Bipedal walking using deep reinforcement learning and proximal policy optimization

Jhon Charaja, Universidade de São Paulo, jhon.charaja@usp.br
Luca Borgonovi, Universidade de São Paulo, lucaborgonovi@usp.br
Adriano Siqueira, Universidade de São Paulo, siqueira@sc.usp.br

Abstrack. *Mathematical models have been developed to describe the dynamic of bipedal walking activity. Some of these models are formulated with a simple physical interpretation of the system. Others consider many nonlinear relationships and kinematic constraints to guarantee high precision and reliability. However, in some cases, using the more detailed models can be a challenging activity due to the number of parameters to be tuned. Likewise, it also does not guarantee the stability of a dynamic system. For this reason, this work focuses on performing bipedal walking using a machine learning approach that does not require a complex mathematical model or control formulation. On the one hand, the MuJoCo dynamic simulator will be used to simulate the dynamics of a two-legged robot. On the other hand, deep reinforced learning with the proximal policy optimization algorithm will be used for the robot to learn to walk.*

Keywords: *two-legged robot, reinforcement learning, proximal policy optimization*

1. INTRODUCTION

Mobile robots have gained more notoriety over the years, as the development of new technologies allows the implementation of these robotic systems in different areas: from agriculture to the exploration of satellites and planets (Rong et al. 2012; Arm et al. 2019). They are classified into wheeled, tracked, undulating, aerial and legged (Siciliano et al. 2009). In this way, they can use abilities already known in fixed robots (e.g. robotic manipulators), such as applying forces greater than those that a human being can perform with high precision, but with a mobile base. For this reason, they are able to perform repetitive, dangerous or demanding work that requires strength and precision, precision beyond what a human being is capable of, in distant places, without an individual needing to take the robot to the task site.

In this context, among the different types of mobile robots, perhaps the best known are wheeled robots due to the efforts to develop fully autonomous cars, and also because of rovers that currently explore Mars (Maurette 2003). However, using these robots in very rough terrain can bring some disadvantages, such as tipping due to some instability that the robot may have due to the unevenness of the ground. Factors that can interfere with the proper functioning of wheeled robots are, for example, holes, steps and steep slopes. In these cases, opting for legged robots is a good option, as they can get around these problems: legs can dodge holes, walk on steps and march up steep climbs.

The design of controllers that can keep these robots stable and functioning properly, becomes more difficult, since it is necessary to control numerous actuators, at the same time, in each of the legs, so that the robot can march correctly (Bledt et al. 2018). Thus, there are different approaches to developing legged robot controllers (Kalakrishnan et al. 2010; Bellicoso et al. 2018). Some of them involve the use of previous robot information, such as dynamic models and computer simulations. However, it is not always so easy to obtain this data, which can complicate the design of the controller. Other approaches involve the use of machine learning, eliminating the need to model the dynamic behavior of the robot, as it will learn from its own mistakes (Kormushev 2013).

Currently, there are several methods to develop drivers for robots with legs. The classic ones use previous knowledge about the robot, such as its dynamic model or information obtained from simulations involving them. For example, the ANYmal robot from ETH uses the inverted pendulum model to generate joint torques, contact forces and motion (Hutter 2016); the MIT Cheetah uses a model predictive control to plan desired contact forces (Bledt 2018).

However, in certain cases, it can be difficult to obtain prior knowledge of the robot through dynamic models and simulations. In this context, a feasible approach to controller design is to use deep learning, more specifically, deep reinforcement learning. Furthermore, another advantage that the use of machine learning provides is that it does not accumulate data generated by the robot, it only uses what is currently being processed, in addition to the fact that the robot can learn to march on its own (Haarnoja 2018).

Controller development methods using deep reinforcement learning may vary depending on the chosen algorithm. The University of California, for example, used a learning algorithm based on maximum entropy reinforcement learning, training a Minitaur robot directly, without having to train a simulation beforehand (Haarnoja 2018).

