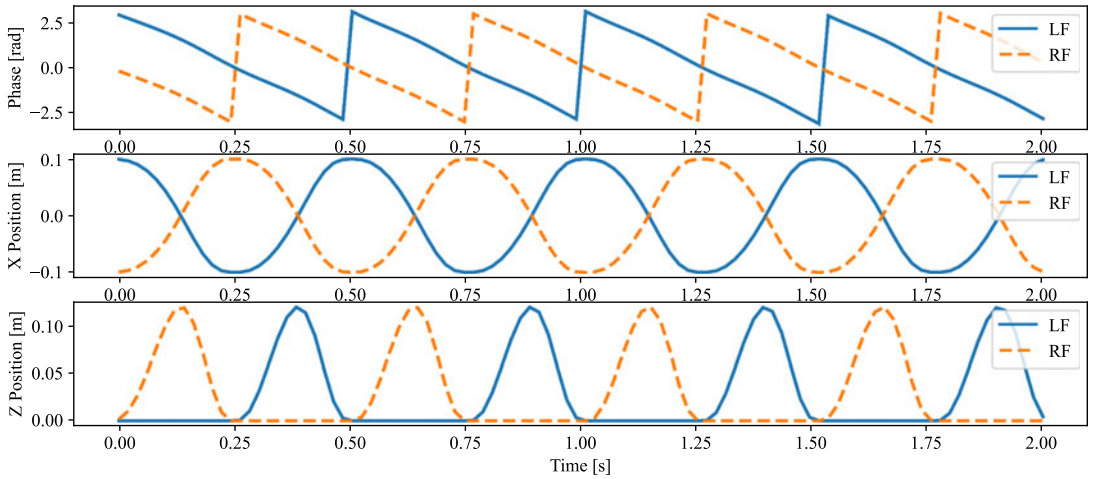


Figure 3. *Phase trajectories and foot trajectories in the body frame for the trot gait. Since the trajectories of diagonal legs are identical, only the trajectories of the two front legs are shown.*

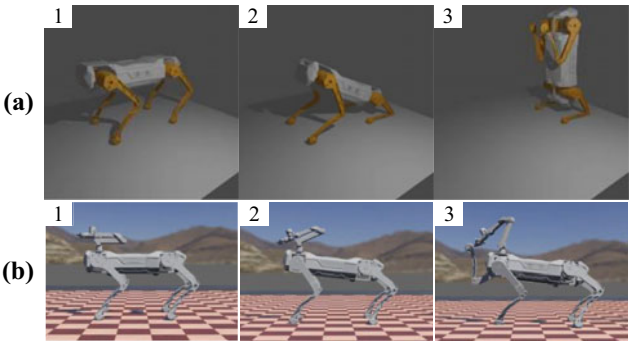


Figure 4. *Key pose design for expressive motions. (**a**) Key poses of the double-leg swinging motion are planned using Blender. From left to right: initial standing pose, intermediate squatting pose, and final upright pose. (**b**) Key poses of the leg-arm dancing motion are planned using Webots. From left to right: initial pose, intermediate pose, and final pose.*

3.2.3. MoCap-based planning of high-dynamic skills

Due to the complexity of high-dynamic motions, it is difficult to obtain corresponding trajectories using analytical trajectory equations or animation design methods. Given the structural similarity between quadruped animals and quadruped robots – and considering that animal motions are often elegant, natural, and fluid – they provide an ideal dataset for IL in quadruped robots. Therefore, we employ motion capture systems to record high-dynamic motion trajectories from quadruped animals and map them onto the robot. Specifically, we attach 28 optical markers to a highly agile Belgian Malinois to track its trajectory during jumping motions, as shown in Figure 5.

Since the morphology of the quadruped robot is not identical to that of the Belgian Malinois, we extract trajectory data from key anatomical landmarks – namely, the mid-spine, shoulders, hips, hands, and feet – to accurately imitate the dog's motions. Due to differences in body dimensions, the collected data must be mapped onto the robot. This mapping process consists of two steps: (i) Data retargeting, which rescales the motion trajectories to match the robot's dimensions; (ii) data optimization, which adjusts the trajectories to satisfy the kinematic constraints of the robot.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

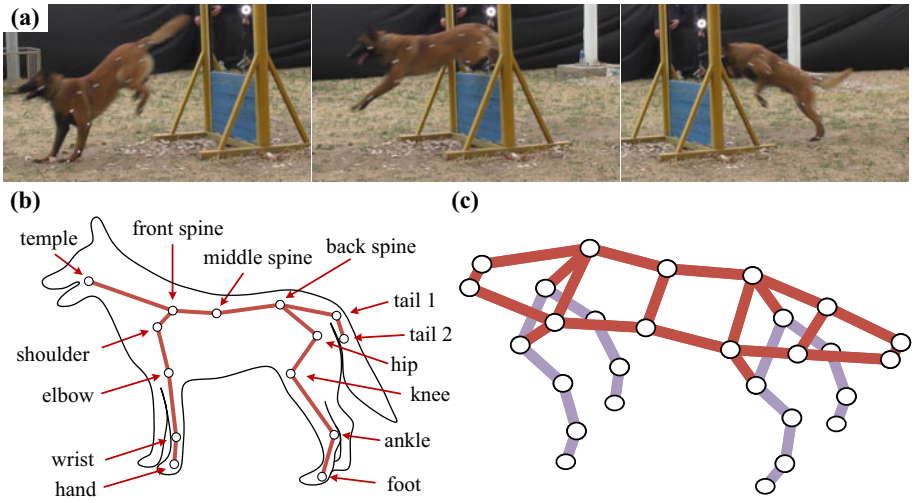


Figure 5. *Motion trajectory data collection of a Belgian Malinois during jumping. (**a**) Snapshots of the jumping process. (**b**) Planar rigid body structure with marker positions. (**c**) 3D stick figure with optical markers.*

Data retargeting: Based on the size ratio between the quadruped robot and the dog, we rescale and retarget the trajectories of the CoM, hip positions, and foot positions:

$$\begin{cases} \boldsymbol{p}' = \xi \boldsymbol{p}_{\text{origin}} \\ \hat{p}_{\text{hip},y} = p'_{\text{CoM},y} + 0.5 k w_{\text{hip}} \\ \hat{p}_{\text{foot},y} = p'_{\text{CoM},y} + 0.5 k w_{\text{foot}} \end{cases} \tag{8}$$

where $\boldsymbol{p}_{\text{origin}}$ is the original data, $\boldsymbol{p}'$ is the rescaled data, and $\xi = 2/3$ is the scaling factor. Given that the robot's torso is wider than the dog's, we further retarget the hip and foot positions in the lateral (y) direction using the rescaled CoM position. Specifically, let $\hat{p}_{\text{hip},y}$ and $\hat{p}_{\text{foot},y}$ denote the y-axis positions of the retargeted hips and feet, respectively. Let w_{hip} and w_{foot} represent the width between the robot's left and right hips and between its left and right feet. The leg selection indicator k is set to $k = 1$ for the left legs and $k = -1$ for the right legs.

Data optimization: The retargeted data are used as reference trajectories, and we formulate an optimization problem to compute the CoM position, base orientation, and leg joint angles that satisfy the robot's kinematic constraints.

