A Deep Reinforcement Learning Algorithm for Objects Balance Control with Hexapod Robot

1st Yu-Hao Tsai
Department of Computer Science
Department of Mechatronic Engineering
National Taiwan Normal University
Taipei, Taiwan
lyhtsai@gmail.com

2nd Saeed Saeedvand
Department of Electrical Engineering
National Taiwan Normal University
Taipei, Taiwan
saeedvand@ntnu.edu.tw

3rd Jacky Baltes
Department of Electrical Engineering
National Taiwan Normal University
Taipei, Taiwan
jacky.baltes@ntnu.edu.tw

***Abstract*—Legged robots have been a prominent focus of research for an extensive period, owing to their enhanced stability and maneuverability in challenging terrains compared to wheeled robots. In recent decades, the application of reinforcement learning (RL) to train legged robots has yielded excellent results. This approach has effectively addressed numerous challenges that traditional methods struggled to overcome, such as navigating through complex environments. The advancements in RL for legged robots have significantly enhanced the feasibility and success of demanding applications, including exploration and resource delivery in outdoor settings. This paper introduces a training architecture for a hexapod robot based on Proximal Policy Optimization (PPO) implementation on a GPU. This architecture enables the hexapod to transport objects across uneven terrain surfaces while adhering to input control commands. Our investigation extends to assessing the performance limitations of the hexapod robot in both flat and uneven ground environments, with a particular focus on evaluating the impact of different activation functions.**

***Index Terms*—hexapod, reinforcement learning, Isaac Gym, objects carrying**

I. Introduction

The study of legged robot control has been a long-standing and intriguing area of research, mainly because of their superior performance in handling complex tasks in challenging environments, unlike wheeled robots. Legged robots bring various advantages, such as diverse walking patterns, robust load-bearing capabilities, and reliance on individual support structures [1] [2]. While recent research has extensively covered the locomotion of quadrupeds and bipedal robots [3], [4], our focus in this study is specifically on hexapods. We chose hexapods due to their inherent stability and lightweight design, distinguishing them from similar-sized four-legged robots and humanoids. Hexapods show promise for practical applications in exploration, resource delivery, and even disaster rescue [5].

Numerous studies have addressed the locomotion of hexapod robots. A commonly employed technique in many methods involves Central Pattern Generators (CPGs) [6]–[8]. CPGs utilize centrally generated rhythms to dictate overall behavior, leveraging their strength in imposing a robust prior on the agent's action space. This results in a significantly reduced parameter count and a more natural gait. Despite the effectiveness of this approach in addressing various disturbances during locomotion, its limitations become apparent when the robot encounters challenging terrains. One recent research even states that no CPG-based approach has proven to be robust enough to handle a variety of challenging terrains [9]. On the other hand, over the past few years, there has been considerable interest in contrasting approaches, which are data-driven methods such as Deep Reinforcement Learning (DRL). This attention is fueled by their capacity to generate more precise and resilient control policies, opening up new viable approaches for motion control in hexapod robots. The adoption of reinforcement learning (RL) in continuous control tasks has been increasingly prominent [10]. Recent studies have shown a preference for RL methods to empower both real and simulated agents in overcoming challenging tasks [11]–[15].

In contrast to the prior inclination toward traditional control system approaches, contemporary deep learning algorithms have ushered in a shift, enabling agents to learn emergent behaviors and tackle dynamic tasks from the ground up [16]. This transformation is notably apparent in the field of robotics, where an increasing number of researchers are now addressing longstanding challenges—such as locomotion and manipulation—through the lens of deep reinforcement learning [17]. The Proximal Policy Optimization (PPO) algorithm stands out as one of the simplest and most effective Deep Reinforcement Learning algorithms [18]. This algorithm adeptly estimates the policy gradient [19] on a batch of states, executing multiple updates while ensuring that the revised policy stays within a reasonable deviation from the current one. We use the PPO algorithm with a separate actor-critic network for our experiment.