Deep reinforcement learning (DRL) is a machine training method to teach an agent to take a good sequence of decisions (actions) (Sutton and Barto 2018). DRL works with the simple idea of giving positive (reward) or negative (punishment) reinforcement to the agent's behavior. In this way, the agent learns to make decisions that guarantee the highest amount of reward. Due to its simple operating mechanism, DRL can be adapted to solve complex optimization problems in the robotics area (Kober et al. 2013). For this reason, recent works use DRL with proximal policy optimization (PPO) algorithm to control the waking of Cassie, ABL-BI and Robotis-op3 robots (Xie et al. 2018; Beranek et al. 2021; Jiang et al. 2020). In these three works, the agent learned strategies to maintain the body's balance despite external disturbances of force and uneven terrain.

The proximal policy optimization algorithm formulate a objective function related with the reward and then use the gradient of the objective function to seek the optimal policy (Schulman et al. 2017). PPO is one of the best algorithms for reinforcement learning applications due to its simplicity and high efficiency (Schulman et al. 2017). This algorithm has been successfully implemented to solve problems in bipedal walking (Melo and Máximo 2019), games (Kristensen and Burelli 2020) and unmanned aerial vehicles (Bohn et al. 2019).

This work perform the walking of a bipedal robot using deep reinforcement learning with proximal policy optimization. For this purpose, the MuJoCo simulator is used to compute the dynamics of the bipedal robot. Likewise, OpenAI gym toolkit is used to compute the reinforcement learning equations.

The document is structured as follows. First, section II describes reinforcement learning framework. Second, section III describes deep reinforcement learning. Third, section IV describes the policy proximal optimization algorithm. Fourth, section V describes the experimental setup. Fifth, section VI describes results. Finally, section VII describes conclusion.

2. Reinforcement learning

The reinforcement learning method trains an agent to take the optimal sequence of decisions (Sutton and Barto 2018). The learning process uses positive and negative reinforcement to increase or decrease the probability of choosing an specific action. In this sense, the agent learns, through an iterative process, to make the sequence of decisions that maximizes the amount of reward that he will receive (Sutton and Barto 2018).

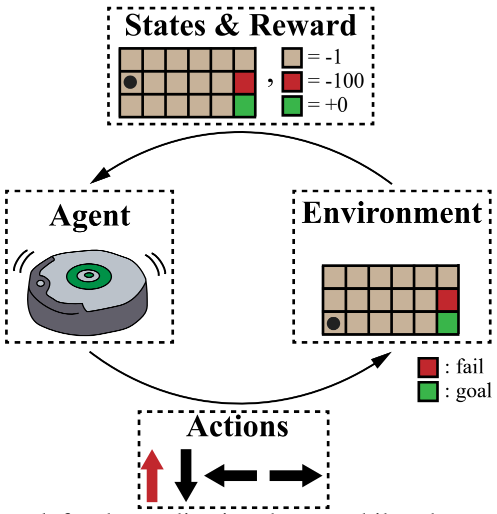

Figure 1: Reinforced learning framework for the application that a mobile robot must reach the desired position. In this example, the agent's actions are up, down, right and left; likewise, the agent's objective is reach the green cell and the agent fails when reach the red cell.

The framework of reinforcement learning is formed by four elements: (i) agent, (ii) actions, (iii) environment, and (iv) states and rewards (Sutton and Barto 2018). Figure 1 describes the reinforcement learning framework for the application that a mobile robot must reach the desired position. First, an agent is the entity who takes decisions based on the reward and punishment that he will receive. Second, action space are all the available actions that the agent could use to interact with the environment. Third, the environment is the space where the agent lives and interact. Fourth, the new agent's state after applying an action in the environment and the reward associated with that specific action. This process will be repeated several times until the agent learns a successful strategy to interact with the environment and maximize the reward.

