Locomotion Learning for Quadruped Robot with a New Structure

Zijian Zhao 20308269

September 1, 2025

Abstract

In this article, we construct a bionic machine mouse to research the motion mechanism of some vertebrates. Different from traditional quadruped robot, our machine mouse has more free degrees, especially in spine and head. Besides, the control method of legs is also an important innovative point. It has been widely proved that the PPO algorithm is useful in the field of robotics. Therefore, we want to use this method to train our machine mouse to walk and cross obstacles.

The code is published at https://github.com/RS2002/Robot-mouse

Please notice that the work in this article is not done by myself. The structure of the mouse is designed by others who are at the same lab with me. And I only applied PPO to this robot.

Keywords: PPO, quadruped robot

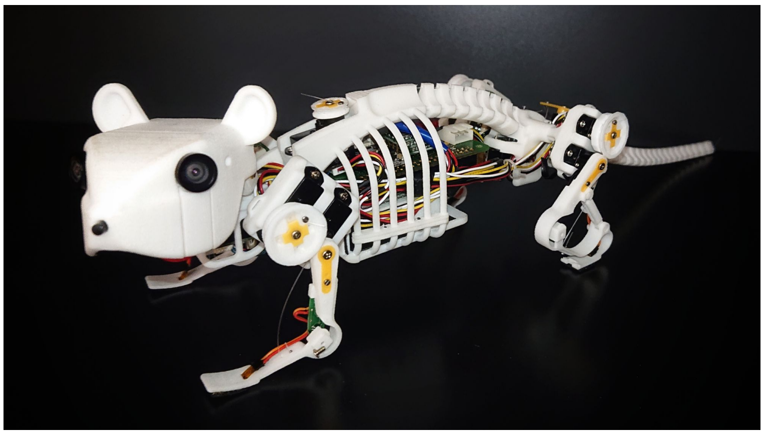


Figure 1: Bio-inspired robotic mouse. This robotic mouse has a fully flexible spine, tail and moveable head.

1 Introduction

The robotics is a significant part of engineering field. Robot can help people to do lots of hard work such as exploring and rescue. What's more, robot can also be

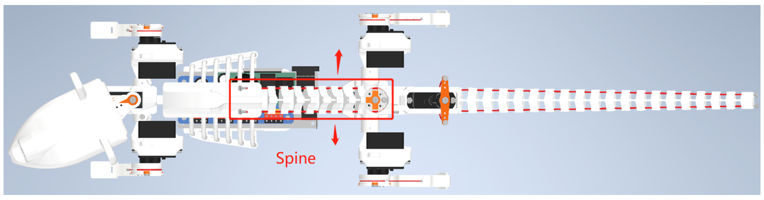


Figure 2: Top View of the mouse: The spine is a soft control structure. A motor controls two lines and drives the spine(or changes the angle of the spine) by pulling one of such two lines.

applied to biology. We can use bionic robot to learn the mechanism of real animal, which is also the reason why we design the machine mouse. Traditional quadruped robots like OpenDog [1] only have free degrees in their joint in legs and foot. On the contrary, the head and spine of our robot are movable. It makes our robot more similar with the mouse in nature. We want to train the machine mouse to walk and cross obstacles so that we can learn more about the motion mechanism of mouse in nature.

When it comes to robot motion problems, reinforcement learning is certainly one of the most important and efficient approaches. In the past, researchers always design exercise strategies on physical models, which needs extensive experience and expertise. However there are countless scenes with different obstacles. It's barely possible for people to find an approach which can adapt to every scenario. Recently, the deep reinforcement-learning(deep RL) [2] [3] [4] has made great process. It has shown excellent performance in the field of locomotion learning for robots. [5] [6] [7] Specifically, Proximal Policy Optimization(PPO) [8] is a vital approach which is simple but effective. Besides, it can be easily parallelized. The PPO algorithm has been widely used in robots [9], whose effectiveness has also been proved.

In this article, we will illustrate the structure of our machine mice at first. Then we will introduce how we apply PPO to the machine mice. At last, we'll use the result of our experiments to show the effectiveness of our approach. Due to the fact of limited time, we only performed the experiments in mujoco simulation environment. In the next step, we will try to apply our model to the real environment.