We first solve for the CoM position and base orientation. The optimization objective is to minimize the discrepancy between the hip positions and the retargeted reference data, while ensuring full-body kinematic feasibility. The optimization problem is formulated as follows:

$$\begin{aligned} \arg\min_{\boldsymbol{x}} \quad & \sum_{i=0}^{n} \left\| \hat{\boldsymbol{p}}_{\text{hip},i} - p_{\text{hip},i}(\boldsymbol{x}) \right\|_{\boldsymbol{Q}_x}^2 \\ \text{s.t.} \quad & p_{\text{hip},i}(\boldsymbol{x}) = \boldsymbol{p}_{\text{CoM}} + \boldsymbol{R}_{\text{ZYX}} \boldsymbol{T}_{\text{hip},i} \\ & \underline{\boldsymbol{x}} \le \boldsymbol{x} \le \overline{\boldsymbol{x}} \end{aligned} \tag{9}$$

where the optimization variable $\boldsymbol{x} \in \mathbb{R}^{6\times 1}$ includes the CoM position $\boldsymbol{p}_{\text{CoM}} \in \mathbb{R}^{3\times 1}$ and base orientation angles $\boldsymbol{\Theta} \in \mathbb{R}^{3\times 1}$; $\hat{\boldsymbol{p}}_{\text{hip},i}$ represents the reference position of the i-th hip; $p_{\text{hip}}(\boldsymbol{x})$ is the function that computes the hip position from the optimization variable; $\boldsymbol{R}_{\text{ZYX}}$ denotes the Z-Y-X Euler angle transformation for the base orientation, and $\boldsymbol{T}_{\text{hip},i}$ is the translational offset from the CoM to the hip positions. n is the number of legs, $\boldsymbol{Q}_x$ is the weighting matrix for the tracking error, and $\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}$ are the lower and upper bounds for the optimization variables.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Then, we solve for the leg joint angles. The optimization objective is to minimize the errors between the joint angles and the foot positions with respect to the reference data, while satisfying the leg kinematic constraints and joint limits. An individual optimization problem is formulated for each leg as follows:

$$\begin{aligned} \arg\min_{\boldsymbol{q}_i} \quad & \left\|\hat{\boldsymbol{p}}_{\text{foot},i} - p_{\text{foot},i}(\boldsymbol{q}_i)\right\|_{\boldsymbol{Q}_f}^2 + \left\|\tilde{\boldsymbol{q}}_i - \boldsymbol{q}_i\right\|_{\boldsymbol{Q}_q}^2 \\ \text{s.t.} \quad & p_{\text{foot},i}(\boldsymbol{q}_i) = p_{\text{hip},i}(\boldsymbol{x}) + \boldsymbol{R}_{\text{ZYX}} \cdot FK(\boldsymbol{q}_i) \\ & \underline{\boldsymbol{q}}_i \le \boldsymbol{q}_i \le \overline{\boldsymbol{q}}_i \end{aligned} \tag{10}$$

where the optimization variable $\boldsymbol{q} \in \mathbb{R}^{3\times 1}$ denotes the three joint angles of the leg. $\hat{\boldsymbol{p}}_{\text{foot}}$ represents the reference foot position, $p_{\text{foot}}(\boldsymbol{q})$ is the function that computes the foot position from the optimization variable, and $FK(\boldsymbol{q}_i)$ represents the forward kinematics function of the leg. $\boldsymbol{Q}_x$ is the weight matrix for the foot position error. $\tilde{\boldsymbol{q}}$ denotes the nominal joint angles of the leg when the robot is in a standing posture, and $\boldsymbol{Q}_q$ is the weight matrix for the joint angle error, which prevents the optimizer from producing unreasonable or singular solutions. $\underline{\boldsymbol{q}}$ and $\overline{\boldsymbol{q}}$ define the lower and upper bounds of the feasible joint angles range for the actual robot.

3.3. Multi-skill motion learning method

3.3.1. Policy learning objective

After completing skill planning, we employ the RL algorithm to train neural network policies to imitate the reference motions. The motion imitation problem is formalized as a goal-conditioned RL task, where the policy is trained to mimic the reference motions in dataset D. The policy is defined as $\pi_\theta(a \mid s, g)$, representing a probability distribution over actions $a \in \boldsymbol{A}$ given the current state $s \in \boldsymbol{S}$ and task goal $g \in \boldsymbol{G}$, with θ denoting the parameters of the neural network. We train two rhythmic policies for balancing locomotion, multiple expressive policies for skill performances, and one high-dynamic policy for dynamic jumping to control the quadruped robot. Each policy is used to imitate a single reference motion, so the training of each policy is treated as an independent RL problem.

During the skill learning phase, the neural network parameters θ are first randomly initialized and then updated using the collected states, actions, and rewards. At each time step t, the agent interacts with the environment based on the policy: given the current environment state s_t and task goal g, an action $a_t \sim \pi_\theta(a_t|s_t, g)$ is sampled from the policy. After executing the action, the environment transitions to a new state s_{t+1} and returns a scalar reward:

$$r_t = r(s_t, a_t, s_{t+1}, g) \tag{11}$$

This reward reflects the quality of the state transition and is used to compute the loss for policy optimization. The specific definition of the reward function is provided in Table III. The action $a_t \in \mathbb{R}^{18\times 1}$ represents the offset relative to the joint's equilibrium angle and is fed into a proportional-derivative (PD) controller to compute the joint torques that drive the robot. The agent's objective is to learn the optimal neural network parameters θ to maximize the expected discounted cumulative reward:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim p(\tau|\pi,g)}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right] \tag{12}$$

where τ denotes a trajectory, consisting of a sequence of states, actions, and rewards: $\tau = \left\{(s_t, a_t, r_t)_{t=0}^{T-1}, s_T\right\}$, and T is the time horizon of the trajectory. $p(\tau|\pi, g)$ denotes the probability of the trajectory under the given policy and goal, where $p(\tau \mid \pi, g) = p(s_0)\, \Pi_{t=0}^{T} p(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t, g)$. $p(s_0)$ is the initial state distribution, and $p(s_{t+1} \mid s_t, a_t)$ denotes the environment dynamics. $\gamma \in [0, 1]$ is the discount factor.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Table II. *Inputs and noise levels of the actor network.*

State variable	Symbol	Noise level	Dimensions
Root projected gravity	$\boldsymbol{g}_t$	0.05	3
Root angular velocity	$\boldsymbol{\omega}_t$	0.2	3
Joint position deviations	$\Delta \boldsymbol{q}_t$	0.08	18
Joint velocities	$\dot{\boldsymbol{q}}_t$	1.5	18
Last actions	$\boldsymbol{a}_{t-1}$	0.0	18

Note: The noise follows a uniform distribution $p \sim \mathcal{U}(-\xi, \xi)$, where ξ denotes the noise level. This noise is added to the original observation data.

3.3.2. Design of asymmetric neural network architecture

Robot motion is represented as temporal trajectories in a high-dimensional state space, consisting of long sequences with multiple time steps and high-dimensional data. If reference motions are provided as inputs to the policy network, it significantly increases network complexity, leading to inefficient learning. Moreover, the policy network tends to "overfit" to the specific shape of the reference trajectory, attempting to imitate every frame of the reference motion. This results in poor generalization, making it difficult for the policy to learn smooth and natural motion transitions. Additionally, any external disturbance that causes deviation from the planned trajectory may cause the policy to fail entirely. Therefore, no reference motion information is used as input to the policy network. Instead, the tracking errors relative to the reference trajectory are fed into the critic network to evaluate how well the policy imitates the reference motion. In addition, robot control in the real world is essentially a partially observable Markov decision process, where certain state information that is easily accessible in simulation is unobservable in the real world. However, this missing information in reality can significantly facilitate policy training in simulation. Although most of the missing yet critical information can be recovered in the real world using state estimation algorithms, the estimation errors can increase the Simulation-to-Real (Sim2Real) transfer gap, making policy deployment on physical robots challenging.

To address this, we design an asymmetric Actor-Critic neural network architecture: the Actor network (policy network) relies only on accurately perceived proprioceptive information, while the Critic network has access to privileged information, such as torso linear velocity and trajectory tracking errors. Both the Actor and Critic networks are implemented as multi-layer perceptrons with ELU activation functions, with the detailed network structure shown in Figure 1. This asymmetric design allows the policy network to generate similar motion skills during deployment without any input of reference motion.

The state vector input to each Actor network is:

$$\boldsymbol{s}_t = (\boldsymbol{g}_t, \boldsymbol{\omega}_t, \Delta \boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{a}_{t-1}) \tag{13}$$

here, $\boldsymbol{g}_t$ denotes the projected gravity vector onto the torso frame, i.e., the projection of the gravity vector in the world frame onto the torso coordinate frame. $\Delta \boldsymbol{q}_t$ represents the deviation between the current joint angles and their corresponding equilibrium angles. $\boldsymbol{a}_{t-1}$ denotes the action output by the policy network at the previous time step.

