2021 6th IEEE International Conference on
Advanced Robotics and Mechatronics
July 3-5, 2021, Chongqing, China

2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM) | 978-1-6654-3909-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICARM52023.2021.9536204

Learning Smooth and Omnidirectional Locomotion for Quadruped Robots

Jiaxi Wu[1], Chen'an Wang[2], Dianmin Zhang[1], Shanlin Zhong[1], Boxing Wang[1], and Hong Qiao[3]

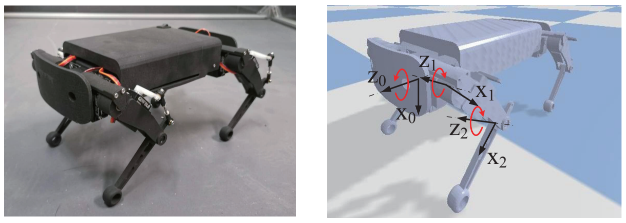
***Abstract*— It often takes a lot of trial and error to get a quadruped robot to learn a proper and natural gait directly through reinforcement learning. Moreover, it requires plenty of attempts and clever reward settings to learn appropriate locomotion. However, the success rate of network convergence is still relatively low. In this paper, the referred trajectory, inverse kinematics, and transformation loss are integrated into the training process of reinforcement learning as prior knowledge. Therefore reinforcement learning only needs to search for the optimal solution around the referred trajectory, making it easier to find the appropriate locomotion and guarantee convergence. When testing, a PD controller is fused into the trained model to reduce the velocity following error. Based on the above ideas, we propose two control framework - single closed-loop and double closed-loop. And their effectiveness is proved through experiments. It can efficiently help quadruped robots learn appropriate gait and realize smooth and omnidirectional locomotion, which all learned in one model.**

I. INTRODUCTION

The quadruped robots have attracted extensive attention due to their flexible behavior, strong ability to surmount obstacles, and free from terrain constraints. At present, there are many excellent quadruped robots, such as Spot Mini and Laikago. Additionally, there are many excellent open-source quadruped robot projects, such as Mini Cheetah and Minitaur. These open-source quadruped robot projects have greatly promoted the development of the quadruped robot field.

In the above project, brushless motors are used as the actuated joints, which has large output torques and high control accuracy. However, it increases the cost and hardware base of quadruped robots. Therefore, many open source projects use low-cost and easy-to-operate steer motors, such as Spot Micro and Pupper. Sbot we designed uses steer motors as actuated joints.

The steer motors can only be position controlled, which brings some trouble to their control accuracy [1]. However, it is more convenient to do reinforcement learning algorithms for quadruped robots actuated by steer motors. The network outputs joint angles directly to control the quadruped robots to achieve various gaits, without considering complex force control.



(a) real model (b) simulatd model

Fig. 1. The quadruped robot designed by ourselves - Sbot. (x_0, z_0) is the hip (abduction/adduction) coordinate system, (x_1, z_1) is the hip (forward/backward) coordinate system, (x_2, z_2) is the knee coordinate system. And the red arrows are the positive directions of rotary joints.

However, if directly outputting the joint angles, reinforcement learning requires large-scale attempts to enable the agent to learn appropriate gaits. However, experiments have found that it is often hard to achieve, which requires delicate reward functions. Even so, it is still challenging to converge to an acceptable result. Usually, the agent can only learn simple motion, but its locomotion is slow and inefficient, far from the gaits of real quadrupeds.

In this paper, the referred trajectory and inverse kinematics are integrated into the training process of reinforcement learning as the prior knowledge to reduce network convergence difficulty. The agent only needs to output the parameters of the referred trajectory and the joint angles correction. Then the preplanned trajectory can automatically generate a series of foot positions to guide the joint movement.

By combining the referred trajectory with reinforcement learning, the network only needs to search around the referred trajectory to find a better solution to achieve smooth locomotion. Therefore, the convergence difficulty of reinforcement learning can be reduced, and it is easier for quadruped robots to learn smooth locomotion.

Additionally, the transformation loss, including symmetry loss and translation loss, is proposed to constrain the agent to behave similarly on four legs, which reduces the search space and helps the quadruped robots learn gaits more similar to real quadrupeds.