Exploring the traversal of terrain by hexapod robots has been a subject of study for several decades, encompassing a variety of research topics. In recent years, significant advancements have been made using a data-driven approach for hexapod locomotion. For instance, [9] enables hexapods to navigate rough terrain in simulations by employing binary contact sensors at the endpoints of their feet. [20] introduces a terrain-adaptable method that combines the Central Pattern Generator (CPG) approach and RL for hexapod locomotion control. Additionally, in a recent study, [21] presented a

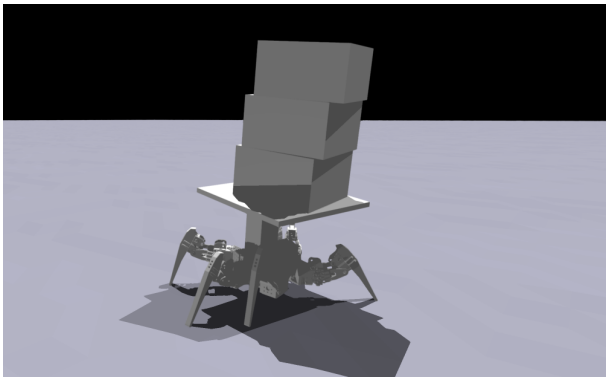

Fig. 1. Our hexapod robot moves and carries objects on uneven ground.

perceptive walking model for legged hexapods, allowing them to traverse terrain with random joist structures using egocentric vision.

While these notable works primarily focus on the locomotion and gait of hexapod robots, research on object carrying and the stability of hexapod robots is relatively scarce. In this study, our objective is to train an RL policy to enable our hexapod robot to carry boxes on uneven ground and follow control commands. The overall concept is shown in Figure 1.

In addition, the robot we used in our experiment has 12 degrees of freedom (DoFs), while most of the hexapod robots used in those other works above, like [5] [9] [21] have 18 DoFs or even more. The simpler structure of our robot leads to the lower cost of the hardware, such as actuators, yet, of course, increases the difficulty of conducting complex missions. Teaching the policy to follow commands is another challenging task since the robot needs to exhibit the opposite behavior upon receiving the corresponding command, which could contrast with the previous instruction. In training the model, the transfer-learning technique [22] [23] plays a crucial role throughout the entire process. The training can be divided into three sequential steps: initially instructing the agent to execute the optimal gait and understand command sensitivity on flat ground, transitioning to uneven terrain, and concluding with the introduction of boxes to finalize the training. All the implementations were conducted in the Nvidia Isaac Gym Environment. With the advantage of parallel computing, the sum of the training time can be reduced within an hour.

II. Methodology

A. Reinforcement Learning and The Training Approach

Our training process can be segmented into three steps, as depicted in Figure 2. Throughout the training, we incorporated the transfer-learning technique. In the initial stage, our focus was on enhancing our agent's proficiency in walking forward and backward, turning left and right, and responding to the control commands: [forward, backward, left, right]. During this phase, we provided distinct commands one at a time and then introduced random commands for each environment. The typical sequence of commands during training was as follows:

1. Forward
2. Backward
3. Forward and backward (randomly)
4. Turn left
5. Turn right
6. Left or right (randomly)
7. Random commands

It's worth noting that, at times, we revisited specific steps to provide additional training if the agent did not perform certain behaviors well.

Following this, we progressed to the next stage. In this phase, we initiated training the hexapod robot on uneven ground to enhance our agent's terrain adaptability. Initially, we began training with a smaller slope and the maximum height of random hills in the terrain. Subsequently, we systematically increased both the slope and the maximum height. This technique aligns with the approach outlined in [21].

Finally, we introduced boxes into the agent's training. We begin with just one box, gradually increasing the number over subsequent sessions. Ultimately, on flat ground, our robot demonstrated the capability to carry five boxes or even more without issue. However, considering our focus on uneven terrain, we encountered limitations. On uneven ground, our current constraint is three boxes, and there remains a possibility of them falling to the ground.