2 Related Work

PPO is an improved version of Trust Region Policy Optimization(TPRO) [10] which has made some improvement on Conservative Policy Iteration(CPI). They all belongs to Actor-Critic(AC) [11] algorithm. CPI has made some restriction on the normal process of policy ascent. It prevent the circumstance that the update step length is too large, which may lead to converge slowly or even fail to converge. However, it only applies to a specific mixture policy, which causes it is unwieldy and restrictive in practice. To solve this problem, Schulman [10] et al firstly proposed the concept of trust region. They use Minorize-Maximization(MM) and importance sample to improve the CPI. The TPRO can apply to all general stochastic policy classes and guarantee monotonic improvement. It can control the update step length by introducing the KL divergence into optimization target. Unfortunately, this approach also has its own disadvantage that the computational complexity of finding optimum solution is very high. As a result, Schulman [8] proposed clipped surrogate objective(CLIP) and adaptive KL penalty coefficient(KLPEN) to relieve this problem. The new method PPO is just what we applied to our machine mouse.

As mentioned above, CLIP and KLPEN are two similar ways to simplify the TPRO. We take the CLIP method in this article, which is also used more widely. Both of the two approaches use an Artist network to predict the value of every state and a Critic network to decide which action should be chosen at given state. Similar to the Policy Gradient(PG) method, the Critic also train the network by policy accent. Schulman design a new loss function which mix the loss of Artist network and Critic network:

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[-L_t^{CLIP}(\theta) + c_1 L_t^{VF}(\theta) - c_2 S[\pi_\theta](s_t)] \tag{1}$$

where c1, c2 are coefficients, and s is the presentation of state, and S denotes an entropy bonus which can ensure sufficient exploration [12] [13], and L_t^{VF} is the squared-error loss of Artist Network, and L_t^{CLIP} is the main object that PPO wants to maximize

$$L_t^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2}$$

where ϵ is a hyperparameter,and $\hat{A}_t$ is the advantage function ,and the definition of clip function is:

$$clip(a,m,M) = \begin{cases} a & (m < a < M) \\ m & (a \leq m) \\ M & (a \geq M) \end{cases} \tag{3}$$

where $M \geq m$
Besides the definition of the advantage function is as follows

$$\hat{A}_t = \sum_{i=0}^{T-t} (\gamma\lambda)^i \delta_{t+i} \tag{4}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(S_t)$ and γ *and* λ are hyperparameters, and the function V is the value function, and r is the value of reward

3 Robot Model

We've constructed a new type of quadruped robot. In this chapter, we will introduce the front leg control mechanism of our machine mice. And you can find more information like the definition of rear leg and spine at https://www.notion.so/MASTER-THESIS-7eb6def41702433eb7fb1a8bf1babbcf

We defines three types of leg models in total as Figure 3. And we will introduce the Type 2 model in this chapter. The model and abstract of the front leg are shown as the Figure 4 and 5. The point D is the edge of machien mouse's foot and the points O,E are the centers of two motors which can control the motion of robot. OA,AC,CD are the constituent parts of front leg and the red line is a rope. If we

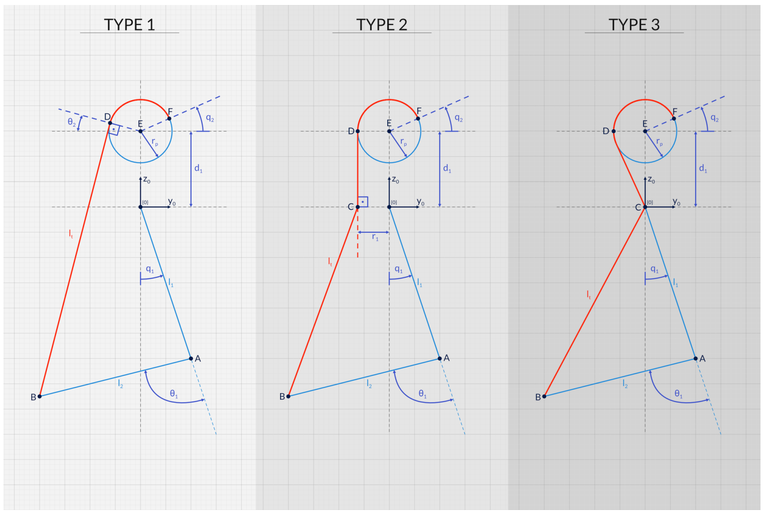


Figure 3: Three types of front leg.

