Journal of Zhejiang University-SCIENCE A (Applied Physics & Engineering) 2024 25(6):443-454
www.jzus.zju.edu.cn; www.springer.com/journal/11582
E-mail: jzus_a@zju.edu.cn

Research Article

https://doi.org/10.1631/jzus.A2300128

A learning-based control pipeline for generic motor skills for quadruped robots

Yecheng SHAO[1,2], Yongbin JIN[1,2], Zhilong HUANG[4], Hongtao WANG[1,2,3]✉, Wei YANG[1,2]

[1]*Center for X-Mechanics, Zhejiang University, Hangzhou 310027, China*
[2]*ZJU-Hangzhou Global Scientific and Technological Innovation Center, Hangzhou 311200, China*
[3]*State Key Laboratory of Fluid Power & Mechatronic Systems, Zhejiang University, Hangzhou 310058, China*
[4]*Institute of Applied Mechanics, Zhejiang University, Hangzhou 310027, China*

Abstract: Performing diverse motor skills with a universal controller has been a longstanding challenge for legged robots. While motion imitation-based reinforcement learning (RL) has shown remarkable performance in reproducing designed motor skills, the trained controller is only suitable for one specific type of motion. Motion synthesis has been well developed to generate a variety of different motions for character animation, but those motions only contain kinematic information and cannot be used for control. In this study, we introduce a control pipeline combining motion synthesis and motion imitation-based RL for generic motor skills. We design an animation state machine to synthesize motion from various sources and feed the generated kinematic reference trajectory to the RL controller as part of the input. With the proposed method, we show that a single policy is able to learn various motor skills simultaneously. Further, we notice the ability of the policy to uncover the correlations lurking behind the reference motions to improve control performance. We analyze this ability based on the predictability of the reference trajectory and use the quantified measurements to optimize the design of the controller. To demonstrate the effectiveness of our method, we deploy the trained policy on hardware and, with a single control policy, the quadruped robot can perform various learned skills, including automatic gait transitions, high kick, and forward jump.

Key words: Quadruped robot; Reinforcement learning (RL); Motion synthesis; Control

1 Introduction

In recent years, learning-based control has shown outstanding performance on quadruped robots traversing complex terrains (Lee et al., 2020; Siekmann et al., 2021a; Agarwal et al., 2022; Miki et al., 2022), achieving better performance (Dao et al., 2021; Jin et al., 2022), or performing agile skills (Huang et al., 2022; Ji et al., 2022). Among various reinforcement learning (RL) methods, imitation-based RL is a convenient but powerful method to obtain control policies based on given examples. Recent studies have shown many impressive skills including high-speed running (Jin, et al., 2022), hopping (Siekmann et al., 2021b), and backflip (Fuchioka et al., 2023). However, most of that studies focus on standalone predefined motion clips. Learning and switching among various skills are hard to achieve without further modifications.

In the computer graphics community, motion synthesis has been widely studied to generate controllable and responsive motions for character animation. While some researchers focus on physics-based character animation, most of the methods used in industry are kinematics-based animation. Those well-established methods for kinematics-based animation are a good starting point for motion imitation for achieving controllable motions for quadruped robots.

In this study, we introduce a control pipeline as a combination of motion synthesis and motion imitation. A series of reference trajectories serve as the interface between those two parts. The interface only contains basic kinematics information and thus is compatible with all kinds of data sources, including motion capture, sketch, and optimization. During training, a control policy is trained to imitate a set of given motion clips.

✉ Hongtao WANG, htw@zju.edu.cn
Hongtao WANG, https://orcid.org/0000-0002-8258-4278

Received Mar. 19, 2023; Revision accepted June 12, 2023;
Crosschecked Dec. 27, 2023; Online first Feb. 12, 2024

Based on those given motions, an animation state machine is constructed to generate reference trajectories for online control according to the command from the user. We deploy the proposed control pipeline on Unitree Go1 and demonstrate skills including locomotion with gait transition, high kick, and jump, from motion capture, sketch, and optimization, respectively. We further investigate the effect of the length of the reference trajectories in the interface. While longer reference trajectory can bring better tracking performance in general, we notice the ability of the policy to predict future steps of the reference trajectory based on given steps. Such ability can reduce the necessary length of the reference trajectory, and the performance gain from the reference trajectory is affected by the predictability of the trajectory itself.

2 Related work

2.1 Imitation-based reinforcement learning for legged robots

The design of the reward function in RL is a key part that affects the final behavior of the trained policy. Motion imitation has been demonstrated as a powerful tool to simplify the design of reward, which is expressed as the sum of goal reward and imitation reward. Current methods in this field can be divided into two categories: trajectory-based methods (Peng et al., 2018) and style-based methods (Peng et al., 2021). Trajectory-based methods are similar to vanilla RL. The imitation term is calculated using the error between the current state and the corresponding state on the reference trajectory. In the observation space, the reference trajectory can be either encoded as phases or directly kept in the form of joint and base states. Various skills including locomotion, hopping, and backflip have been presented in previous studies with such methods (Peng et al., 2020; Siekmann et al., 2020, 2021b; Jin et al., 2022; Shao et al., 2022; Fuchioka et al., 2023). Policies with phase encoding are limited to a certain motion or a small number of motions that share the same encoding while, with raw joint and base states, the policy can be applied to more tasks.

To extend to more skills, some of the research (Siekmann et al., 2021b; Shao et al., 2022) uses a shared encoding among various gaits to achieve gait transitions, but the extensibility of those encodings is limited. The multiplicative model (Peng et al., 2019) can learn multiple motion clips and transfer that learning to composite tasks, but it involves a complex model and the structure of the high-level policy is different among various tasks. Peng et al. (2020) used a series of future reference steps in the observation and held the same structure among various tasks, but each policy only worked for the corresponding motion clip.

Style-based methods are more complex than vanilla RL. Instead of frame-to-frame comparison, a discriminator is used to distinguish between the reproduced motions and the reference motions. The imitation reward is obtained based on the discriminator. The training is done in an adversarial way. Since such methods do not involve frame-to-frame matching to the reference motion, no phase or reference trajectory is used in the observation. Even though the style-based reward ensures more flexibility for the policy, seamlessly switching among various tasks requires more effort. Compared to trajectory-based methods, there is less work on legged robots (Escontrela et al., 2022; Li et al., 2022; Vollenweider et al., 2022) with style-based methods.