In reinforcement learning, the agent's strategy is called policy ($\pi(s|a)$) and indicates which actions the agent should chooses in each state (Sutton and Barto 2018). Policies are usually stochastic to consider the uncertainties and probabilities of real world (Tedrake et al. 2004). In this way, policies assign a probability of success to each action, and the agent

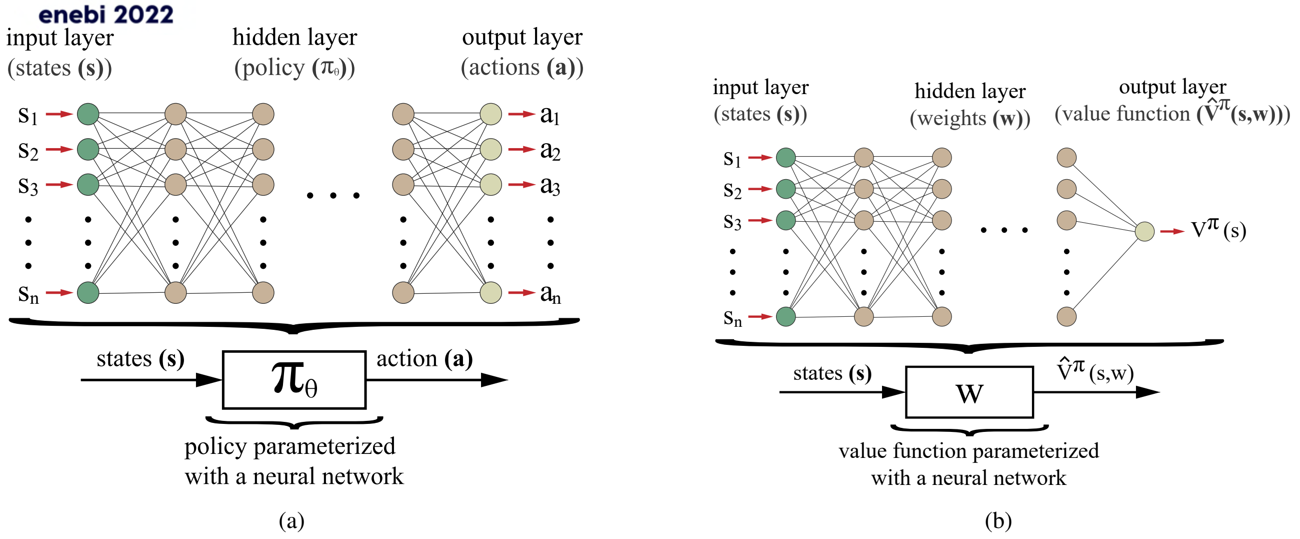

Figure 2: Deep neural networks scheme: (a) predict optimal action and (b) estimate value function.

chooses the action with the highest probability. Hence, the objective of reinforcement learning algorithms is to find the optimal policy that maximizes the amount of reward.

The sum of the accumulated rewards from an initial state s_t is the standard way of quantifying how good or bad the policy is (Sutton and Barto 2018). This quality parameter is called value function ($V^{\pi}(s_t)$) and indicates the discounted sum of rewards that an agent will obtain starting in state s_t and following the policy π (Sutton and Barto 2018). The value function can be computed as

$$V^{\pi}(s_t = s) = E_{\pi}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s\right],$$

where r_t is the reward at time t, γ is a discounted term and s_t is the initial agent's state. However, calculating the value function requires the agent to start in all available states of the environment and in complex activities this can require a large amount of storage and computational cost. For this reason, more recent work uses deep neural networks to estimate the value function and optimal policy (Henderson et al. 2018).

3. Deep reinforcement learning

Deep reinforcement learning (DRL) represent the combination of deep neural networks with reinforcement learning framework (Li 2017). Deep neural networks estimate the value function and predict the policy actions considering the agent's state as input. Figure 2 describes the policy and value function parameterized with a neural network. Therefore, the goal of DRL is to find the neural network parameters that describe the optimal policy that generates the highest reward.