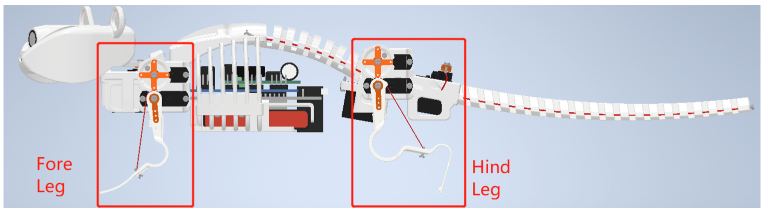


Figure 4: Left View of the mouse: For each leg, there are two motors. One is to control the length of the line and change the angle of the knee, and the other controls the angle of the hip directly. And the structures of the foreleg and hind leg are something different, but their controllers are similar.

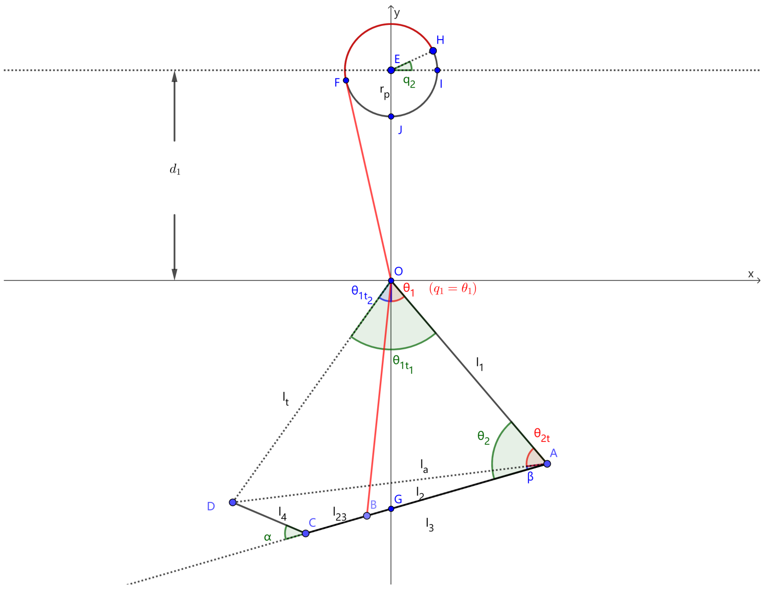


Figure 5: The abstract of front leg.

want to make D arrive the point(x,y), we must compute the rotation angle (q_1,q_2) of the two motors. The given condition of Figure 5 includes l_1 l_4,d_1,α,(x,y) *and* r_p which is the radius of circle E. And the initial angle of q1 and a2 are 0. The initial length of |BO|=l_{r_0} which is also a constant value. Besides, we assume that the sign of θ_{1t_2} is same with the sign of x. The symbols in Figure 5 are described in Equation (5)

$$\begin{cases} \angle HEI = q_2 \\ \angle AOG = q_1 = \theta_1 = \theta_{1t_2} + \theta_{1t_1} \\ \angle DOG = \theta_{1t_2} = sign(x)arccos(\frac{y}{l_t}) \\ \angle DOA = \theta_{1t_1} \\ \angle DQO = \theta_{2t} \\ \angle DAC = \beta \\ \angle OAC = \theta_2 = \theta_{2t} + \beta \\ \angle DCJ = \alpha \\ |AO| = l_1 \\ |AB| = l_2 \\ |AC| = l_3 \\ |BC| = l_{23} = l_3 - l_2 \\ |CD| = l_4 \\ |AD| = l_a \\ |DO| = l_t = \sqrt{x^2 + y^2} \end{cases} \tag{5}$$