The state vector input to each Critic network is

$$\boldsymbol{o}_t = (\boldsymbol{v}_t, \boldsymbol{\theta}_t, \boldsymbol{\omega}_t, \Delta \boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{a}_{t-1}, \Delta \boldsymbol{x}_t) \tag{14}$$

where $\Delta \boldsymbol{x}_t$ represents the deviation between the current robot state and the reference state.

In addition, to bridge the gap between simulation and the real world, realistic random noise is added to the proprioceptive inputs of the Actor network. The observation inputs and noise levels are listed in Table II.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Table III. *Definition of reward function.*

Category	Reward term	Definition	Weight
Imitation	Torso position	$\exp(-20.0 \cdot \|\boldsymbol{p} - \hat{\boldsymbol{p}}\|^2)$	5.0
	Torso orientation	$\exp(-200.0 \cdot \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|^2)$	10.0
	Linear velocity	$\exp(-2.0 \cdot \|\boldsymbol{v} - \hat{\boldsymbol{v}}\|^2)$	2.0
	Angular velocity	$\exp(-0.2 \cdot \|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}\|^2)$	2.0
	Leg joint positions	$\exp(-5.0 \cdot \|\boldsymbol{q}_{\text{leg}} - \hat{\boldsymbol{q}}_{\text{leg}}\|^2)$	5.0
	Arm joint positions	$\exp(-20.0 \cdot \|\boldsymbol{q}_{\text{arm}} - \hat{\boldsymbol{q}}_{\text{arm}}\|^2)$	10.0
	Leg joint velocities	$\exp(-0.1 \cdot \|\dot{\boldsymbol{q}}_{\text{leg}} - \hat{\dot{\boldsymbol{q}}}_{\text{leg}}\|^2)$	5.0
	Arm joint velocities	$\exp(-\|\dot{\boldsymbol{q}}_{\text{arm}} - \hat{\dot{\boldsymbol{q}}}_{\text{arm}}\|^2)$	10.0
	Toe positions	$\exp(-200.0 \cdot \|\boldsymbol{p}_{\text{toe}} - \hat{\boldsymbol{p}}_{\text{toe}}\|^2)$	10.0
Smoothness	Joint torques	$\|\boldsymbol{\tau}\|^2$	$-1.0 \cdot 10^{-5}$
	Joint torque rates	$\|\boldsymbol{\tau}_t - \boldsymbol{\tau}_{t-1}\|^2$	$-1.0 \cdot 10^{-6}$
	Joint accelerations	$\|\ddot{\boldsymbol{q}}\|^2$	$-5.0 \cdot 10^{-6}$
	Action smoothness	$\|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\|^2 + \|\boldsymbol{a}_t - 2\boldsymbol{a}_t + \boldsymbol{a}_{t-2}\|^2 + 0.01\|\boldsymbol{a}_t\|_1$	$-1.0 \cdot 10^{-2}$
	Feet contact forces	$\sum \min(\max(\|\boldsymbol{f}_j\| - f_{\max}, 0), F_{\text{clip}})$	$-2.0 \cdot 10^{-2}$
	Collisions	n_c	-10.0
	Joint position limits	$\sum_{q_i < \underline{q}_i} (q_i - \underline{q}_i) + \sum_{q_i > \bar{q}_i} (q_i - \bar{q}_i)$	-10.0

3.3.3. Design of imitation–smoothness reward function

The designed reward function encourages both accurate imitation of the specified reference motion and smoothness of the generated motion. Therefore, the reward function consists of two components: the skill imitation reward and the skill smoothness reward, formulated as follows:

$$r_t = r_t^{\text{imitation}} + r_t^{\text{smoothness}} \tag{15}$$

The detailed definitions of the reward function and weights of rhythmic trotting are listed in Table III. In the formulas, variables with a hat (^) denote reference state quantities. The variables are defined as follows: $\boldsymbol{\tau}$ denotes the vector of all joint torques; $\boldsymbol{f}_j$ represents the foot contact force vector of the j-th leg at time t; f_{max} is the maximum allowable contact force; F_{clip} limits the maximum penalty value to prevent the total reward from becoming negative and overwhelming the incentive provided by imitation; $\bar{q}_i$ and $\underline{q}_i$ indicate the lower and upper joint limits for the i-th joint; n_c counts the number of collisions, including self-collisions and collisions between the calf, thigh, torso, and manipulator with the ground. However, during the training of the double-leg swinging tasks, collisions involving the calf and thigh with the ground are ignored. The weights in Table III are determined through an experience-driven iterative tuning process: initial values are first set, and then each weight is gradually adjusted through multiple rounds of training experiments. For different skills, we adopt a unified reward function structure and share most of the weight parameters, with only a few skill-specific terms being fine-tuned according to their distinct characteristics.

The skill imitation reward is designed to ensure that the policy accurately imitates the reference motion. It is formulated as a measure of discrepancy between the current state and the target state, defined using a Gaussian kernel function:

$$r_t^{\text{imitation}} = e^{-\lambda \|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\|^2} \tag{16}$$

where λ is a hyperparameter of the Gaussian kernel, and $\hat{\boldsymbol{x}}_t$ and $\boldsymbol{x}_t$ represent the reference and actual motion states at time t, respectively.

The skill smoothness reward is primarily designed to encourage the policy to generate smooth action trajectories, ensuring continuous and seamless transitions between motions while reducing mechanical jitter and energy consumption. We design smoothness rewards at both the policy output level and the

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Table IV. *PPO algorithm hyperparameters.*

Entry	Value
Step time seconds	0.005
Learning rate	0.001
Entropy coef	0.01
Mini batch size	30,000
Max iterations	10,000

physical manifestation level. At the policy output level, we introduce an action smoothness reward to ensure the smoothness of the action trajectories produced by the policy. The first-order difference term penalizes sudden changes in actions, the second-order difference term penalizes abrupt changes in the rate of action variation, and the L_1-norm term penalizes excessively large actions. At the physical manifestation level, we design a set of rewards, including joint torque reward, joint torque rate reward, joint acceleration reward, foot contact force reward, joint angle limit reward, and collision reward. Together, these rewards promote compliant and smooth motions in the actual robot execution. The joint torque reward penalizes the use of large torques to reduce energy consumption. The joint torque rate reward penalizes rapid changes in torque to prevent instability or jitter. The joint acceleration reward limits the magnitude of joint accelerations to avoid sudden jolts. The contact force reward encourages compliant foot-ground contact. The joint angle limit reward penalizes movements that approach or exceed the mechanical limits of the joints. The collision reward penalizes both self-collisions among robot parts and collisions with the ground.

To improve the efficiency of policy learning, different reward mechanisms are designed for each skill based on its characteristics. During rhythmic skill learning, the imitation reward only tracks joint angles and torso pose, ensuring that the generated gait achieves the desired posture control while producing rhythmic leg movements. During expressive skill learning, the imitation rewards of different body parts are coupled to encourage synchronized and coordinated limb movements, making the generated motions more visually appealing. For example, when training the double-leg swinging skill, the imitation reward for the forelegs is coupled with the torso pose, encouraging the robot to reach a stable posture before swinging its forelegs. During high-dynamic skill learning, the imitation reward tracks only the robot's terminal states, without strictly enforcing a specific motion trajectory, which enhances the robustness of the generated motions.

Additionally, to achieve the double-leg standing capability, a specialized reward function is designed:

$$r_{\text{st}} = (1 - C_{\text{front}}) \exp\left(-\left(\sum_{i=1}^{3} |\hat{\theta}_i - \theta_i|\right)\left(\sum_{i=1}^{6} |\hat{q}_i^{\text{front}} - q_i^{\text{front}}|\right)\right) \tag{17}$$

where $C_{\text{front}} \in \{0, 1\}$ indicates whether the forelegs are in contact with the ground, $\hat{\theta}_i$ and θ_i represent the reference and actual torso Euler angles, respectively, and $\hat{q}_i^{\text{front}}$ and q_i^{front} denote the reference and actual foreleg joint angles, respectively.