However, if the model trained by reinforcement learning is used directly, its control accuracy is low and cannot follow the desired velocity. When realizing omnidirectional locomotion in the plane, including axial velocity, lateral velocity, and rotational velocity, it is difficult to ensure that the quadruped robots follow the desired velocity due to the

[1]J.Wu, D.Zhang, S.Zhong, and B.Wang are with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China `wujiaxi2019@ia.ac.cn`

[2]C.Wang is with School of Mechanical Engineering, University of Science and Technology Beijing, Beijing 100083, China

[3]H.Qiao is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and also with Center for Excellence in Brain Science and Intelligence Technology, Institute of Neuroscience, Chinese Academy of Sciences, 200031, China `hong.qiao@ia.ac.cn`

large search space. Therefore, a PD controller is added into the model after training to help the agent eliminate the speed following error.

The main contributions of this paper are:

- With the help of the referred trajectory, the quadruped robots learn smooth and omnidirectional locomotion by reinforcement learning, which is learned in one model rather than multiple studies and model fusion.
- The transformation loss, including symmetry loss and translation loss, is proposed to improve the locomotion performance and make the learned locomotion more similar to real quadrupeds.
- Without affecting the learned gait, the PD controller is integrated into the trained model to reduce the velocity following error and improve the control accuracy.

II. RELATED WORK

With the breakthrough in games [2] and go [3], deep reinforcement learning has gradually attracted the interest of many researchers [4] [5]. And they are no longer satisfied with virtual applications in games, but hope to apply it to other tasks such as robotic arm grasping and quadruped robot gaits.

J. Hwangbo et al. used reinforcement learning to make ANYmal learn to recover from a fall [6], walk on uneven terrain [7]. If using traditional control and optimization methods, those tasks would be very complicated. But reinforcement learning helped the robot learn how to accomplish these tasks better and improved the success rate.

However, learning the smooth gait of quadruped robots is a challenging problem. A. Iscen et al. used the trajectory generator to help Minitaur learn to walk and bound [8]. S. Kolathaya et al. applied smooth trajectories on the feet by Bézier curves [9]. Z. Xie et al. used the referred motion to help Cassie walk forward [10].

Imitation learning is another way to solve this problem. X. B. Peng et al. used imitation learning to enable Laikago to learn various gaits, including pace and trot, from real dogs [11]. With the help of imitation learning, Z. Xie et al. made Cassie learn to move forward and backward [12]. However, they all used different networks to realize their various locomotion. Although [12] realized a variety of locomotion modes, the final model was combined by multiple models, rather than just one model.

Dynamic methods are integrated into the training framework to help reinforcement learning to learn a better solution. Y. Yang et al. used model predictive control, in conjunction with model-based reinforcement learning, to make Minitaur learn to walk fast [13]. G. A. Castillo et al. used hybrid zero dynamics to let Cassie learn to move forward and sideways at desired velocity [14].

III. OMNI-DIRECTIONAL LOCOMOTION CONTROLLER

A. Referred Trajectory

1) Locomotion Strategy: For quadruped robots driven by steer motors, the general control method is trajectory planning with inverse kinematics. Firstly, a closed trajectory, which can be divided into the swing and stance phases, is applied to the quadruped robot's feet. Then the joint angles are solved by inverse kinematics according to the current target foot position. Finally, the joint angle is converted to PWM and applied to steer motors.

As shown in Fig. 2, the closed trajectory can be defined by forward length S_x, lateral length S_y, lift height H, gait period T, swing ratio ϕ (swing phase time / total time), current time t, and so it can be expressed as $traj(S_x, S_y, H, T, \phi, t)$.

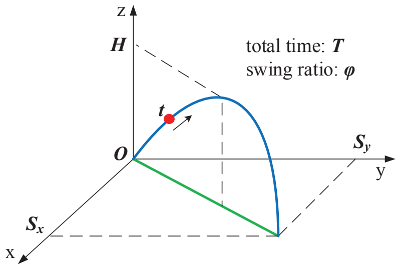
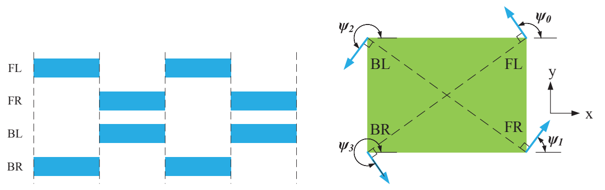


Fig. 2. Closed trajectory defined by $(S_x, S_y, H, T, \phi, t)$. Blue line: swing phase; green line: stance phase.