Firstly, we can compute l_a in ΔACD by cosine theorem:

$$\begin{cases} l_a = \sqrt{l_3^2 + l_4^2 - 2l_3l_4cos(\pi - \alpha)} \\ \beta = arccos(\frac{l_a^2+l_2^2-l_4^2}{2l_2l_a}) \\ \theta_{2t} = arccos(\frac{l_1^2+l_a^2-l_4^2}{2l_al_1}) \\ \theta_{1t_1} = arccos(\frac{l_1^2+l_t^2-l_a^2}{2l_1l_t}) \end{cases} \tag{6}$$

By now, we have known the value of $q1$
Then we want to do some computation based on point B to prepare for the computation of q_2

$$\begin{cases} \angle BGO = \theta_1 + \theta_2 \\ x_B = l_1sin(\theta_1) - l_2sin(\pi - \angle BGO) \\ y_B = -l_1cos(\theta_1) - l_2cos(\pi - \angle BGO) \\ |OB| = \sqrt{x_B^2 + y_B^2} \end{cases} \tag{7}$$

OF is the tangent line of circle E. So $|OF| = \sqrt{d_1^2 - r_p^2}$ is a constant value. And we have known that the length of the red line is constant. So $|OB| + \overset{\frown}{FH}$ is also constant. Then we can compute q_2 as follows:

$$\begin{cases} \overset{\frown}{HI} = ||OB| - l_{r_0}| \\ q_1 = \frac{\overset{\frown}{HI}}{r_p} \end{cases} \tag{8}$$

We have computed the value of (q_1,q_2)

4 Method

The point of our method is considering how to apply PPO to our machine mouse. The structure of the Artist and Critic Network is shown as Figure 6, which is a 3 layers DFN. In other word, what we should solving now is to define the action space, the presentation of state, and the reward function.

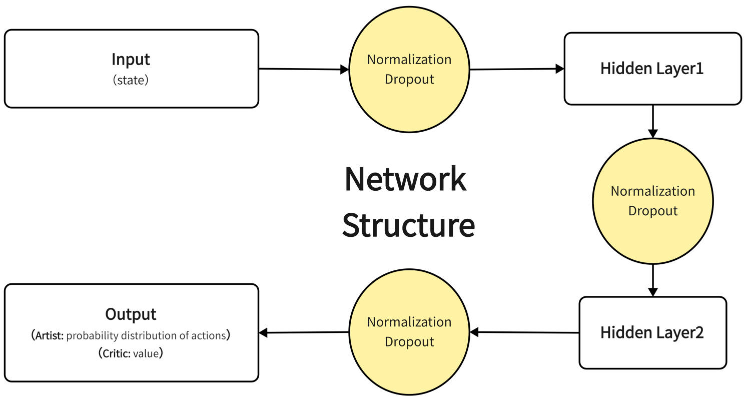


Figure 6: The structure of network.

4.1 Observation and Action Space

Firstly, we want to choose some factors from sensor data as the observation. The dimension of all sensor data is too high. Therefore, if we use all data as observation, it may lead to converge slowly. Besides, a compact observation space helps to transfer the policy to the real robot. [9] In our experiments, the observations include the velocity and azimuth of the whole frame and the position of some joint points.

Secondly, we need to design the action space. For legs, we use the position of the edge of foot to control the movement. Then we can compute the angle of motors as we mentioned in chapter 3. We choose to use the position instead of using the angle of motors due to the fact that it's safer and easier for robot learning. [14] As to the control of head, neck and spine, we use the angle of motors directly.

4.2 Reward Function

We design a reward function based on [9] and we also make some changes:

$$r_n = (y_n - y_{n-1}) - f(s) - g(y_n - y_{n-m}) - w\Delta t|\tau_n \cdot \dot{q_n}| \tag{9}$$

where axis y is the desired running direction and w is a hyperpatameter and Δt is the time step and τ_n are the motor torques and $\dot{q_n}$ are the motor velocities.
The first term encourage the mouse go forward along the axis y. The las term measures the energy expenditure which will lead the robot choose a strategy to

save energy to some extent. Besides, f and g are two punishments functions. The function f uses the sensor data of the height of mouse's centroid and azimuth to judge if the mouse fell. If the mouse fell, f is a positive value and the current epoch will be finished. What's more if the mouse didn't go forward in the past m steps, we should add a punishment g. That is to say, if x is non-positive, g(x) will be a positive value. On the other hand, if the mouse goes forward normally, the value of f and g are both 0.