3.3.4. PPO-based policy training method

All neural network policies are trained using the proximal policy optimization algorithm, with the hyperparameter settings listed in Table IV. The simulation training platform is Isaac Gym, where the policy network outputs actions at 50 Hz, while the dynamics simulation runs at 200 Hz.

Each training episode is terminated at the duration of the reference motion, but will end early if the onboard manipulator or torso comes into contact with the ground. For the four-leg standing task, we expect the robot to learn to recover from arbitrary states to a standing posture; therefore, early termination

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Table V. *Domain randomization parameters and their uniform sampling ranges.*

Parameter	Randomization range
Friction	[0.4, 2.0]
Restitution	[0.0, 0.4]
Add base mass	[−1.0, 5.0] (kg)
Add link mass	[0.0, 0.2] (kg)
CoM position	[−0.2, 0.2] (m)
Motor gain multiplier	[0.8, 1.2]
Push robot	every 5 s at 1 m/s or 0.6 rad/s

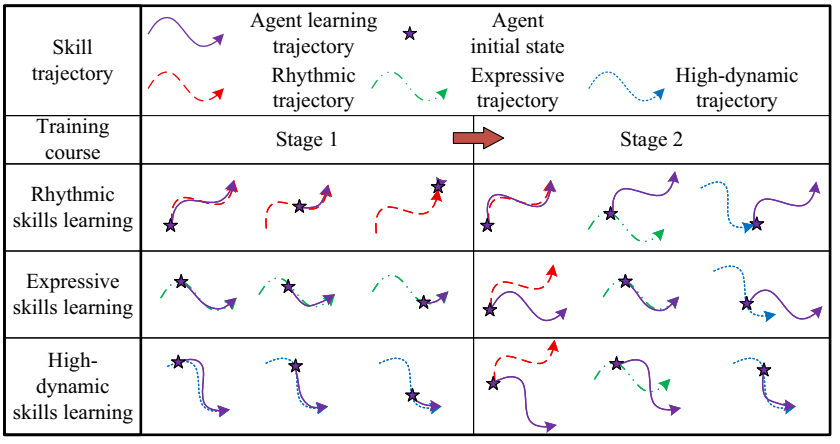


Figure 6. *Two-stage RSI training curriculum.*

due to collisions is disabled during its training. In addition, to facilitate Sim2Real transfer, domain randomization is employed during training to improve the generalization capability of the policy. By introducing random perturbations and parameter variations to both the robot and the environment, the agent learns to perform target motions robustly under real-world physical conditions. The randomized variables and their uniform sampling ranges are listed in Table V.

3.3.5. Multi-skill transition method

The initial state distribution determines the state from which the agent starts at the beginning of each training episode. If the agent always starts from a fixed initial state during training (e.g., always from the beginning of the reference motion), the learned policy is typically limited to initiating motion only from that specific state. Moreover, since the multi-skill trajectory library contains trajectories for individual skills but lacks transition data between them, simple IL methods often fail to achieve continuous motion transitions. This issue is particularly pronounced for heterogeneous skills with significantly different state spaces, where directly switching between policies can result in discontinuous actions.

To enable continuous transitions between different motion skills and allow the robot to start from arbitrary states, we adopt an RSI method and design a two-stage training curriculum, as illustrated in Figure 6. In the first stage, at the beginning of each training episode for a given skill, an initial state is randomly sampled from the corresponding reference trajectory of that skill. In the second stage, the initial state is randomly sampled from all reference trajectories. Through this process, the initial state distribution for each skill is gradually expanded to cover the entire state space of all reference motions. This enables the robot to start from arbitrary states during execution and achieve continuous transitions

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press



Figure 7. *The quadruped robot prototype Panda7-Arm equipped with a 6-DoF manipulator.*

across multiple motion skills. In addition, the RSI method allows the agent to learn different phases of the reference motions in parallel, thereby improving the efficiency of policy learning [42].

3.4. Policy transfer and real-world deployment

The target platform is the quadruped robot Panda7-Arm equipped with a 6-DoF manipulator, as shown in Figure 7. The manipulator is mounted at the center of the robot's back, resulting in a total of 18 DoFs (12 for the legs and 6 for the manipulator). The prototype parameters are listed in Table VI. The actions output by the policy network represents offsets relative to the joint's equilibrium angles, which are then passed through a PD controller to generate torque commands for each joint:

$$\boldsymbol{\tau} = \boldsymbol{k}_{\mathrm{p}}(\boldsymbol{a} + \boldsymbol{q}_0 - \boldsymbol{q}) - \boldsymbol{k}_{\mathrm{d}}\dot{\boldsymbol{q}} \tag{18}$$

where $\boldsymbol{q}$ and $\boldsymbol{q}_0$ represent the current and equilibrium joint angles, respectively, and $\boldsymbol{k}_{\mathrm{p}}$ and $\boldsymbol{k}_{\mathrm{d}}$ denote the proportional and derivative gains.

Finally, the trained policy networks are exported in TorchScript format and evaluated using both Simulation-to-Simulation (Sim2Sim) and Sim2Real transfer methods. The policies are first deployed in a higher-fidelity RaiSim [63] physics simulation environment for validation and parameter tuning and then transferred to the real robot prototype. The policies run at 50 Hz on the robot's onboard AI supercomputer (NVIDIA Jetson AGX Orin), while the low-level joint torque control loop runs at 1,000 Hz. A remote controller sends control commands to the robot via UDP at 200 Hz to enable switching between different motion skills.

4. Results

To evaluate the proposed multi-skill learning method, we first verify its effectiveness in generating diverse motion skills, then demonstrate its capability for seamless transitions among multiple skills, and finally assess its performance under external disturbances. The experimental results show that the policies trained in simulation can be successfully transferred to real hardware. The quadruped robot achieves a variety of motion skills, including rhythmic trotting, dynamic jumping, double-leg standing, torso swaying, single-leg swinging, double-leg swinging, and leg–arm dancing. Furthermore, continuous

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Table VI. *Structural parameters of the Panda7-Arm prototype.*

Entry	Link	Mass (kg)	Dimensions (m)
Quadruped robot	Total	65.9	$1.06 \times 0.5 \times 0.67$
	Torso link	37.9	$0.98 \times 0.35 \times 0.21$
	Hip link	2.4	0.11
	Thigh link	4.1	0.34
	Calf link	0.5	0.38
6-DoF robotic arm	Total	3.4	0.77
	Arm link 1	0.2	0.06
	Arm link 2	0.3	0.10
	Arm link 3	1.1	0.26
	Arm link 4	1.1	0.25
	Arm link 5	0.2	0.07
	Arm link 6	0.3	0.08
	Arm link 7	0.2	0.07