The feet phase of trot is demonstrated in Fig. 3(a). For forward locomotion, the four legs' trajectories can be set as $traj(L_x, 0, H, T, 0.5, t)$; for lateral locomotion, the four legs' trajectories can be set as $traj(0, L_y, H, T, 0.5, t)$; for rotation locomotion, the four legs' trajectories can be set as $traj(R\cos(\psi_i), R\sin(\psi_i), H, T, 0.5, t)$, where ψ_i is defined in Fig. 3(b). Therefore when lift height H and gait period T are set as constant, the locomotion of quadruped robots can easily be controlled by (Lx, Ly, R, t).

2) Single Closed-loop: The locomotion of quadrupeds is a periodic process, which needs to form a closed-loop trajectory at the end of the feet. It is hard to learn directly from reinforcement learning. Similar to [8], a closed trajectory defined above is used to help the convergence of reinforcement learning. As shown in Fig. 4, the policy network inputs desired velocity $v_d = (v_{xd}, v_{yd}, v_{yawd})$, quadruped robot's sensor state s, referred trajectory's parameters ζ, outputs locomotion parameters l and the correction of 12 joint angles a_c. The refer joint angles a_r are obtained by the referred trajectory and inverse kinematics. They are added to a_c to obtain the desired joint angles a_d assigned to the steer motors.

With the help of referred trajectory, the network only needs to search for optimal solutions near the referred trajectory.



(a) leg phase of trot (b) phase offset ψ_i in rotation

Fig. 3. Gait description. The blue bar is the swing phase and the blank is the stance phase, i.e. the swing ratio ϕ is 0.5. FL: front left leg; FR: front right leg; BL: back left leg; BR: back right leg.

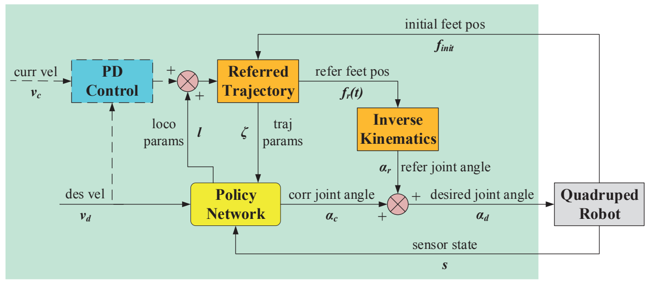


Fig. 4. Single closed-loop control framework. $l = (L_x, L_y, R)$, $\zeta = (Lx, L_y, R, H, T, \phi, t)$.

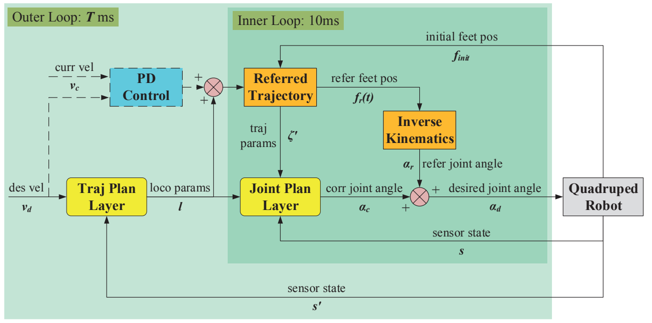


Fig. 5. Double closed-loop control framework. $\zeta' = (H, T, \phi, t)$.

It is equivalent to narrowing the search space, so it is easier to get better results and accomplish smooth locomotion.

3) Double Closed-loop: Meanwhile, a double closed-loop framework is proposed. As shown in Fig. 5, trajectory planning (outer loop) and joint angles correction (inner loop) are divided into two parts and trained, respectively.

The trajectory planning layer is responsible for generating locomotion parameters l under the guidance of desired velocity v_d and robot state s'. It makes decisions at the beginning of each gait cycle. The joint angles correction layer is responsible for generation joint angles corrections under the trajectory parameters $\zeta = (l, \zeta')$, robot state s. It executes once each simulation step.

Specially, the networks of inner and outer rings are trained together. Therefore it only needs to be trained once to get a controller that achieves omnidirectional locomotion.

B. Transformation Loss

The network's convergence can be guaranteed with the help of referred trajectory, but the learned gait is not symmetry and consistent, which is different from real quadrupeds. Inspired by [15], we propose transformation loss to increase the similarity of the movements of four legs.