To extend to more skills, one-hot labels (Vollenweider et al., 2022) can be used to switch among three kinds of skills. Previous research (Escontrela et al., 2022) shows that with adversarial motion priors (AMP), a simple model can learn locomotion in various gaits, but only the velocity is controllable. Hierarchy models can be used to mix the pre-trained skills (Peng et al., 2022), but a task-specific high-level policy is also involved, and currently, to the best of our knowledge, there is no hardware transfer of this method.

2.2 Character animation

In computer graphics, many techniques for character animation have been developed in the fields of research and industry. Character animation can be divided into two classes: kinematics-based animation, which does not consider physics laws, and physics-based animation. Kinematics-based animation has been widely used in production and the branch most relevant to us is motion synthesis, which aims to generate vivid and responsive motions according to user commands, commonly based on motion capture. Animation state machines and motion matching (Clavet, 2016) are two popular methods in the industry, while in the research community, many data-driven approaches have

been proposed to work with a large amount of data in more efficient ways (Holden et al., 2017, 2020; Zhang et al., 2018; Starke et al., 2019, 2022; Ling et al., 2020). On the other hand, physics-based character animation aims to build controllers for the character in a physics simulation to make the animation look real (Peng et al., 2018, 2021, 2022). The idea of using RL-based motion imitation for legged robots is also transferred from this area (Xie et al., 2019; Peng et al., 2020). However, those methods typically focus on dealing with the physics in low-level control, rather than high-level motion synthesis compared to kinematics-based animations.

Our study can be regarded as a combination of the above two kinds of methods and its transfer to the robot. We build a control pipeline using the animation state machine for motion synthesis and reference-based motion imitation for control; thus, controllable motions can be generated by the animation state machine and tracked by the controller on the robot.

3 Control pipeline

3.1 Overview

Fig. 1 shows an overview of the proposed control pipeline. From the most abstract point of view, the control pipeline consists of two major modules: motion generation and motion imitation. The reference trajectory serves as an interface across those two parts. The motion generation module generates the kinematic reference trajectories according to the user, and the motion imitation module tracks them.

More precisely, it starts with a collection of desired motion clips, from either motion capture, optimization, or sketch. The motion clips are retargeted to the desired quadruped robot if needed. Then, a control policy is trained in simulation to imitate the collected motions. When it comes to online control, an animation state machine is designed to generate controllable motions from the dataset. With those motions organized in the same way as offline training, the robot hardware can track those motions under the trained control policy.

3.2 Motion capture data

An open-source motion-capture dataset for quadruped animals (Zhang et al., 2018) is used in this study. Natural locomotion and gait transitions are recorded in the dataset. The dataset is originally collected for animation, and thus the morphology is different from the robot, as shown in Fig. 2. The motion capture data is retargeted to the robot through inverse kinematics (IK). Since the robot has a rigid trunk, the position and orientation of the trunk are fitted from four shoulders in motion capture with the least square method. Once the trunk is fixed, the joint positions are obtained using IK by keeping four feet as the key points, after scaling the size from animal to robot. Due to technical limitations

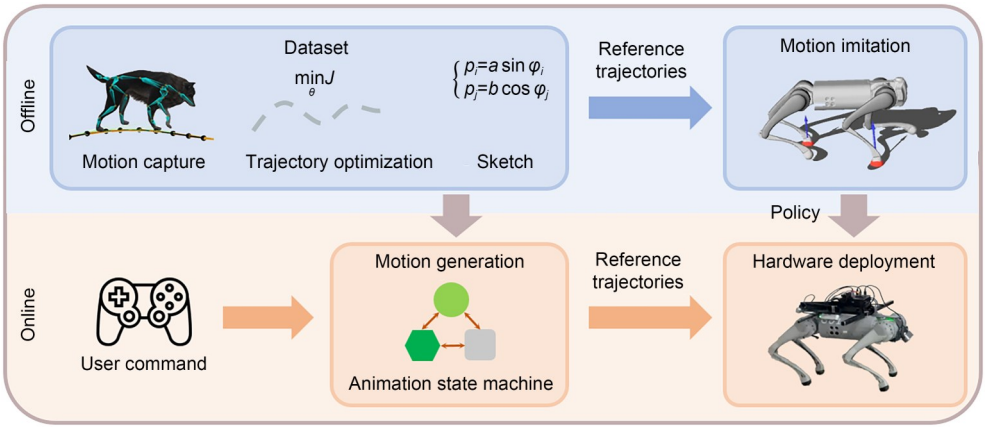

Fig. 1 Illustration of the overall control pipeline proposed in this study. In offline training, the policy is trained to imitate motions from the presented dataset. For online control, a state machine is used to generate motions according to user command based on the dataset, and the trained policy takes the generated reference trajectories as part of the input to track the commanded motion with the hardware. $\min_\theta J$ stands for a typical formulation for trajectory optimization problems, where J is the target function and θ are the optimization variables. $[p_i, p_j]^T=[a\sin\varphi_i, b\cos\varphi_j]^T$ stands for a typical formulation of sketch for motions, where p_i and p_j represent the base and joint positions of the robot, respectively; a and b represent the parameters for the designed motions; φ_i and φ_j represent the time variables

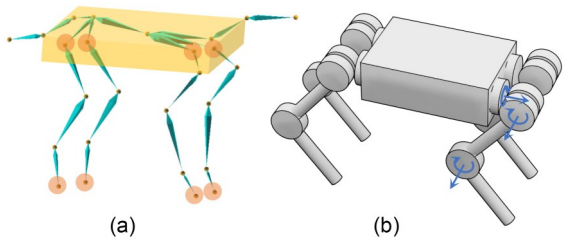

Fig. 2 Skeleton of motion capture (a) and configuration of the robot (b). Circles represent shoulders and feet while the cuboid represents the trunk. There are overall 12 joints on the robot and three for each leg. All 12 joints are actuated by motors. Arrows represent the directions of the joints on one leg

of motion capture, the accuracy of heights of the stand feet is not satisfactory, which may confuse the controller in the following workflow. To mitigate the effect of such noise, a thresholding technique (Kang et al., 2021) for foot height is adopted in pre-processing. Feet heights under the threshold are set to zero and gradually increase to the original values.

3.3 Other motions

Apart from motion capture, sketch and optimization are two other sources of reference motion. A major advantage of motion capture is that it looks natural, but it is hard to record the motions for certain tasks such as the kick and jump of a quadruped animal. Compared to motion capture, sketch and optimization are efficient ways to design motions. Sketch trajectory is suitable for simple tasks. Sketch starts from the design of an approximate trunk trajectory and task-related foot trajectories regardless of dynamics, and then joint states are obtained through IK. For highly dynamic tasks, the sketched trajectories may be too far from the dynamically feasible trajectories and be hard to reproduce even in simulation. For trajectory optimization, the solution satisfies constraints from the approximated robot dynamics determined by the model. Such optimized trajectories are better references for highly dynamic tasks, compared to sketch trajectories without any dynamic constraints.