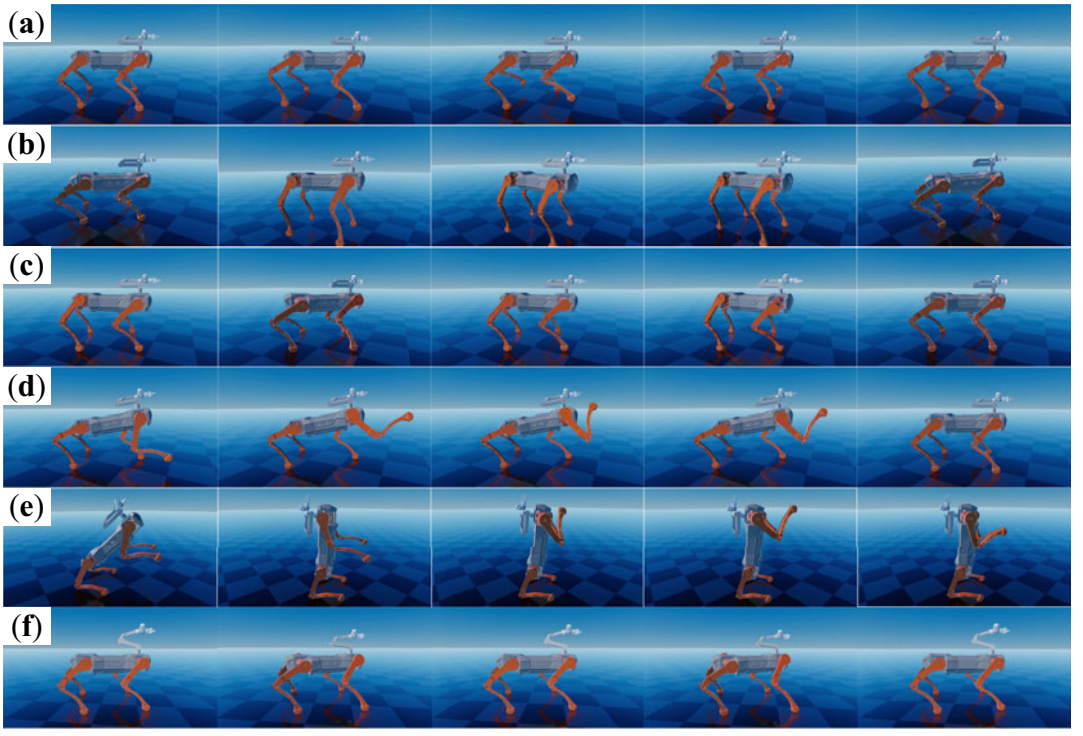


Figure 8. *The policy generates multiple motion skills in the Sim2Sim experiments. Subfigures (**a–f**) correspond to rhythmic trotting, dynamic jumping, torso swaying, single-leg swinging, double-leg swinging, and leg-arm dancing, respectively.*

multi-skill behaviors can be realized through seamless skill transitions, and the system exhibits strong robustness against unknown disturbances.

4.1. Single-skill motion generation

Figures 8 and 9 present snapshots of multiple motion skills achieved by the policy in both Sim2Sim and Sim2Real scenarios. The corresponding motion cycle durations are summarized in Table VII. Our method not only enables versatile legged locomotion for quadruped robots but also facilitates more

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

Table VII. *Periods of the generated motion skill in Sim2Sim and Sim2Real experiments.*

Motion name	Reference (s)	Sim2Sim (s)	Sim2Real (s)	Error (%)
Rhythmic trotting	0.50	0.51	0.49	2.0
Dynamic jumping	0.80	0.87	0.80	11.3
Torso swaying	1.20	1.22	1.29	7.5
Single-leg swinging	0.80	0.70	0.77	12.5
Double-leg swinging	$2\pi/3$	2.06	2.22	6.2
Leg-arm dancing	1.00	1.06	–	6.0

Note: The error is the maximum error relative to the reference motion.

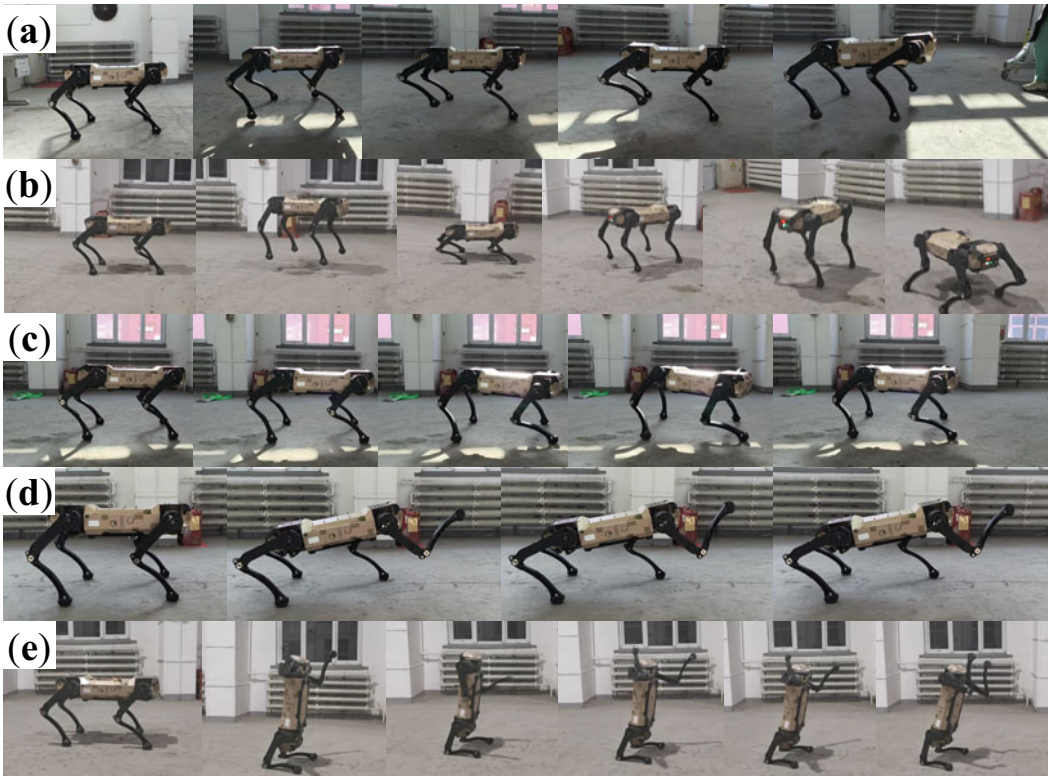


Figure 9. *The policy generates multiple motion skills in the Sim2Real experiments. Subfigures (**a**–**e**) correspond to rhythmic trotting, dynamic jumping, torso swaying, single-leg swinging, and double-leg swinging, respectively.*

complex coordinated leg-arm motions when equipped with a manipulator. The maximum frequency error between the generated motions and the reference motions is 12.5%, demonstrating good imitation performance.

Figure 10 illustrates the variation ranges of the torso Euler angles during motion. Except for the double-leg swinging motion, the pitch angle remains within the range of $[-0.06, 0.08]$ rad and the roll angle within $[-0.1, 0.04]$ rad, indicating stable torso posture control. In the double-leg swinging mode, an upright torso is required, leading to larger motion amplitudes: the pitch angle ranges from $[-0.67, 0.01]$ rad and the roll angle from $[-1.55, -1.46]$ rad. After transitioning from the initial state to steady motion, each skill forms a stable periodic loop in the phase plot, indicating that the Euler angles evolve cyclically within a stable range.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

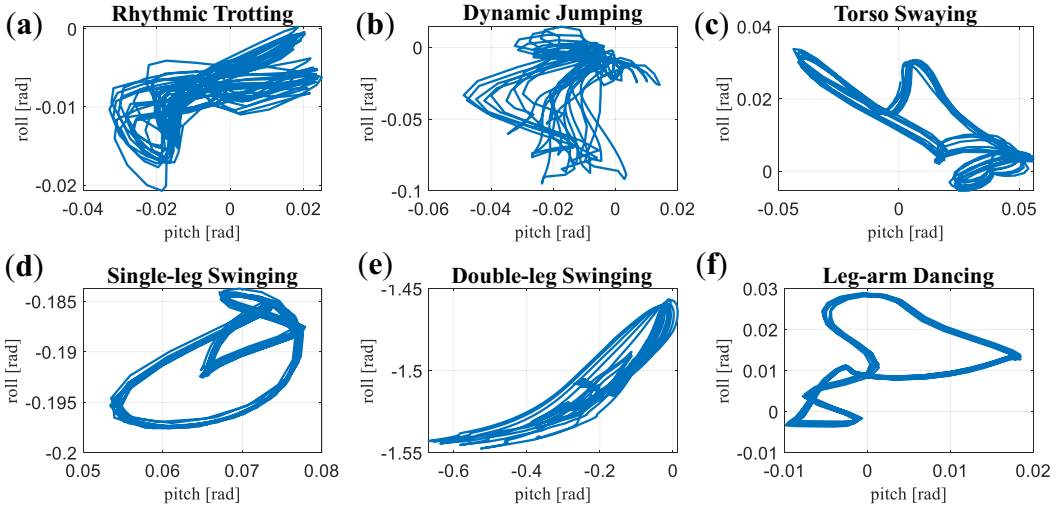


Figure 10. *Phase plots of the torso's pitch and roll angles. Subfigures (**a–f**) correspond to rhythmic trotting, dynamic jumping, torso swaying, single-leg swinging, double-leg swinging, and leg-arm dancing, respectively.*