The transformation loss consists of two parts: symmetry loss and translation loss. As [15], the symmetry loss is defined as

$$L_s(\theta) = \sum_{i=0} ||\pi_\theta(s_i) - \Psi_a(\pi_\theta(\Psi_o(s_i)))||^2 \tag{3}$$

where $\Psi_a(\cdot)$ and $\Psi_o(\cdot)$ do the mirror operations for the action and state. Similar to symmetry loss, the translation loss is defined as

$$L_t(\theta) = \sum_{i=0} ||\pi_\theta(s_i) - \Phi_a(\pi_\theta(\Phi_o(s_i)))||^2 \tag{4}$$

where $\Phi_a(\cdot)$ and $\Phi_o(\cdot)$ do the translation operations for the action and state.

When the symmetry loss is zero, the left and right legs will do the mirror action under the mirror state. Moreover, when the translation loss is zero, the front and back legs will do the consistent action under the consistent state. By combining symmetry loss and translation loss, the locomotion of four legs becomes more similar, just like real quadrupeds.

Considering the difficulty of optimization, the symmetry loss L_s and translation loss L_t are added to the total loss function as soft constraints. Then the loss function of reinforcement learning algorithm can be expressed as

$$L(\theta) = L_{RL}(\theta) + \alpha L_s(\theta) + \beta L_t(\theta) \tag{5}$$

where $L_{RL}(\theta)$ is the loss function of reinforcement learning, α and β are the weights for symmetry loss L_s and translation loss L_t respectively.

C. PD Controller

The model obtained from reinforcement learning can achieve complex tasks, but its control accuracy is low. The PD controller is fused into the control framework to reduce the velocity following error, as demonstrated in the dashed parts in Fig. 4 and Fig. 5. The formula of the PD controller is

$$l_{pd} \mathrel{+}= k_p * (v_d - v_c) + k_d * (0 - \dot{v_c}) \tag{6}$$

where v_d and v_c are the desired velocity and current velocity, $\dot{v_c}$ is the acceleration of quadruped robot's body. Then l_{pd} is added with the network's outputs l to obtain the final locomotion parameters to the referred trajectory.

The PD controller mainly plays the role of correction, and the main control work is achieved by the model learned by reinforcement learning. When there is a small deviation between the desired velocity and current velocity, the PD controller is responsible for generating a small increment to eliminate the deviation.

It is important to note that the PD controller is added to the control framework only during the evaluation, and there is no PD controller during the training process.

D. Reinforcement Learning Settings

The reinforcement learning used in this paper is PPO [16]. However, the control framework proposed in this paper is applicable not only to PPO, but also to other RL algorithms.

1) State Space: In the single closed-loop framework, there are three main parts of the state: the first part is the sensor data of the robot, including body height, posture (represented by Euler), linear velocity, angular velocity, joint angles, and angular velocities. The second is the desired velocity v_d and velocity deviation Δv. The third is the referred trajectory parameters ζ.

The double closed-loop framework's state space is similar to single closed-loop, but there are two agents trained together. The inner loop's state space is the same as the single closed-loop framework. In contrast, the outer loop's state space removes joint angles, angular velocities, and referred trajectory parameters.

2) Action Space: The single closed-loop framework's action space is shown in Fig. 4, including the locomotion parameters l and joint angles correction α_c. In the double closed-loop framework, the inner loop's action space is the joint angles correction α_c, while the outer loop's action space is the locomotion parameters l.

3) Reward Function: The quadruped robot is expected to locomote at the desired velocity v_d while keeping its balance and making its locomotion smooth. Therefore the reward function of the single closed-loop framework is as follows:

$$r_i = \omega_1 r_{vl} + \omega_2 r_{va} + \omega_3 r_{bh} + \omega_4 r_{rp} \tag{7}$$

where r_{vl} is the linear velocity deviation (set the desired z-axis velocity as 0), r_{va} is the angular velocity deviation (set the desired roll and pitch velocity as 0), r_{bh} is the body height deviation, r_{rp} is the square root of roll and pitch.

In the double closed-loop framework, the inner loop reward and outer loop reward are

$$r_{in} = \omega_1 r_{vz} + \omega_2 r_{vrp} + \omega_3 r_{bh} + \omega_4 r_{rp} \tag{8}$$

$$r_{out} = \omega_1 r_{vl} + \omega_2 r_{va} \tag{9}$$

where r_{vz} is the z-axis velocity deviation, r_{vrp} is the roll and pitch velocity deviation. It should be noted that in the outer loop, the current body velocity is the average velocity of the entire gait cycle, rather than the single step's velocity.