3.4 Animation state machine

The animation state machine is a popular approach for character animation in the field of computer graphics. The animation is broken down into several states and transitions, and each has a corresponding motion clip. With a set of transition rules, the character may transit from one state to another, or stay in the current state, according to the command from the user. In the meanwhile, the corresponding motion clips are played one by one, resulting in the controllable animation shown on the screen.

To achieve controllable locomotion, nine states are defined by traversing three levels of forward velocities (stand, slow speed, and fast speed) and three moving directions (forward, left, and right). All pairwise transitions among all the locomotion states are allowed and the corresponding motion clips are extracted from the dataset. The rules for gait transitions are automatically included here. For sketched and optimized motions, only transitions between task motion and the standing state are allowed. The robot in locomotion states can first stop to the standing state and then transit into the task motion, since designing direct transitions between locomotion and special tasks is tedious and unnecessary.

3.5 Sim-to-real reinforcement learning

The policy is trained in simulation and transferred to the robot hardware. The policy is trained to imitate the given motion while satisfying the rules of physics and other constraints defined in the simulator.

The overall control architecture is illustrated in Fig. 3 and the control policy in RL is formulated in the same way. The observation of the policy is $\boldsymbol{X}=\left\{\boldsymbol{q}_{\mathrm{f}},\ \dot{\boldsymbol{q}}_{\mathrm{f}},\ \dot{\boldsymbol{q}}_{\mathrm{b_R}},\ \boldsymbol{r}_{\mathrm{g}},\ \boldsymbol{x}_{t:t+N}^{\mathrm{ref}}\right\}\in\mathbb{R}^{30+19N}$ and can be divided into two categories. The notation $\boldsymbol{W}\in\mathbb{R}^{D}$ means that the dimension of vector $\boldsymbol{W}$ is D. The first part is the state of the robot, including the positions and velocities of 12 joints and the orientation and angular velocities of the trunk. The second part is a series of reference trajectories starting from the next step. $\boldsymbol{x}_t^{\mathrm{ref}}=\left\{\hat{\boldsymbol{v}},\ \hat{\boldsymbol{r}}_{\mathrm{g}},\ \hat{h},\ \hat{\boldsymbol{q}}_{\mathrm{j}}\right\}\in\mathbb{R}^{19}$ represents the reference motion at step t. Velocity,

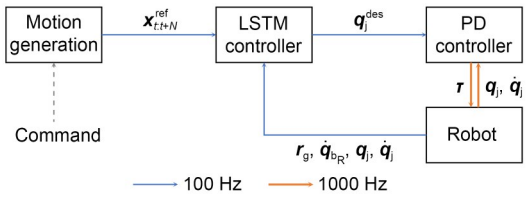

Fig. 3 Control architecture of the proposed method. The colors of the arrows stand for the frequencies, and the command for motion generation is not required to be updated at a specific frequency. LSTM: long short-term memory; PD: proportional-derivative. References to color refer to the online version of this figure

body orientation, height, and joint positions are included in the observation as a description of the imitation task. The effect of these trajectories will be discussed in Section 4. Detailed description of the parameters can be found in Table 1.

Table 1 Parameters for model representation

Parameter	Dimension	Description
$\boldsymbol{q}_{\mathrm{j}}$	12	Joint positions
$\dot{\boldsymbol{q}}_{\mathrm{j}}$	12	Joint velocities
$\dot{\boldsymbol{q}}_{\mathrm{b_R}}$	3	Angular velocities of the trunk
$\boldsymbol{r}_{\mathrm{g}}$	3	Angular positions of the trunk, represented by the components of the gravity direction in the trunk frame
$\boldsymbol{x}_{t:t+N}^{\mathrm{ref}}$	19N	N steps of future reference trajectories
$\hat{\boldsymbol{v}}$	3	Velocity of the reference state in the forward, lateral, and yaw directions
$\hat{\boldsymbol{r}}_{\mathrm{g}}$	3	Angular positions of the trunk of the reference state, in the same representation as $\boldsymbol{r}_{\mathrm{g}}$
$\hat{h}$	1	Trunk height in the reference
$\hat{\boldsymbol{q}}_{\mathrm{j}}$	12	Joint positions of the reference state
$\boldsymbol{q}_{\mathrm{j}}^{\mathrm{des}}$	12	Desired joint positions

The action of the policy is $\boldsymbol{q}_{\mathrm{j}}^{\mathrm{des}}\in\mathbb{R}^{12}$, followed by a PD controller to obtain the final low-level torque command. LSTM neural network is used as the policy due to its ability to capture historical information during dynamic processes (Siekmann et al., 2020).

The reward design is like many previous studies using motion imitation. It is a weighted sum of several terms:

$$r=0.2r_{\mathrm{v}}+0.1r_{\mathrm{h}}+0.1r_{\mathrm{b}}+0.2r_{\tau}+0.4r_{\mathrm{j}}, \tag{1}$$

$$r_{\mathrm{v}}=\exp\left(-8\left\|\dot{\boldsymbol{q}}_{\mathrm{b}}-\hat{\boldsymbol{v}}\right\|^2\right), \tag{2}$$

$$r_{\mathrm{h}}=\exp\left(-80\left\|q_{\mathrm{b}_z}-\hat{h}\right\|^2\right), \tag{3}$$

$$r_{\mathrm{b}}=\exp\left(-80\left\|\boldsymbol{r}_{\mathrm{g}}-\hat{\boldsymbol{r}}_{\mathrm{g}}\right\|^2\right), \tag{4}$$

$$r_{\tau}=0.5\exp\left(-\left\|0.05\boldsymbol{\tau}\right\|^2\right)+0.5\exp\left(-\left\|0.5\dot{\boldsymbol{\tau}}\right\|^2\right), \tag{5}$$

$$r_{\mathrm{j}}=0.25\exp\left(-2\left\|\Delta\boldsymbol{q}_{\mathrm{j}}\right\|^2\right)+0.75\exp\left(-2\left\|\Delta\dot{\boldsymbol{q}}_{\mathrm{j}}\right\|^2\right), \tag{6}$$

where r_{j} encourages the policy to track the reference trajectories at joint level and is the major part during learning. $\Delta\boldsymbol{q}_{\mathrm{j}}$ and $\Delta\dot{\boldsymbol{q}}_{\mathrm{j}}$ represent the differences of joint positions and velocities between the simulated states and the reference trajectories, respectively. r_{v}, r_{h}, and r_{b} reward the policies for the tracking velocities, height, and posture of the trunk, respectively. $\dot{\boldsymbol{q}}_{\mathrm{b}}$ stands for the velocity of the robot base and q_{b_z} for the height. Those are the tracking tasks at high level. r_{τ} penalizes the large or rapid torque command for the joints, which is undesired in the hardware. $\boldsymbol{\tau}$ is the torque command and $\dot{\boldsymbol{\tau}}$ stands for the change rate of the torque command.