4.2. Multi-skill motion transitions

To demonstrate the policy's capability for skill transitions, a long-horizon multi-skill experiment was designed by composing multiple motion skills. The experimental results show that, upon receiving a skill-switching command, the policy can respond rapidly, transitioning to the desired skill within half a motion cycle and maintaining stable execution. During the multi-skill experiments in both Sim2Sim and Sim2Real settings, the trunk Euler angles and the joint angles of the front-right leg are illustrated in Figure 11. Figure 11(a) and (b) presents the Sim2Sim results. The transition sequence is four-leg standing, leg-arm dancing, single-leg swinging, rhythmic trotting, torso swaying, dynamic jumping, double-leg standing, double-leg swinging 1, double-leg swinging 2, and four-leg standing. The robot begins the transition from leg–arm dancing to single-leg swinging at 8 s. The front leg begins to swing rhythmically at 9.5 s. This delay is not due to a slow policy response, but rather because switching to single-leg swinging requires completing a leg-lifting motion beforehand. Similar skill transitions can also be achieved on the real robot, as shown in Figure 11(c) and (d). The transition sequence is four-leg standing, rhythmic trotting, torso swaying, single-leg swinging, double-leg standing, double-leg swinging, four-leg standing, dynamic jumping, four-leg standing, torso swaying, rhythmic trotting, single-leg swinging, torso swaying, and four-leg standing.

These results demonstrate that the policy trained with the reference-state initialization curriculum can initiate motion from arbitrary time steps and achieve seamless transitions between different skills. The policy is able to execute predefined motions and compose multiple skills into continuous behaviors without any reference motion input. In contrast, the policy trained without the reference-state initialization curriculum fails to perform effective skill transitions.

4.3. Skill robustness experiment

To evaluate the policy's ability to resist disturbances, the most balance-sensitive skill – double-leg standing – was selected for testing. A sufficiently large push was applied from behind the robot three times consecutively to cause it to fall. Figure 12 shows snapshots of the robot during the test and the variation in trunk Euler angles. After being pushed, the robot pitches forward; the forelegs actively extend forward and push against the ground, eventually restoring the initial double-leg standing state. When subjected to the second disturbance, the first push-off is insufficient for recovery. The trunk begins to pitch forward again, triggering a second foreleg push-off. After two push-offs, the robot regains a stable double-leg standing posture, which results in two oscillations in the pitch angle.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

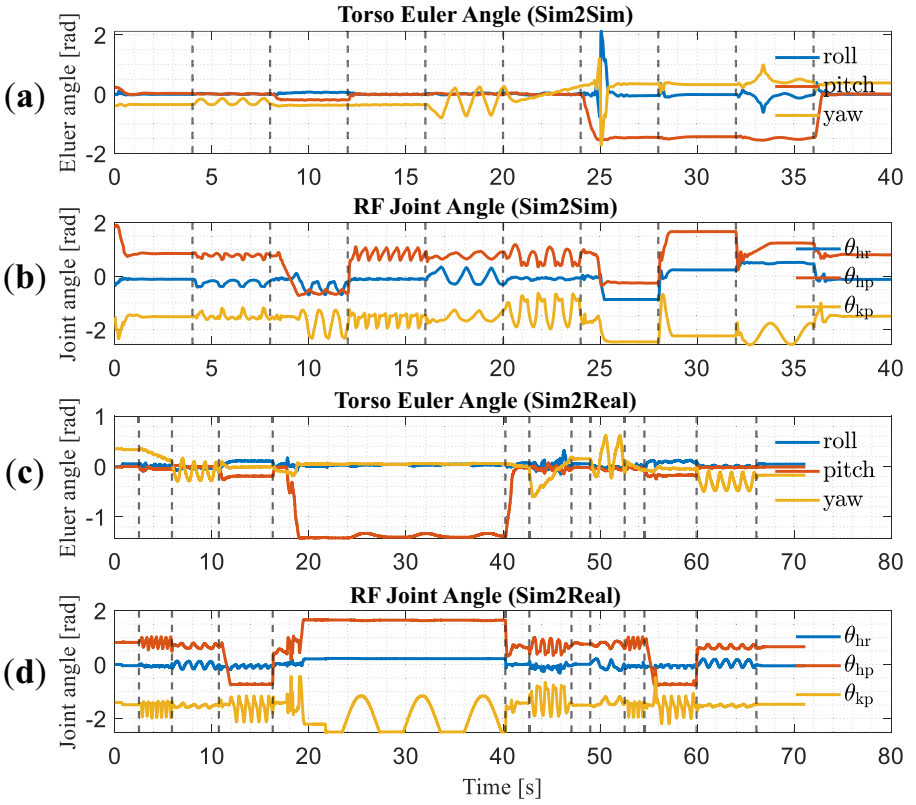


Figure 11. *Torso Euler angles and right-front leg joint angles in the multi-skill motion experiments. In the Sim2Sim experiment, motion skills are switched every 4 s, while in the Sim2Real experiment, skill transitions are triggered by remote control commands. The vertical gray dashed lines indicate the start of each transition. hr, hp, and kp denote the hip roll, hip pitch, and knee pitch DoFs, respectively. (a) Torso Euler angles in the Sim2Sim experiment. (b) Right-front leg joint angles in Sim2Sim. (c) Torso Euler angles in the Sim2Real experiment. (d) Right-front leg joint angles in Sim2Real.*

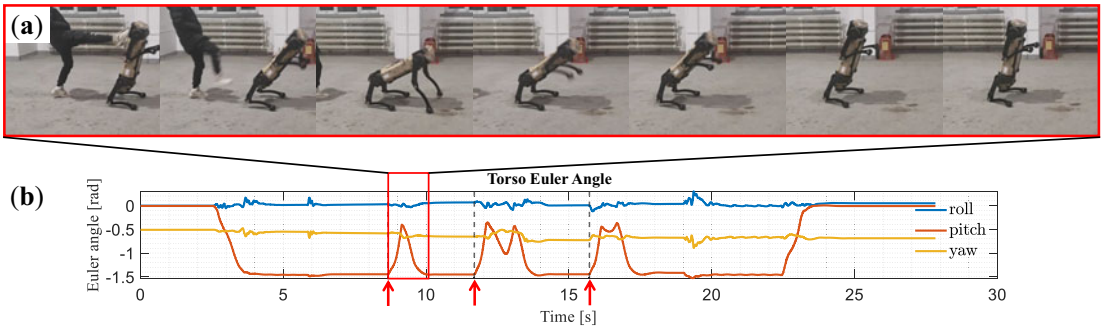


Figure 12. *Recovery capability test of the robot under push disturbances during double-leg standing. (a) Snapshots of the robot recovering to a double-leg standing posture after being pushed and falling at 8.6 s. (b) Torso Euler angle trajectories under three consecutive push disturbances; red arrows indicate the instants when external forces are applied. The robot transitions to double-leg standing at 2.5 s, and it transitions to four-leg standing at 22.5 s.*

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

During the recovery process, a stable standing posture may not be achieved in a single attempt. However, the policy can autonomously adjust its outputs, perform multiple corrections, and ultimately complete the task without human intervention. The results demonstrate that the proposed policy can react to unexpected failures at any time, achieve rapid recovery, and re-enter stable motion, exhibiting robust full-body control capability.