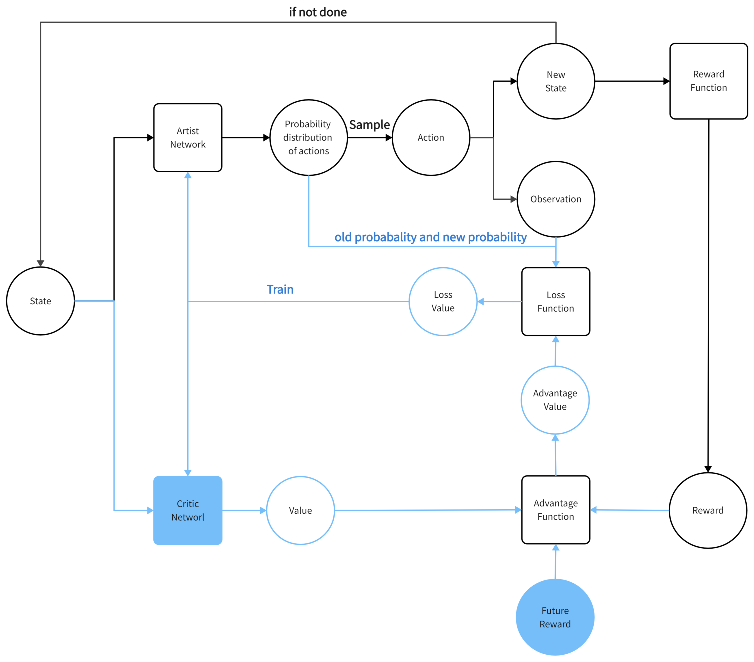
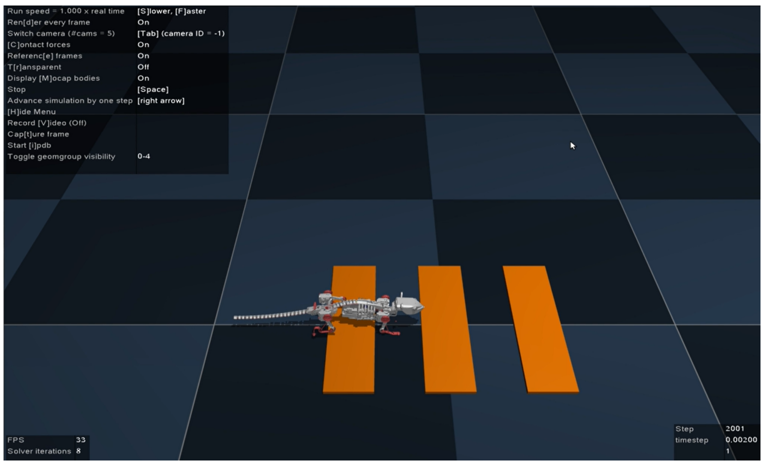


Figure 7: The structure of PPO. The blue lines means they happen when updating the network.

5 Experiments and Results



```
Run speed = 1.000 x real time     [S]lower, [F]aster
Ren[d]er every frame              On
Switch camera (#cams = 5)         [Tab] (camera ID = -1)
[C]ontact forces                  On
Referenc[e] frames                On
T[r]ansparent                     Off
Display [M]ocap bodies            On
Stop                              [Space]
Advance simulation by one step    [right arrow]
[H]ide Menu
Record [V]ideo (Off)
Cap[t]ure frame
Start [i]pdb
Toggle geomgroup visibility       0-4

FPS                33
Solver iterations  8

Step       2001
timestep   0.00200
           1
```

Figure 8: The simulation environment.

We experiment in the mujoco simulation environment. First, we train the mouse on the flat ground to let it learn the basic locomotion. Then we save the parameters of two networks and continue to add some different obstacles to the environment. We want to make the mouse learn how to cross obstacles through this way.