The policy gradient method optimizes the parameters of a neural network that define the behavior of the policy, and in this way, maximize the accumulated reward (Sutton and Barto 2018). The policy gradient algorithms formulate an objective function related to the accumulated reward and then use the gradient ascent method to modify the parameters of the neural network (Thomas 2017). The general form of the objective functions of the policy gradient algorithms is given by

$$L(\theta) = \hat{\mathbb{E}}_t\left[\log\pi_{\theta}(a|s)\hat{A}_t\right],$$

with,

$$\hat{A}_t = V_t^{\pi_\theta} - \hat{V}_t,$$

where $L(\theta)$ is the objective function, $\pi_\theta(a|s)$ is policy parameterized with neural network parameters (θ), $\hat{A}_t$ is advantage function, $V_t^{\pi_\theta}$ is accumulated reward following policy (π_θ) and $\hat{V}_t$ is expected accumulated reward by the neural network.

4. Proximal policy optimization

PPO is a recent policy gradient method that strikes a good balance between rapid implementation and efficiency (Schulman et al. 2017). The PPO algorithm seeks to maximize accumulated reward, minimize estimation error of value

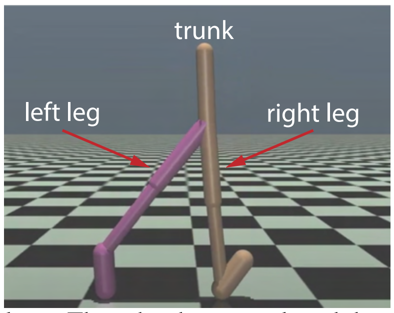

Figure 3: Bipedal robot in MuJoCo simulator. The robot has a trunk and three joints in each leg (ankle, knee and hip).

function and explore different sequence of actions. Likewise, the PPO algorithm adds upper and lower limits to avoid large modifications of the neural network parameters (Schulman et al. 2017). In this way, PPO increases stability and reduces oscillations during neural network training.

The main objective function of PPO algorithm is given by

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right],$$

with,

$$r_t(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta,\text{old}}(a|s)},$$

where $r_t(\theta)$ is the probability ratio between new and old policies, $\text{clip}(\cdot)$ is the clip function to limit upper and lower increments, and ϵ is an hyperparameter that usually is 0.2 (Schulman et al. 2017).

The complete objective function of PPO algorithm can be computed as

$$L^{\text{PPO}}(\theta) = \hat{\mathbb{E}}_t\left[L^{\text{CLIP}}(\theta) - c_1 L^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t)\right],$$

with,

$$L^{\text{VF}}(\theta) = \left(V_\theta(s_t) - V_t^{\text{target}}\right)^2,$$

where $L^{\text{CLIP}}(\theta)$ is the main objective function, $c_{1,2}$ are hyperparameters, $L^{\text{VF}}(\theta)$ is square-error loss and S is an entropy term to increase exploration (Schulman et al. 2017).

5. Simulation setup

5.1 Bipedal robot simulation

The legged robot has two legs (right and left). Likewise, each leg has three joints (ankle, knee, hip) and one joint for trunk orientation. Therefore, the robot has 14 states considering position and velocity. Finally, the dynamic behavior of the bipedal robot is computed with the Multi-Joint dynamics with Contact (MuJoCo) physics simulator (Todorov et al. 2012). Figure 3 describe the bipedal robot in a simulation environment.

5.2 Reinforcement learning: reward configuration

The algorithm must learn to balance the robot as it moves forward. In this sense, the algorithm must be rewarded positively based on how long it can keep its balance while walking. As a result, giving positive rewards until the robot loses its balance is a simple way to set up the reward system. On the one hand, we establish that robot loses its balance when trunk's angular position exceeds $50°$. On the other hand, robot receives a positive reward of $+1$ for each iteration that maintains balance and reward equal to magnitude of linear speed of the whole body to encourage forward movement.