5. Conclusion and discussion

To enable continuous multi-skill motion for quadruped robots, we propose an IRL framework that generates and transitions motion skills by imitating reference trajectories. The policy is trained in simulation and successfully transferred to real hardware. The quadruped robot is able to robustly reproduce all reference motion skills and acquire the ability to seamlessly transition among multiple skills. It can not only achieve long-horizon multi-skill behaviors through policy switching but also autonomously recover from external disturbances. These results validate the feasibility of our proposed three types of motion skill trajectory generation methods and demonstrate the effectiveness of the IRL framework. By formulating the multi-skill motion problem as motion imitation and continuous transition, we avoid the challenges associated with long-horizon motion planning and sparse reward design. The proposed reference-state initialization curriculum enables effective learning of continuous skill transitions, allowing the robot to switch from arbitrary states to target motion skills through policy switching. In addition, the designed asymmetric Actor-Critic network architecture enables the policy to generate consistent motion skills without requiring any reference motion information during deployment.

This work provides a practical solution for reference trajectory acquisition, motion skill generation, and skill transition in quadruped robots. It enriches the motion capabilities of quadruped systems and promotes their applications in entertainment and performance industries. In future work, we plan to incorporate music perception into the multi-skill learning framework, where motion generation and transitions are driven by musical rhythm, enabling continuous multi-skill quadruped locomotion synchronized with music.

Author contributions. All first authors contributed equally to this work. C.P. and S.H. led the research, proposed the core concepts, implemented the framework, and wrote the manuscript. X.Z. supervised the research process and provided critical feedback at each stage. W.L. helped with the research, engaged actively in technical discussions, and helped write the manuscript. H.Z., J.W,, and T.Q. undertook the experimental work and data processing. P.X. and L.J. contributed to the study conception and design and provided critical guidance. All authors commented on previous versions and approved the final manuscript.

Data availability. Data available from the corresponding author upon reasonable request.

Financial support. This work was supported by the STI 2030—Major Projects (grant numbers 2021ZD0201403) and the Beijing Natural Science Foundation (grant numbers L243004).

Competing interests. The authors declare no conflicts of interest exist.

Ethical standards. Not applicable.

Use of AI. ChatGPT (version 5.4, OpenAI) was used solely for minor language editing and polishing to improve the readability of the manuscript. All scientific content, analysis, and conclusions were developed by the authors.

References

[1] G. B. Margolis and P. Agrawal, "Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior," **In:** *Conference on Robot Learning* (2023) pp. 22–31.

[2] Y. Zhang, W. Liu and N. Tan, "Motion control of legged robots based on gradient central pattern generators," *Robotica* **42**(9), 3102–3131 (2024).

[3] H. Kim, H. Oh, J. Park, Y. Kim, D. Youm, M. Jung, M. Lee and J. Hwangbo, "High-speed control and navigation for quadrupedal robots on complex and discrete terrain," *Sci. Robot.* **10**(102), eads6192 (2025).

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

[4] J. Liu, X. Xu, W. Wang, Y. Tang and S. Lu, "Adaptive optimization for virtual model control of quadruped robots based on BP neural network," *Robotica* **43**(4), 1258–1290 (2025).

[5] W. Li, Z. Zhou and H. Cheng, "Dynamic Locomotion of a Quadruped Robot with Active Spine via Model Predictive Control," **In:** *IEEE International Conference on Robotics and Automation* (2023) pp. 1185–1191.

[6] R. Grandia, F. Farshidian, E. Knoop, C. Schumacher, M. Hutter and M. Bacher, "DOC: Differentiable optimal control for retargeting motions onto legged robots," *ACM Trans. Graph.* **42**(4), 1–14 (2023).

[7] Z. Li, C. Cummings and K. Sreenath, "Animated Cassie: A Dynamic Relatable Robotic Character," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2020) pp. 3739–3746.

[8] Z. Cai, R. Cai, Z. Xie, N. Huang and Q. Li, "Load-adaptive control for quadruped mobile manipulation platforms," *Robotica* **44**(1), 195–209 (2026).

[9] T. Chen, X. Rong and Y. Li, "Optimized Method for Planning and Controlling the Somersault Motion of Quadruped Robot," **In:** *IEEE International Conference on Robotics and Automation* (2021) pp. 7658–7664.

[10] J. Ding, M. A. van Löben Sels, F. Angelini, J. Kober and C. D. Santina, "Robust jumping with an articulated soft quadruped via trajectory optimization and iterative learning," *IEEE Robot. Autom. Lett.* **9**(1), 255–262 (2024).

[11] C. Nguyen and Q. Nguyen, "Contact-Timing and Trajectory Optimization for 3D Jumping on Quadruped Robots," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2022) pp. 11994–11999.

[12] Z. Song, L. Yue, G. Sun, Y. Ling, H. Wei, L. Gui and Y. H. Liu, "An Optimal Motion Planning Framework for Quadruped Jumping," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2022) pp. 11366–11373.

[13] C. Nguyen, L. Bao and Q. Nguyen, "Continuous Jumping for Legged Robots on Stepping Stones via Trajectory Optimization and Model Predictive Control," **In:** *IEEE 61st Conference on Decision and Control* (2022) pp. 93–99.

[14] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo and S. Kim , "Optimized Jumping on the MIT Cheetah 3 Robot," **In:** *IEEE International Conference on Robotics and Automation* (2019) pp. 7448–7454.

[15] J. Humphreys, J. Li, Y. Wan, H. Gao and C. Zhou, "Bio-inspired gait transitions for quadruped locomotion," *IEEE Robot. Autom. Lett.* **8**(10), 6131–6138 (2023).

[16] Y. Ding, A. Pandala, C. Li, Y. H. Shin and H. W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Trans. Robot.* **37**(4), 1154–1171 (2021).

[17] M. Bjelonic, R. Grandia, M. Geilinger, O. Harley, V. S. Medeiros, V. Pajovic, E. Jelavic, S. Coros and M. Hutter, "Offline motion libraries and online MPC for advanced mobility skills," *Int. J. Rob. Res.* **41**(9-10), 903–924 (2022).

[18] M. Chignoli and S. Kim, "Online Trajectory Optimization for Dynamic Aerial Motions of a Quadruped Robot," **In:** *IEEE International Conference on Robotics and Automation* (2021) pp. 7693–7699.

[19] M. Chignoli, S. Morozov and S. Kim, "Rapid and Reliable Quadruped Motion Planning with Omnidirectional Jumping," **In:** *IEEE International Conference on Robotics and Automation* (2022) pp. 6621–6627.

[20] S. Gu, F. Meng, B. Liu, J. Gao and Q. Huang, "High dynamic bounding and jumping motion of quadruped robot based on stable optimization control," *J. Bionic Eng.* **21**(1), 101–111 (2024).

[21] H. W. Park, P. M. Wensing and S. Kim, "Jumping over obstacles with MIT Cheetah 2," *Robot. Auton. Syst.* **136**, 103703 (2021).

[22] M. Boukheddimi, D. Harnack, S. Kumar, R. Kumar, S. Vyas, O. Arriaga and F. Kirchner, "Robot Dance Generation with Music Based Trajectory Optimization," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2022) pp. 3069–3076.

[23] G. Xia, J. Tay, R. Dannenberg and M. Veloso, "Autonomous Robot Dancing Driven by Beats and Emotions of Music," **In:** *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems* (2012) pp. 205–212.

[24] Y. Liang, Z. Peng, Y. Zhao and W. Yan, "Learning robust bipedal running via structured gait and trajectory guidance," *Robotica* **44**(1), 150–168 (2026).

[25] D. Youm, H. Jung, H. Kim, J. Hwangbo, H. W. Park and S. Ha, "Imitating and finetuning model predictive control for robust and symmetric quadrupedal locomotion," *IEEE Robot. Autom. Lett.* **8**(11), 7799–7806 (2023).

[26] J. Norby, Y. Yang, A. Tajbakhsh, J. Ren, J. K. Yim, A. Stutt, Q. Yu, N. Flowers and A. M. Johnson, "Quad-SDK: Full Stack Software Framework for Agile Quadrupedal Locomotion," **In:** *ICRA Workshop on Legged Robots* (2022) pp. 1–5.

[27] S. Xu, L. Zhu, H. T. Zhang and C. P. Ho, "Robust convex model predictive control for quadruped locomotion under uncertainties," *IEEE Trans. Robot.* **39**(6), 4837–4854 (2023).