Before each update of the neural network, we need to collect the simulated experience by running 25 roll-outs where each roll have 1000 steps at most. The simulation steps of learning in the flat ground or environment with obstacles are both 7 million. The hypermeters are shown as follows and the learning result can be found at Figure 9 and 10.

Observation Dimension	Policy Network Size	Value Network Size
19	(19,64,32,12)	(19,64,32,1)

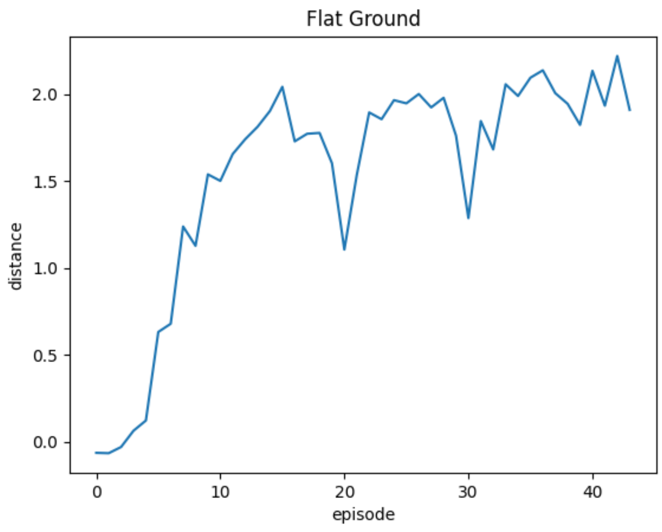


Figure 9: The learning result on flat ground.

6 Conclusion

In this article, we firstly defined a bionic machine mouse which is more likely to the mouse in nature than other quadruped robots. We want to use it to learn more about the motion mechanism of some similar vertebrates. Then we applied the PPO algorithm to the mouse for locomotion learning. And it has shown excellent performance. Our mouse has successfully learned to walk and cross obstacles.

In the next step, we plan to run our model in real environment. There may be lots of problems waiting to be solved, due to the fact that there are many other factors in real world like fractions, delay, and inertia, just to name a few. We should adjust our model to narrow the gap between reality and physics simulator.

Besides, we will also try some other reinforcement learning approaches in the mouse to find the most efficient way. We may also try some other reward functions and state space and choose the best-performing definition.

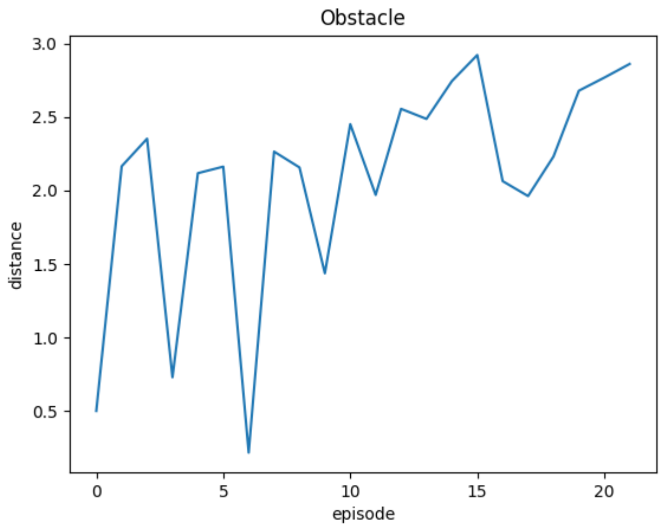


Figure 10: The learning result on the environment with obstacles. (We put some obstacles at the pos 0.3. So the mouse has successfully passed the obstacles.)

7 Reference

[1] Pratik Vyavahare, Sivaranjani Jayaprakash, and Krishna Bharatia. Construction of urdf model based on open source robot dog using gazebo and ros. In *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–5, 2019.

[2] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[4] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[5] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[6] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning. *arXiv preprint arXiv:2206.14176*, 2022.

[7] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel van de Panne. Iterative reinforcement learning based design of dynamic locomotion skills for cassie. *arXiv preprint arXiv:1903.09537*, 2019.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[9] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[10] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[11] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[12] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[13] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[14] Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13, 2017.