E. Performance Score

Some performance scores are proposed to evaluate the trained model. Since the quadruped robot is expected to locomote at the desired velocity, the deviation and standard deviation between the desired velocity v_d and the current velocity v_c are used to measure the following performance.

$$\delta v_x = \frac{1}{MN}\sum_{i=1}^{M}\sum_{s=1}^{N}||v_{xd}(i,s) - v_{xc}(i,s)|| \tag{10}$$

$$\delta v_y = \frac{1}{MN}\sum_{i=1}^{M}\sum_{s=1}^{N}||v_{yd}(i,s) - v_{yc}(i,s)|| \tag{11}$$

$$\delta v_{yaw} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{s=1}^{N}||v_{yawd}(i,s) - v_{yawc}(i,s)|| \tag{12}$$

where i is the current episode, s is the current step, M is the number of test episodes, and N is the number of steps per episode. Similarly, the standard deviation between v_d and v_c is defined as

$$\sigma v_x = \frac{1}{M}\sum_{i=1}^{M}\sqrt{\frac{1}{N}\sum_{s=1}^{N}(\Delta v_x(i,s) - \overline{\Delta v_x}(i))^2} \tag{13}$$

$$\sigma v_y = \frac{1}{M}\sum_{i=1}^{M}\sqrt{\frac{1}{N}\sum_{s=1}^{N}(\Delta v_y(i,s) - \overline{\Delta v_y}(i))^2} \tag{14}$$

$$\sigma v_{yaw} = \frac{1}{M}\sum_{i=1}^{M}\sqrt{\frac{1}{N}\sum_{s=1}^{N}(\Delta v_{yaw}(i,s) - \overline{\Delta v_{yaw}}(i))^2} \tag{15}$$

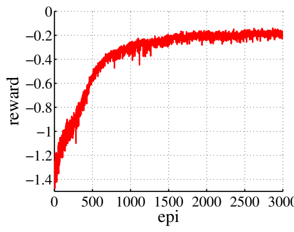
where $\Delta v(i,s) = v_d(i,s) - v_c(i,s)$, and $\overline{\Delta v}(i)$ is the average deviation velocity in one episode. The smaller the parameters defined above, the better the performance of the quadruped robot following the desired velocity v_d.

In addition, the standard deviation of z-axis velocity σv_z, roll velocity σv_{rol}, and pitch velocity σv_{pit} is proposed to measure the balance of the quadruped robot. Their formulas are similar to σv_x, σv_y, and σv_{yaw}, so they are not repeated here. The smaller σv_z, σv_{rol} and σv_{pit} are, the less quadruped robot's body wobbles during locomotion, which means it locomotes more smoothly.

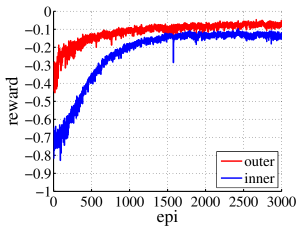
IV. EXPERIMENTAL RESULTS

A. Omini-directional Locomotion

The simulation physics engine used in this paper is PyBullet [17]. Two control frameworks are used to learn smooth and omnidirectional locomotion, respectively. They are trained for 3000 episodes, each episode lasts for 40s or the quadruped robot falls down, and the simulation step is 10ms. The lift height H and gait cycle T are set as 0.05m and 0.4s, respectively. And the reward function curves are demonstrated in Fig. 6.



(a) single closed-loop



(b) double closed-loop

Fig. 6. Reward curve of 3000 episodes.

After training, the agent can control the quadruped robot to locomotion in all directions, including forward, backward, sideways, and rotation. It is worth noting that these locomotion modes are all accomplished in the same model, and those different locomotion modes can be combined to achieve complex locomotion. The video is available in `https://youtu.be/37G5_ShUCJA`.

B. Performance Comparison

To evaluate the performance of the trained models, we test 100 times, each test lasts for 40s, and the desired velocity was

TABLE I

EXPERIMENT RESULTS. SCL: SINGLE CLOSED-LOOP, DCL: DOUBLE CLOSED-LOOP, TL: TRANSFORMATION LOSS, PD: PD CONTROLLER. THE LINE IN BOLD IS THE BEST ONE.