[28] Q. Zhou, S. Yang, X. Jiang, D. Zhang, W. Chi, K. Chen, S. Zhang, J. Li, J. Zhang and R. Wang, "Max: A wheeled-legged quadruped robot for multimodal agile locomotion," *IEEE Trans. Autom. Sci. Eng.* **21**(4), 7562–7582 (2023).

[29] D. Vogel, R. Baines, J. Church, J. Lotzer, K. Werner and M. Hutter, "Robust ladder climbing with a quadrupedal robot," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2025) pp. 7239–7244.

[30] T. Miki, J. Lee, L. Wellhausen and M. Hutter, "Learning to Walk in Confined Spaces Using 3D Representation," **In:** *IEEE International Conference on Robotics and Automation* (2024) pp. 8649–8656.

[31] N. Rudin, D. Hoeller, P. Reist and M. Hutter, "Learning to Walk in Minutes using Massively Parallel Deep Reinforcement learning," **In:** *Conference on Robot Learning* (2022) pp. 91–100.

[32] Y. Ma, A. Cramariuc, F. Farshidian and M. Hutter, "Learning coordinated badminton skills for legged manipulators," *Sci. Robot.* **10**(102), eadu3922 (2025).

[33] N. Rudin, J. He, J. Aurand and M. Hutter, "Parkour in the wild: Learning a general and extensible agile locomotion policy using multi-expert distillation and RL fine-tuning," *Int. J. Robot. Res.* **0**(0), 02783649261455067 (2026).

[34] N. Rudin, D. Hoeller, M. Bjelonic and M. Hutter, "Advanced Skills by Learning Locomotion and Local Navigation End-to-End," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2022) pp. 2497–2503.

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press

[35] L. Schneider, J. Frey, T. Miki and M. Hutter, "Learning Risk-Aware Quadrupedal Locomotion using Distributional Reinforcement Learning," **In:** *IEEE International Conference on Robotics and Automation* (2024) pp. 11451–11458.
[36] D. Hoeller, N. Rudin, D. Sako and M. Hutter, "ANYmal parkour: Learning agile navigation for quadrupedal robots," *Sci. Robot.* **9**(88), eadi7566 (2024).
[37] X. Cheng, K. Shi, A. Agarwal and D. Pathak, "Extreme Parkour with Legged Robots," **In:** *IEEE International Conference on Robotics and Automation* (2024) pp. 11443–11450.
[38] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn and H. Zhao, "Robot parkour learning," **In:** *Conference on Robot Learning* (2023) pp. 73–92.
[39] C. Li, S. Blaes, P. Kolev, M. Vlastelica, J. Frey and G. Martius, "Versatile skill control via self-supervised adversarial imitation of unlabeled mixed motions," **In:** *IEEE International Conference on Robotics and Automation* (2023) pp. 2944–2950.
[40] C. Ran and J. Su, "Task-oriented self-imitation learning for robotic autonomous skill acquisition," *Int. J. Humanoid Robot.* **21**(05), 2450001 (2024).
[41] E. Chane-Sane, C. Roux, O. Stasse and N. Mansard, "Reinforcement learning from wild animal videos," **In:** *Conference on Robot Learning* (2024).
[42] X. B. Peng, P. Abbeel, S. Levine and M. Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.* **37**(4), 1–14 (2018).
[43] X. B. Peng, E. Coumans, T. Zhang, T. W. Lee, J. Tan and S. Levine, "Learning agile robotic locomotion skills by imitating animals," **In:** *Robotics: Science and Systems* (2020).
[44] R. Grandia, E. Knoop, M. A. Hopkins, G. Wiedebach, J. Bishop, S. Pickles, D. Müller and M. Bächer, "Design and control of a bipedal robotic character," **In:** *Robotics: Science and Systems* (2024).
[45] X. B. Peng, Z. Ma, P. Abbeel, S. Levine and A. Kanazawa, "AMP: Adversarial motion priors for stylized physics-based character control," *ACM Trans. Graph.* **40**(4), 1–20 (2021).
[46] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg and P. Abbeel, "Adversarial Motion Priors Make Good Substitutes for Complex Reward Functions," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2022) pp. 25–32.
[47] C. Li, M. Vlastelica, S. Blaes, J. Frey, F. Grimminger and G. Martius, "Learning Agile Skills via Adversarial Imitation of Rough Partial Demonstrations," **In:** *Conference on Robot Learning* (2023) pp. 342–352.
[48] F. Jenelten, J. He, F. Farshidian and M. Hutter, "DTC: Deep tracking control," *Sci. Robot.* **9**(86), eadh5401 (2024).
[49] Y. Yang, T. Zhang, E. Coumans, J. Tan and B. Boots, "Fast and Efficient Locomotion via Learned Gait Transitions," **In:** *Conference on Robot Learning* (2022) pp. 773–783.
[50] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich and A. Garg, "Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion," **In:** *Conference on Robot Learning* (2021) pp. 883–894.
[51] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. Kim and P. Agrawal, "Learning to jump from pixels," **In:** *Proceedings of the 5th Conference on Robot Learning* (2022) pp. 1025–1034.
[52] Y. Shao, Y. Jin, X. Liu, W. He, H. Wang and W. Yang, "Learning free gait transition for quadruped robots via phase-guided controller," *IEEE Robot. Autom. Lett.* **7**(2), 1230–1237 (2021).
[53] J. Tu, X. Wei, Y. Zhang, T. Hou, X. Gao, Z. Dong, P. Zhai and L. Zhang, "Continuous Control of Diverse Skills in Quadruped Robots Without Complete Expert Datasets," arXiv:2503.03476 (2025).
[54] J. Tu, P. Zhai, Y. Zhang, X. Wei, Z. Dong and L. Zhang, "Seamless multi-skill learning: Learning and transitioning non-similar skills in quadruped robots with limited data," *Front. Robot. AI* **12**, 1542692 (2025).
[55] W. Yu, F. Acero, V. Atanassov, C. Yang, I. Havoutis, D. Kanoulas and Z. Li, "Discovery of skill-switching criteria for learning agile quadruped locomotion," *Front. Robot. AI* **13**, 1697159 (2026).
[56] C. Yang, C. Pu, Y. Zou, T. Wei, C. Wang and Z. Li, "Bio-inspired neural networks with central pattern generators for learning multi-skill locomotion," *Sci. Rep.* **15**(1), 10165 (2025).
[57] R. Watanabe, C. Li and M. Hutter, "DFM: Deep Fourier Mimic for Expressive Dance Motion Learning," arXiv:2502.10980 (2025).
[58] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Netw.* **21**(4), 642–653 (2008).
[59] F. Dzeladini, N. Ait-Bouziad and A. Ijspeert, "CPG-Based Control of Humanoid Robot Locomotion," **In:** *Humanoid Robotics: A Reference* (2017) pp. 1–35.
[60] V. Matos, C. Santos and C. Pinto, "A Brainstem-Like Modulation Approach for Gait Transition in a Quadruped Robot," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2009) pp. 2665–2670.
[61] L. Righetti and A. J. Ijspeert, " Pattern Generators with Sensory Feedback for the Control of Quadruped Locomotion," **In:** *IEEE International Conference on Robotics and Automation* (2008) pp. 819–824.
[62] W. Xiao and W. Wang, "Hopf Oscillator-Based Gait Transition for a Quadruped Robot," **In:** *IEEE International Conference on Robotics and Biomimetics* (2014) pp. 2074–2079.
[63] J. Hwangbo, J. Lee and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robot. Autom. Lett.* **3**(2), 895–902 (2018).

Cite this article: C. Pi, S. Huang, W. Li, H. Zhao, J. Wu, T. Qiu, P. Xu, L. Jiang and X. Zhang (2026). "Continuous multi-skill motion generation for quadruped robots based on imitation–reinforcement learning", Robotica **44**, 1630–1651. https://doi.org/10.1017/S0263574726103634

https://doi.org/10.1017/S0263574726103634 Published online by Cambridge University Press