For each rollout, the reference state initialization (RSI) technique (Peng et al., 2018) is used to initiate the robot at a random timestep on the reference trajectory. To accelerate training, fixed trajectories sampled from the dataset are used rather than synthesis motion. Early termination (Peng et al., 2018) is also used to stop the rollout when the robot falls, or the state is too far from the reference trajectory. For sim-to-real, domain randomization is adopted to minimize the gap between simulation and real world and prevent overfitting. For the robot, manufactory errors are simulated by randomizing the dynamics and geometric sizes around the designed values. For the ground, a wide range of friction coefficients is used for different rollouts to simulate various ground conditions. Random external forces and torques are applied to the robots, and stochastic errors are added to the observations according to the accuracy of the sensors of the robot. Details on domain randomization can be found in Table 2. $U(x_1, x_2)$ stands for uniform distribution in the interval of $[x_1, x_2]$.

Table 2 Domain randomization

Term	Distribution
Mass for each part (kg)	N(1.00, 0.05)×origin values
Geometric size (m)	N(1.00, 0.05)×origin values
Ground friction	U(0.4, 1.2)
Joint position noise (rad)	N(0.000, 0.002)
Joint velocity noise (rad/s)	N(0.0, 0.3)
Body posture noise (rad)	N(0.0, 0.1)
Angular velocity noise (rad/s)	N(0.0, 0.3)

4 Future reference trajectory

In the problem of optimal control, the target functional consists of costs at the endpoints and along the whole process. The specific design of the target functional varies among different tasks, but in reference tracking tasks, the most common costs consist of the tracking errors against the reference and actuation costs. Model predictive control (MPC) solves the optimal control problem repeatedly using the latest state and

performs the latest output of the optimal control problem. In the tracking problem, each time MPC updates, a slide window of future reference trajectory is fetched, and new results are obtained with regard to the new measurements and the reference. To this end, the problems of MPC and RL are in the same formulation, both obtaining commands based on the current state and a slice of future reference trajectory, and both optimizing tracking errors and actuation costs.

The major difference is that MPC is based on online optimization while RL is based on offline optimization. As an online optimization approach, MPC itself does not contain any prior knowledge of the reference motion. The length of the reference trajectory and the prediction horizon must be the same to formulate the target functional. Usually, the prediction horizon and the updating frequency are a trade-off limited by the onboard computational power. A longer prediction horizon will bring a larger calculation burden, while a shorter one will harm the quality of the results. As an offline optimization approach, the RL policy can gain knowledge about the trained reference dataset. With this knowledge, the policy can predict later parts of the trajectories based on the given ones and is able to work with very few steps of reference trajectories without worrying about performance.

The task of prediction is fitting a dataset like $\left\{(\boldsymbol{x}^{\mathrm{ref}}_{1-N:1},\ \boldsymbol{x}^{\mathrm{ref}}_{1:1+M}),\ (\boldsymbol{x}^{\mathrm{ref}}_{2-N:2},\ \boldsymbol{x}^{\mathrm{ref}}_{2:2+M}),\ \cdots,(\boldsymbol{x}^{\mathrm{ref}}_{t-N:t},\ \boldsymbol{x}^{\mathrm{ref}}_{t:t+M})\right\}$ (t= 1, 2, ⋯, n), and thus the capability of prediction is determined by the correlation inside the dataset, or the mutual information between previous steps and later steps, instead of the policy. For trajectories with a strong correlation between the adjacent data before and after, the policy can predict longer later trajectory based on a few steps of the given trajectory. For example, for periodic motions, the policy can memorize the whole trajectory and identify the current position on it to achieve a completely accurate prediction. For trajectories with weak correlation, such as unexpected transitions, the policy is unable to predict future changes based on given steps, and thus more steps are required to complete the necessary information. Otherwise, a degradation in performance will appear. For a dataset with many trajectories, the overall correlation, or the predictability, also depend on those trajectories within this dataset. The correlation tends to be strong when a set of simple and dissimilar trajectories is included in the dataset and tends to be weak in the case of complex and similar trajectories. The prediction ability only applies to the learned dataset and has no contribution to trajectories completely out of the distribution, like many other machine learning problems. However, since the proposed method only uses learned motions in the control pipeline, this limitation can be ignored in our study, along with the issue of overfitting.

5 Results

To demonstrate the efficiency of our control pipeline, the control policy is trained in simulation and then deployed on the hardware of Unitree Go1. The agent is trained to imitate motions from motion capture, sketch, and optimization simultaneously. Similar performances are observed on both simulation and hardware. This section first demonstrates three sorts of learned motions on hardware, as shown in Videos S1–S4 of the electronic supplementary materials (ESM). All the skills are controlled by the same policy, as shown in Video S5 of the ESM. The policy can achieve similar performance when learning multiple skills and single skills. Besides, we evaluate the role of the future reference trajectories by detailed analysis on data from simulation.

5.1 Experimental setup

The control policy used in this section is trained in simulation. RaiSim (Hwangbo et al., 2018) is used as the physics engine, with an LSTM neural network composed of two hidden layers of 64 units in each. The proximal policy optimization (PPO) algorithm (Schulman et al., 2017) implemented by the stable-baseline (Hill et al., 2018) package is used for training. Similar to previous study (Shao et al., 2022), the standard deviation of the action distribution is trainable and initialized at 1.0 but ended at 0.1 to keep the exploration field at a reasonable scale in the later stage of the training process. The policy is designed to work at 100 Hz while the physics engine works at a higher frequency of 400 Hz. On a workstation with Intel Xeon E5-2296v4 computer processing unit (CPU) and GTX 1080Ti GPU, the policy converges in about 3 h. For deployment, the LSTM policy is re-implemented with C++ and Eigen. The policy can run at 100 Hz as designed both on a small personal computer (PC) with i5-1135G7 and Raspberry 3 on Unitree Go1.