	δv_x	δv_y	δv_{yaw}	σv_x	σv_y	σv_{yaw}	σv_z	σv_{rol}	σv_{pit}
SCL	0.007	0.005	0.049	0.030	0.025	0.176	0.020	0.297	0.233
SCL+TL	0.008	0.005	0.045	0.026	0.021	0.176	0.026	0.165	0.146
SCL+TL+PD	**0.000**	**0.001**	**0.000**	**0.022**	**0.021**	**0.162**	**0.028**	**0.265**	**0.209**
DCL	0.007	0.007	0.044	0.054	0.049	0.331	0.033	0.364	0.307
DCL+TL	0.012	0.005	0.031	0.040	0.053	0.303	0.029	0.208	0.208
DCL+TL+PD	0.000	0.001	0.002	0.045	0.055	0.312	0.032	0.263	0.262

TABLE II

REFERRED TRAJECTORY COMPARISON. TRAINED BY SCL + TL, EVALUATED BY SCL + TL + PD. THE LINE IN BOLD IS BASELINE.

	δv_x	δv_y	δv_{yaw}	σv_x	σv_y	σv_{yaw}	σv_z	σv_{rol}	σv_{pit}
Sinusoid	**0.000**	**0.001**	**0.000**	**0.022**	**0.021**	**0.162**	**0.028**	**0.265**	**0.209**
Cycloid	0.000	0.002	0.001	0.028	0.035	0.165	0.014	0.266	0.238
Ellipse	0.001	0.002	0.002	0.024	0.024	0.175	0.055	0.298	0.241
Triangle	0.000	0.002	0.001	0.022	0.023	0.153	0.016	0.299	0.250

TABLE III

VARIOUS LIFT HEIGHTS AND GAIT CYCLES. TRAINED BY SCL + TL, EVALUATED BY SCL + TL + PD. THE LINE IN BOLD IS BASELINE.

H	T	δv_x	δv_y	δv_{yaw}	σv_x	σv_y	σv_{yaw}	σv_z	σv_{rol}	σv_{pit}
0.05	**0.4**	**0.000**	**0.001**	**0.000**	**0.022**	**0.021**	**0.162**	**0.028**	**0.265**	**0.209**
0.04	0.40	0.003	0.003	0.003	0.027	0.029	0.156	0.024	0.317	0.246
0.06	0.40	0.001	0.001	0.001	0.022	0.025	0.188	0.030	0.297	0.255
0.05	0.32	0.000	0.001	0.001	0.019	0.020	0.170	0.034	0.253	0.222
0.05	0.48	0.006	0.008	0.010	0.038	0.035	0.181	0.023	0.315	0.256

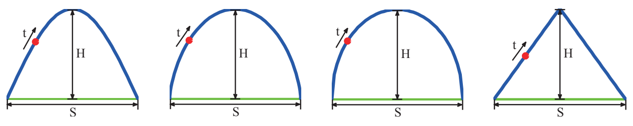
randomly selected for every 10s. The experimental results are shown in Table. I.

1) Two Frameworks: By comparing SCL and DCL in Table. I, it can be found that the model trained by SCL makes the standard deviation of velocity following and maintaining body balance smaller (the last six columns in Table. I). Therefore it makes the locomotion of quadruped robots more smooth.

This is because DCL has two networks, and the outer-loop only considers the velocity following, not the body balance. Sometimes a change in the locomotion parameters is detrimental to the body balance, but DCL cannot consider this relationship. While SCL has only one network, it can consider both velocity following and body balance, so it has a better balance under the same performance of the velocity following.

In Table. I, the referred trajectory we use is sinusoid, as demonstrated in Fig. 7(a). Meanwhile, cycloid, ellipse, and triangle, shown in Fig. 7 are used as referred trajectories. Their results trained by the single closed-loop framework with transformation loss are shown in Table II.

The sinusoid trajectory performance is the best, but the other three trajectories can also help the quadruped robot learn smooth and omnidirectional locomotion. Is means that the referred trajectory is only used to provide a closed curve to assist reinforcement learning to learn the proper locomotion without relying too much on it.



(a) sinuoid (b) cycloid (c) ellipse (d) triangle

Fig. 7. Trajectory comparison. All the trajectories are defined by $(S_x, S_y, H, T, \phi, t)$, and S is the square root of (S_x, S_y).