6. Results

The bipedal robot has an ankle, knee and hip for each leg (right and left). The dynamic equations of the bipedal robot was computed with MuJoCo software. The reinforcement learning framework, neural networks and proximal policy optimization algorithm was implemented in Python language using the TensorFlow library, which is an open-source

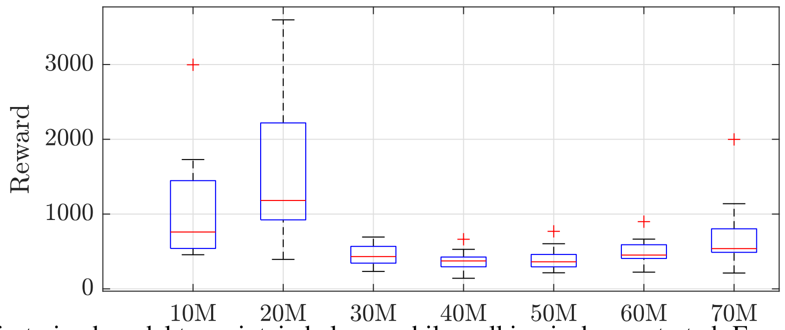

Figure 4: The ability of a trained model to maintain balance while walking is demonstrated. For each model, the box plot considers ten attempts; red line indicates mean value.

library for artificial intelligence and machine learning. The training of the reinforcement learning agent was carried out on a computed with AMD®Ryzen 9 94900HS 3.3 GHz processor, 16 GB of RAM, Ubuntu 20.04.3 LTS 64 bits. The Python version 3.8 and the MuJoCo 2.1 were the platforms where the simulations took place.

Figure 4 shows the variation of reward with respect to number of training iterations from 10 to 70 millions. In this figure, red line indicates mean value, horizontal lines maximum and minimum values[1]. In this figure, at the beginning, the reward increases with the number of iterations. However, after 20 million iterations the reward decreases dramatically and after 40 million iterations the reward begins to increase. This behavior is related to the way the robot learns to control its body to maximize the reward. On the one hand, up to 20 million iterations the robot only uses one leg (right) to balance its body during the walk. On the other hand, between 20 and 40 million the robot tries to use both legs without knowing how to control the other leg (left). This causes the robot to lose its balance quickly and reduces the amount of reward it will receive. Finally, from 40 million onwards, the robot knows how to control both legs and now only focuses on coordinating the movements of each joint to achieve the walk.

The Policy Gradient algorithm was implemented in Python language using the TensorFlow library, which is an open-source library for artificial intelligence and machine learning.

To carry out the simulations, the MuJoCo software was used (Multi-Joint Dynamics with Contact), a simulator for multi-body dynamics with contact. The computational model used to implement the algorithm was a two-dimensional bipedal robot Walker2d-v2, which has 7 joints and two translational movements (horizontal and vertical). During the simulation, the model went through 60 million iterations to assess how gait evolved as the Policy Gradient algorithm was implemented.

All simulations were carried out on a computer with AMD®Ryzen 9 94900HS 3.3 GHz processor, 16 GB of RAM, Ubuntu 20.04.3 LTS 64 bits. The Python version 3.8 and the MuJoCo 2.1 were the platforms where the simulations took place.

7. Conclusions

In this work, we presented the evolution of the walking of a bipedal robot using deep reinforcement learning with proximal policy optimization. The approach was used to avoid using complex models to control the robot.

The algorithm learned to coordinate the movements of the robot's legs after 70 million iterations. Analyzing the results of the simulations, it can be observed that the robot acquired a very different gait from that of a human being or any biped. The strange way of walking of the robot is related to the reward system that was established in this work. The reward system just motivates the algorithm to move forward and maintain balance, not to establish a human gait. For example, the algorithm only uses one leg at the start of training. However, as the algorithm evolves, it begins to use the other leg, although walking is still unconventional for biological systems. Finally, it was observed that during training, the robot made movements that removed its center of mass from the support polygon. This causes the robot to lose its balance in the following steps.

As work in the future, it will be considered to keep the center mass within the support polygon to increase the stability of the system, as well as the possibility of adding muscles to achieve natural movements.