5.2 Controllable locomotion with gait transitions

The proposed control pipeline is deployed on the hardware as a validation of the overall method. In this sub-section, we evaluate the control performance to track commanded locomotion with gait transitions. Fig. 4a shows the experiment of command tracking on the hardware. The animation state machine generates reference motion for turning in changing direction according to user commands, and the robot can follow those commands with the policy. More results of command tracking on the hardware can be found in Video S1 of the ESM. Fig. 5 shows the results of gait transitions among stand, trot, and pace sequentially. The animation state machine extracts the rules for gait transitions from motion capture data and reproduces those transitions automatically according to the speed command. For rapid starts from the standing state, animals tend to use trot gait, in which the movements of

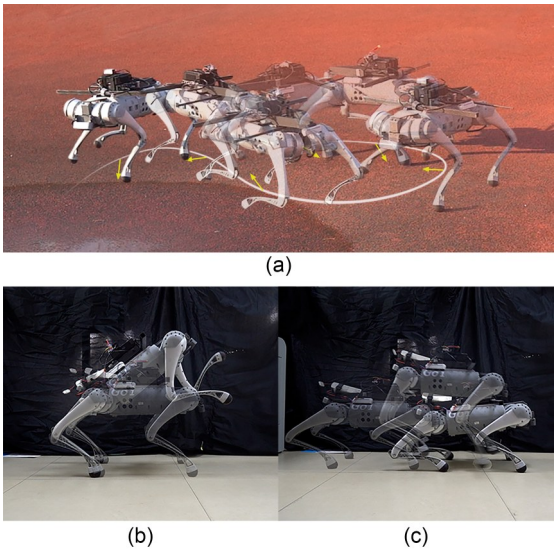

Fig. 4 Hardware experiment for command tracking from motion capture (a), high kick from sketch (b), and forward jump from optimization (c)

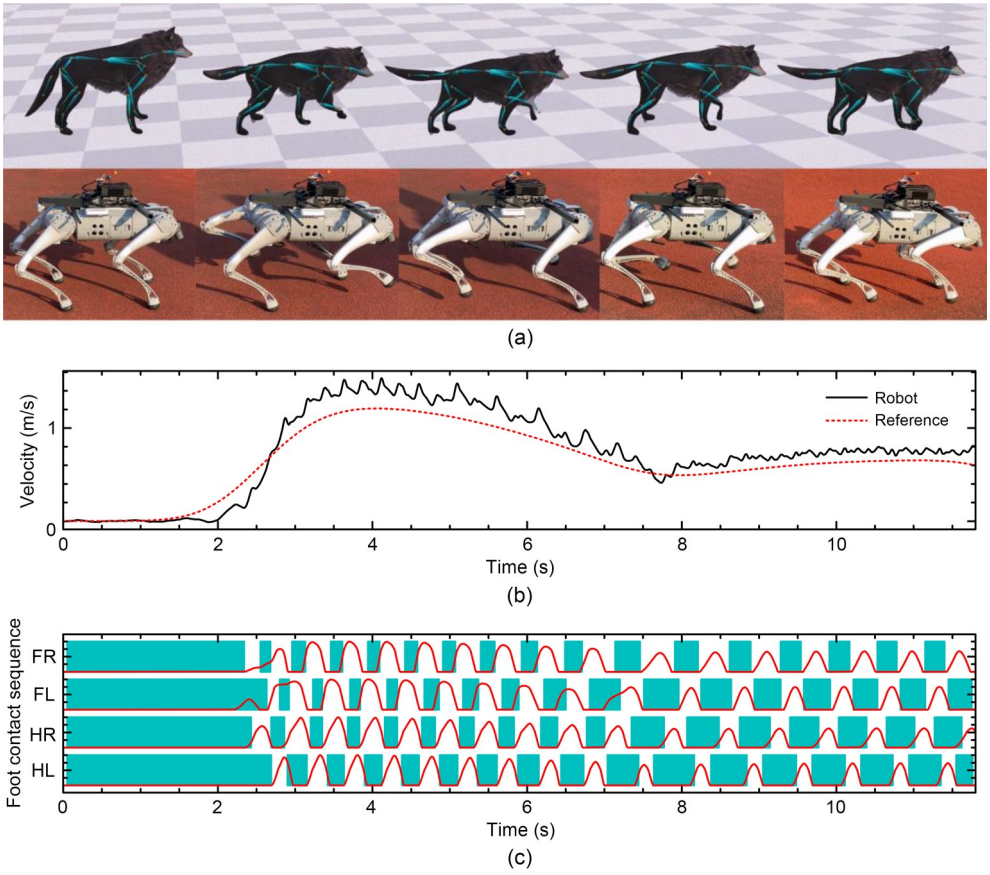

Fig. 5 Locomotion containing stand, trot gait, and pace gait sequentially: (a) comparison between the synthesized motion and the controlled hardware; (b) comparison of forward trunk velocities between the robot and the reference motion from simulation; (c) results of the foot contact sequence of the robot. The blocks represent the actual contact status and the curves represent the height of four feet in the reference. FR, FL, HR, and HL represent front right, front left, hind right, and hind left, respectively

diagonal legs are approximately in sync, and for low-speed locomotion, animals tend to use pace gait, in which the movements of legs on the same lateral side are in sync. With our controller, the natural gait transitions are carried out on the robot. Besides, Fig. 5 also presents the velocity and foot contact sequence, which are both major characteristics of the animation. The data from simulation shows that the robot can track the reference velocities well and reproduce the same gaits as the animation. Hardware experiments for gait transitions can be found in Video S2 of the ESM.

5.3 Motions from sketch and optimization

In addition to motion synthesis, motions from sketch and optimization are also learned and deployed on the hardware. For sketch motion, the robot is designed to raise the body in pitch direction up to 40° and use one of the forelegs to reach a target height up to 0.50 m. The reference motion is a manually scripted kinematics trajectory without physics constraints. For optimization, the robot is designed to jump forward in 0.45 m. Floating base model and direct collation methods are used for numerical optimization. The optimized trajectory is dynamically feasible but open-loop. Fig. 4 shows some snapshots for the hardware experiments, demonstrating the effectiveness of our approach. As mentioned above, the control policy for those two tasks is the same one as the previous subsection. More detailed results can be found in Videos S3 and S4 of the ESM.