2) Transformation Loss: After adding transformation loss to the loss function, body balance has been further improved. That is because the transformation loss restricts four legs' actions, making them behave more like real animals, rather than some incongruent actions.

When not using transformation loss, it is sometimes observed that a leg is deliberately shifted inward or outward, which means that it is trapped in a locally optimal solution. These kinds of actions are different from real quadrupeds. After adding the transformation loss, it can be seen that this situation is greatly modified, and it rarely occurs again.

3) PD Controller: Although the single closed-loop framework with transformation loss can achieve the locomotion as we desire, the velocity following error cannot be negligible. Since it is not zero, when the desired velocity v_d is set to $(0, 0, 0)$, the quadruped robot will still drift overtime and deviate from initial position and posture. After adding a PD controller to the original control framework, the velocity following error is eliminated. It overcomes the low control accuracy of the agent acquired by reinforcement learning, as shown in the first three columns of Table. I.

Simultaneously, due to the trained model's robustness, although its locomotion parameters l have been changed, it can still ensure smooth locomotion. Though there is an increment in σv_{rol} and σv_{pit} statistically, there is no difference to the naked eyes.

C. Generalization

Meanwhile, the model trained by SCL + TL (evaluated by SCL + TL + PD) can achieve smooth and omnidirectional locomotion when the trajectory parameters change.

The lift height H and gait cycle T are changed and evaluated directly using the model trained when $H = 0.05m$ and $T = 0.4s$ (the first row in the Table. III).

As seen in Table. III, when the lift height H and gait cycle T increase/decrease by 20%, the quadruped robot can still accomplish smooth and omnidirectional locomotion under the control of the trained model.

However, when $T = 0.48s$, the balance of the quadruped robot's body can still be guaranteed, but there is a certain error in velocity following. This is because as the gait cycle T slows, its maximum velocity decreases, making it difficult to reach the large desired velocity.

V. CONCLUSIONS

In this paper, we propose two control frameworks - single closed-loop and double closed-loop, to help quadruped robots learn smooth and omnidirectional locomotion with the help of referred trajectory. Through only once learning, various locomotion modes are realized according to the desired velocity, including forward, backward, sideways, and rotation. To improve the performance, the transformation loss is used as a restriction on the actions between four legs to help them produce symmetrical and consistent actions. Additionally, a PD controller is fused into the control framework to eliminate the velocity following error in case the quadruped robot drifts over time.

However, our control framework is only verified in simulation. Next, we will carry out experiments on Sbot in reality. To overcome the reality gap, domain randomization will be used to realize sim-to-real [18] [19]. In addition, we only implement trot in this paper, and the other gaits will be implemented based on our framework, including pace, bound, gallop, and so on.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China (2017YFB1300200, 2017YFB1300203), the National Natural Science Foundation of China under Grant 61627808, the National Natural Science Foundation of China under Grant 91648205, the Strategic Priority Research Program of Chinese Academy of Science under Grant XDB32000000.

REFERENCES

[1] J. Zhong and R. Ma, "Dynamic characteristics of a jumping robot with coordinated quadruped legs," *Assembly Automation*, 2019.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] Y. Li, K. P. Tee, R. Yan, and S. S. Ge, "Reinforcement learning for human-robot shared control," *Assembly Automation*, 2019.

[5] H. Qiao, J. Chen, and X. Huang, "A survey of brain-inspired intelligent robots: Integration of vision, decision, motion control, and musculoskeletal systems," *IEEE Transactions on Cybernetics*, 2021.

[6] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[7] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[8] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," in *Proceedings of the Conference on Robot Learning (CoRL)*. PMLR, 2018, pp. 916–926.

[9] S. Kolathaya, A. Joglekar, S. Shetty, D. Dholakiya, A. Sagi, S. Bhattacharya, A. Singla, S. Bhatnagar, A. Ghosal, B. Amrutur *et al.*, "Trajectory based deep policy search for quadrupedal walking," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019, pp. 1–6.

[10] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1241–1246.

[11] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[12] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," *arXiv preprint arXiv:1903.09537*, 2019.

[13] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Proceedings of the Conference on Robot Learning (CoRL)*. PMLR, 2020, pp. 1–10.

[14] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Hybrid zero dynamics inspired feedback control policy design for 3d bipedal locomotion using reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8746–8752.

[15] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–12, 2018.

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

[17] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[18] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[19] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–8.