8. REFERENCES

Arm, P.; Zenkl, R.; Barton, P.; Beglinger, L.; Dietsche, A.; Ferrazzini, L.; Hampp, E.; Hinder, J.; Huber, C.; Schaufelberger, D. Spacebok: A dynamic legged robot for space exploration. In 2019 international conference on robotics

[1]visual material of the training of the two-legged robot in `https://www.youtube.com/watch?v=gldBp-X0U9k&ab_channel=LucaBorgonovi`

and automation (ICRA). IEEE, 2019. p. 6288-6294.

Bellicoso, C. D.; Jenelten, F.; Gehring, C. and Hutter, M. Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2261–2268, 2018.

Beranek, R.; Karimi, M. and Ahmadi, M. A behavior-based reinforcement learning approach to control walking bipedal robots under unknown disturbances. IEEE/ASME Transactions on Mechatronics, 2021.

Bledt, G.; Powell, M. J.; Katz, J.; Di Carlo, J.; Wensing, P. M. and Kim, S. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 2245–2252.

Bøhn, E.; Coates, E. M.; Moe, S. and Johansen, T. A. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In 2019 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2019, pp. 523–533.

Haarnoja, T.; Ha, S.; Zhou, A.; Tan, J.; Tucker, G. and Levine, S. Learning to walk via deep reinforcement learning. arXiv preprint arXiv:1812.11103, 2018.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D. and Meger, D. Deep reinforcement learning that matters. In Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1, 2018.

Hutter, M.; Gehring, C.; Jud, D.; Lauber, A.; Bellicoso, C. D.; Tsounis, V.; Hwangbo, J.; Bodie, K.; Fankhauser, P.; Bloesch, M.; Diethelm, R.; Bachmann, S.; Melzer, A. and Hoepflinger, M. Anymal - a highly mobile and dynamic quadrupedal robot. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 38–44.

Jiang, Y.; Zhang, W.; Farrukh, F. U. D.; Xie, X. and Zhang, C. Motion sequence learning for robot walking based on pose optimization. In 2020 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2020, pp. 1877–1882.

Kalakrishnan, M.; Buchli, J.; Pastor, P.; Mistry, M. and Schaal, S. Fast, robust quadruped locomotion over challenging terrain. In 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010, pp. 2665–2670.

Kober, J.; Bagnell, J. A. and Peters, J. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238–1274, 2013.

Kormushev, P.; Calinon, S. and Caldwell, D. G. Reinforcement learning in robotics: Applications and real-world challenges. Robotics, vol. 2, no. 3, pp. 122–148, 2013.

Kristensen, J. T. and Burelli, P. Strategies for using proximal policy optimization in mobile puzzle games. In International conference on the foundations of digital games, 2020, pp. 1–10.

Li, Y. Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274, 2017.

Maurette, M. Mars rover autonomous navigation. Autonomous Robots, vol. 14, no. 2, pp. 199–208, 2003.

Melo, L. C. and Máximo, M. R. O. A. Learning humanoid robot running skills through proximal policy optimization. In 2019 Latin american robotics symposium (LARS), 2019 Brazilian symposium on robotics (SBR) and 2019 workshop on robotics in education (WRE). IEEE, 2019, pp. 37–42.

Rong, Y.; Jin, Z. and Cui, B. Configuration analysis and structure parameter design of six-leg agricultural robot with parallel-leg mechanisms. Transactions of the Chinese Society of Agricultural Engineering, vol. 28, no. 15, pp. 9–14, 2012.

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

Siciliano, B.; Sciavicco, L.; Villani, L. and Oriolo, G. Robotics modeling, planning and control springer. Verlag, London, 2009.

Sutton, R. S. and Barto, A. G. Reinforcement learning: An introduction. MIT press, 2018.

Tedrake, R.; Zhang, T. W. and Seung, H. S. Stochastic policy gradient reinforcement learning on a simple 3d biped. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3. IEEE, 2004, pp. 2849–2854.

Thomas, P. S. and Brunskill, E. Policy gradient methods for reinforcement learning with function approximation and action-dependent baselines," arXiv preprint arXiv:1706.06643, 2017.

Todorov, E.; Erez, T. and Tassa, Y. Mujoco: A physics engine for model-based control," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012, pp. 5026–5033.

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