5.4 Learning performance

Reference trajectories from sketch and motion capture are kinematical trajectories and cannot be used for control directly. The solution of trajectory optimization contains dynamic information but is open-loop and based on a simplified mode. The proposed method can form a dynamic-feasible close-loop controller for all three types of skills based on those inaccurate trajectories. Fig. 6 shows the learning performance for multiple skills and a single skill. Compared to policies for a specific skill, the policy trained for all three types of skills can achieve similar performance for each skill. Learning multiple skills only slows down the learning speed but does not compromise the final performance.

5.5 Effect of future reference trajectory

Since it is very difficult to directly measure the correlation among high-dimensional data, a neural network is used to describe the predictability of the reference trajectory. The neural network fits the dataset using the first N steps of the reference trajectory as input and the last M steps as output. The specific behavior of this ad hoc neural network is not identical to the controller, and the prediction model itself has no effect on control performance. However, the prediction error from the neural network can be treated as a semi-quantitative indicator of the predictability on a certain interval of the trajectory. To fully reflect the correlation among reference trajectories, the model is tuned to the best accuracy. For illustration, a multi-layer preceptor is trained to predict the next 30 steps based on two steps, and the prediction error and results for the reference trajectory used in Fig. 5 are shown in Fig. 7. For common gaits, e.g., at T=7 or 13 s, the prediction error is low, and the trained multilayer perceptron (MLP) can accurately predict the future time steps, as shown in Fig. 7b. In the standing state, the prediction error is even lower. However, during gait transitions, the prediction

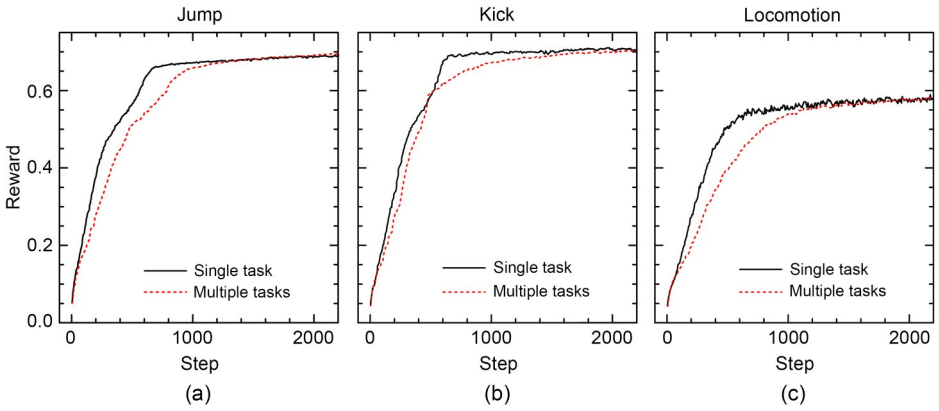

Fig. 6 Comparison of learning curves for multiple-skill policy and single-skill policy: (a) performance for jump task; (b) performance for kick task; (c) performance for locomotion task

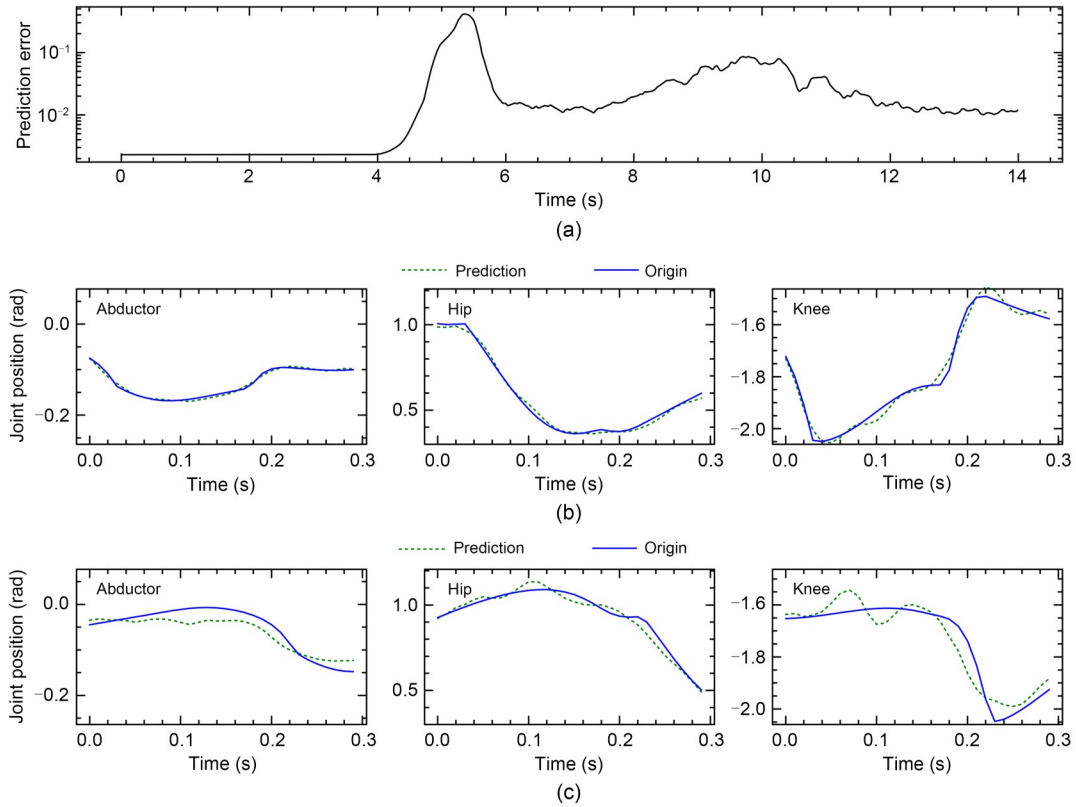

Fig. 7 Illustration for the measurement of predictability: (a) prediction error over time in the task from stand to trot gait and ending in pace gait (the first transition is around T=5 s and the second is around T=10 s); positions of the abductor, hip, and knee joints of the front right leg selected to illustrate the comparison of the predicted trajectory and the original trajectory at T=13.41 s (b) and T=9.92 s (c), as examples of periodic motions and transitions, respectively

error rises, corresponding to low predictability, and the prediction results of T=9.9 s are shown in Fig. 7c as an example. The MLP can no longer predict future reference trajectories accurately.

To evaluate the effect of the predictability of the trajectory on the final control performance, controllers with different numbers of steps in the input are trained for comparison. The zero-padding technique is used to keep all controllers in the same structure. For control policies with less than 32 steps of reference trajectories, the additional dimensions are set to zero. Fig. 8 shows the root mean squared error (RMSE) of velocities, joint positions, and body postures of different lengths of reference trajectories at different levels of predictability in locomotion tasks. The velocity error stands for the performance of command tracking, which is the major performance indicator in locomotion tasks. It is not affected by the length of the reference trajectory in highly predictable cases like periodic motions, while in cases with a low level of predictability like gait transitions and velocity changing, there is a significant decrease in tracking errors as the length of the reference trajectory increases. The joint position error represents the similarity between the reproduced motion and the reference motion. The similarity slightly increases as the length of the reference trajectory increases regardless of predictability. The body posture error measures the difference of the attitude between the reproduced motion and reference motion; it can be obtained through $\varepsilon_R=\boldsymbol{r}_g\cdot\hat{\boldsymbol{r}}_g$, in which $\boldsymbol{r}_g$ represents the actual body posture and $\hat{\boldsymbol{r}}_g$ represents the reference trajectories. There is no significant effect on the balance of the robot from the length of reference trajectory or the level of predictability in the test cases because the variation of body posture on the dataset of locomotion is small compared to joint position and velocity. Overall, the statistical results show that for trajectories that can be easily predicted, longer reference trajectory in input has no contribution to the control performance. The controller itself can learn to predict the future trajectory and take actions according to a long reference trajectory. For unpredictable motions, like sudden start or gait transitions, the

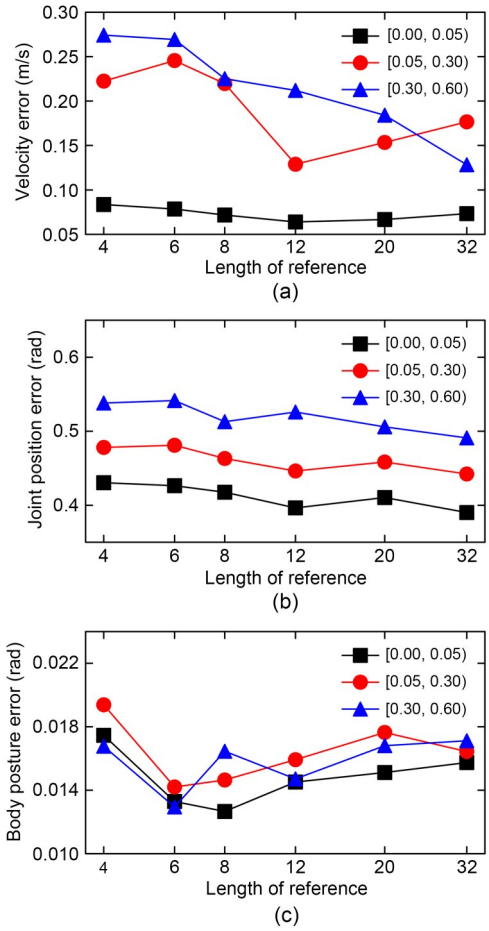

Fig. 8 Effect of the length of reference trajectory on velocity error (a), joint position error (b), and body posture error (c). The results are measured at three levels of predictability: highly predictable (prediction error <0.05), moderately predictable (prediction error from 0.05 to 0.30), and hardly predictable (prediction error from 0.30 to 0.60)

controller can no longer predict the future reference trajectory. Controllers with less reference trajectory can only act according to the given length of reference or on some incorrect guesses, resulting in poor control performances including large tracking errors or unsteady body posture.

For sketch and optimized motion, the trajectories are highly predictable except for the moment when the action is first initiated from a static state. Fig. 9 shows the effect of future reference trajectories on the task of high kick and compares the performance of two policies with two and 16 steps of reference trajectory. Zero-padding is also used in this experiment. At the starting point, the controller with 16 steps of future reference trajectory takes actions earlier than the controller with only two steps of future reference trajectory, making good preparation for the incoming high dynamic motion. The comparison of the body pitch angle represents the high-level tracking performance in the high kick task. Since the trajectory is simple and highly predictable, the length of reference trajectory has no significant effect on high-level performance.

6 Discussion and conclusions

In this study, we present an RL-based control pipeline for generic motor skills for quadruped robots. The proposed pipeline consists of motion synthesis and RL-based motion imitation and uses reference trajectories with basic kinematics information as the interface

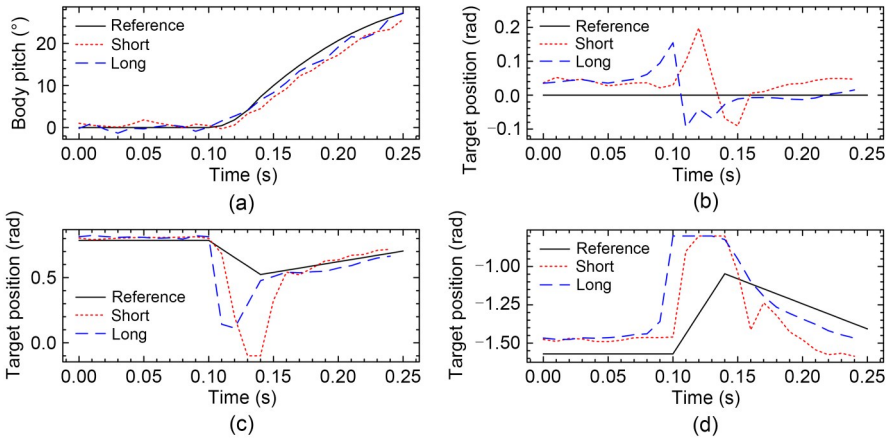

Fig. 9 Detail comparison for control policy with different lengths of reference trajectory for the task of high kick from sketch. The body pitch angle (a) and the target positions of abductor (b), hip (c), and knee (d) (the action from the policy) on the front right legs are shown in comparison to the reference. The label "Short" stands for the results of the policy with two steps and "Long" for 16 steps

between those two major modules. The animation state machine generates motions according to user commands. Motor skills from various sources, including motion capture, sketch, and optimization, can be tracked by a single RL controller, and be reproduced on the hardware.

The reference trajectory in the input of the controller is key to the control performance. From simulation experiments, we notice the ability of the controller to predict future motion based on a few steps. This capability makes the RL controller able to work with very few steps of reference trajectory, which is a situation infeasible for MPC. The effect of the future reference trajectory length on control performance is mediated by the predictability of the trajectory. Longer reference trajectories lead to gains in control performance only when they can provide information that the controller cannot predict.

Although only simple results for quadruped robots are presented, the proposed pipeline is not limited to quadruped robots, and not limited to a specific motion synthesis tool. In future studies, it can be extended to humanoid robots and reproduce more complex motions with the most advanced motion synthesis techniques.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 12132013).

Author contributions

Wei YANG and Hongtao WANG initiated the project. Hongtao WANG created the experimental protocols. Yecheng SHAO did the experiments and processed the corresponding data. Hongtao WANG organized the manuscript, and revised and edited the final version. All authors contributed to the discussion.

Conflict of interest

Yecheng SHAO, Yongbin JIN, Zhilong HUANG, Hongtao WANG, and Wei YANG declare that they have no conflict of interest.

References

Agarwal A, Kumar A, Malik J, et al., 2022. Legged locomotion in challenging terrains using egocentric vision. Proceedings of the 6th Conference on Robot Learning, p.403-415.

Clavet S, 2016. Motion matching and the road to next-gen animation. Game Developers Conference.

Dao J, Duan HL, Green K, et al., 2021. Pushing the limits: running at 3.2 m/s on cassie. Dynamic Walking Meeting.

Escontrela A, Peng XB, Yu WH, et al., 2022. Adversarial motion priors make good substitutes for complex reward functions. IEEE/RSJ International Conference on Intelligent Robots and Systems, p.25-32. https://doi.org/10.1109/IROS47612.2022.9981973

Fuchioka Y, Xie ZM, van de Panne M, 2023. Opt-mimic: imitation of optimized trajectories for dynamic quadruped behaviors. International Conference on Robotics and Automation.

Hill A, Raffin A, Ernestus M, et al., 2018. Stable baselines. GitHub. https://github.com/hill-a/stable-baselines

Holden D, Komura T, Saito J, 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics*, 36(4):42. https://doi.org/10.1145/3072959.3073663

Holden D, Kanoun O, Perepichka M, et al., 2020. Learned motion matching. *ACM Transactions on Graphics*, 39(4):53. https://doi.org/10.1145/3386569.3392440

Huang XY, Li ZY, Xiang YZ, et al., 2022. Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning. arXiv:2210.04435. https://arxiv.org/abs/2210.04435

Hwangbo J, Lee J, Hutter M, 2018. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895-902. https://doi.org/10.1109/LRA.2018.2792536

Ji G, Mun J, Kim H, et al., 2022. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2): 4630-4637. https://doi.org/10.1109/LRA.2022.3151396

Jin YB, Liu XW, Shao YC, et al., 2022. High-speed quadrupedal locomotion by imitation-relaxation reinforcement learning. *Nature Machine Intelligence*, 4(12):1198-1208. https://doi.org/10.1038/s42256-022-00576-3

Kang D, Zimmermann S, Coros S, 2021. Animal gaits on quadrupedal robots using motion matching and model-based control. IEEE/RSJ International Conference on Intelligent Robots and Systems, p.8500-8507. https://doi.org/10.1109/IROS51168.2021.9635838

Lee J, Hwangbo J, Wellhausen L, et al., 2020. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986. https://doi.org/10.1126/scirobotics.abc5986

Li CH, Vlastelica M, Blaes S, et al., 2022. Learning agile skills via adversarial imitation of rough partial demonstrations. Proceedings of the 6th Conference on Robot Learning, p.342-352.

Ling HY, Zinno F, Cheng G, et al., 2020. Character controllers using motion VAEs. *ACM Transactions on Graphics*, 39(4):40. https://doi.org/10.1145/3386569.3392422

Miki T, Lee J, Hwangbo J, et al., 2022. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822. https://doi.org/10.1126/scirobotics.abk2822

Peng XB, Abbeel P, Levine S, et al., 2018. DeepMimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):143. https://doi.org/10.1145/3197517.3201311

Peng XB, Chang M, Zhang G, et al., 2019. MCP: learning

composable hierarchical control with multiplicative compositional policies. Proceedings of the 33rd International Conference on Neural Information Processing Systems, article 331.

Peng XB, Coumans E, Zhang TN, et al., 2020. Learning agile robotic locomotion skills by imitating animals. Proceedings of the 14th Robotics: Science and Systems XVI.

Peng XB, Ma Z, Abbeel P, et al., 2021. AMP: adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40(4):144. https://doi.org/10.1145/3450626.3459670

Peng XB, Guo YR, Halper L, et al., 2022. ASE: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4):94. https://doi.org/10.1145/3528223.3530110

Schulman J, Wolski F, Dhariwal P, et al., 2017. Proximal policy optimization algorithms. arXiv:1707.06347. https://arxiv.org/abs/1707.06347

Shao YS, Jin YB, Liu XW, et al., 2022. Learning free gait transition for quadruped robots via phase-guided controller. *IEEE Robotics and Automation Letters*, 7(2):1230-1237. https://doi.org/10.1109/LRA.2021.3136645

Siekmann J, Valluri S, Dao J, et al., 2020. Learning memory-based control for human-scale bipedal locomotion. Proceedings of the 14th Robotics: Science and Systems XVI.

Siekmann J, Green K, Warila J, et al., 2021a. Blind bipedal stair traversal via sim-to-real reinforcement learning. Proceedings of the 14th Robotics: Science and Systems XVII.

Siekmann J, Godse Y, Fern A, et al., 2021b. Sim-to-real learning of all common bipedal gaits via periodic reward composition. IEEE International Conference on Robotics and Automation, p.7309-7315. https://doi.org/10.1109/ICRA48506.2021.9561814

Starke S, Zhang H, Komura T, et al., 2019. Neural state machine for character-scene interactions. *ACM Transactions on Graphics*, 38(6):209. https://doi.org/10.1145/3355089.3356505

Starke S, Mason I, Komura T, 2022. DeepPhase: periodic autoencoders for learning motion phase manifolds. *ACM Transactions on Graphics*, 41(4):136. https://doi.org/10.1145/3528223.3530178

Vollenweider E, Bjelonic M, Klemm V, et al., 2022. Advanced skills through multiple adversarial motion priors in reinforcement learning. arXiv:2203.14912. https://arxiv.org/abs/2203.14912

Xie ZM, Clary P, Dao J, et al., 2019. Learning locomotion skills for cassie: iterative design and sim-to-real. Proceedings of the 3rd Annual Conference on Robot Learning, p.317-329.

Zhang H, Starke S, Komura T, et al., 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics*, 37(4):145. https://doi.org/10.1145/3197517.3201366

Electronic supplementary materials

Videos S1–S5