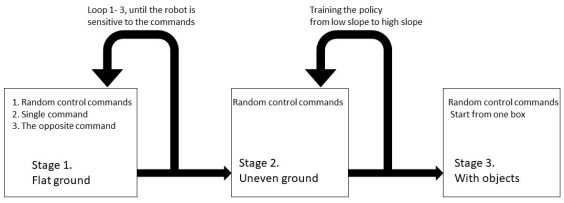

Fig. 2. The overall training process

In the context of reinforcement learning, the environment constitutes a Markov Decision Process with a state space S, action space A, transition operator T, and reward function r where:

$$\forall s_t + 1, s_t \in S, a_t \in A, \tag{1}$$

$$T_{i,j,k} = p(s_t + 1 = i | s_t = j, a_t = k)$$

$$r : S \times A \to \mathbb{R} \tag{2}$$

The objective of reinforcement learning is to determine a policy $\pi_\theta(a_t|s_t)$ that maximizes the cumulative reward:

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \right] \tag{3}$$

The total reward collected in one trajectory is denoted as $\sum_{t=0}^{T} \gamma^t r_t$, where r_t is the reward collected at time t, T is the length of each episode, and $\gamma \in [0, 1]$ is the discount factor determining the weight of future rewards. We utilize Proximal Policy Optimization (PPO) [18] to train our policies, as it efficiently optimizes Equation 3 and has exhibited remarkable success in numerous continuous control tasks. Our network comprises three hidden layers with 512, 256, and 128 neurons, respectively, employing the ELU activation function [24]. Table 1. provides the hyperparameter settings for training with PPO.

TABLE I
HYPER-PARAMETER SETTINGS FOR TRAINING OF PPO.

parameter	***value***
λ for generalized advantage estimation	0.99
PPO clip threshold	0.2
Entropy coefficient	0.0
Learning rate	3×10^{-4}
Kullback–Leibler divergence threshold	0.008
PPO batch size	12288
Discount factor	0.95

The agent receives the most recent observations from the environment at each timestep. In our case, the simulation timestep was set to 0.01. While opting for a smaller timestep would proportionally increase the training time, it allows the physics engine to simulate physics with greater accuracy.

The observations serve as inputs to the policy network. These observations contain information about the current state of the agent, encompassing details such as the current position and velocity of each joint (in our case, they are angle and angular velocity), the linear and angular velocity of the base of our hexapod robot, and the last actions. Additionally, to give our agent a basic sense of the environment, we also passed the torques of each joint and the contact forces of each tibia to the observation tensor. For being capable of carrying objects, we added a force sensor on the plate and passed the sensed force to the observation. Another important observation is our control commands. To make our neural network sensitive enough and correctly understand the control commands, we eventually scaled the forward/backward and the left/right commands by multiplying them with 8 and 1, respectively. The complete observation tensor is shown in Table 2.

To enhance the policy's robustness and generalization properties and inject variability into the training loop, we implemented domain randomization [19]. Throughout training, we systematically sample a varied set of physics parameters for each environment whenever a reset is required. This includes randomizing the initial position and velocity of joints, as well as randomizing commands upon reset. Optimizing the policy on trajectories generated with a diverse range of physics parameters compels the agent to acquire behavior that performs effectively across all variations.

TABLE II
HYPER-PARAMETER SETTINGS FOR TRAINING OF PPO.

Observations	
Types of Information	***Dimentionality***
linear velocity of agent	3
angular velocity of agent	3
agent's orientation	3
commands	2 (forward/backward; left/right)
force from sensor	3
positions of joints	18
velocity of joints	18
torques of joints	18
contact forces	18 (x, y, z for each leg)
actions	18

B. Actions of The Hexapod Robot

In Isaac Gym, there are three ways of controlling degrees of freedom (DoF) — position, velocity, and effort. In our experiment, we adopt the first method to control our agent. It's worth mentioning that we don't directly instruct our policy to output absolute positions (degrees of each joint). Instead, we treat the actions generated by the policy as angular displacements. These displacements are then added to the cumulative final positions, subsequently updating our agent based on these new positions. We observed that employing this approach contributes to a smoother behavior of our agent in the training scenario.

Our agent comprises six legs, each with three joints. Consequently, in Isaac Gym, the total number of actions is 18. However, as previously mentioned in the introduction, each leg includes one mimic joint. Hence, the actual degrees of freedom amount to 12. We computed the relationship between the control joint and the corresponding mimic joint to address this. Rather than relying on the policy to dictate actions, we manually set the actions based on this established relationship.

In the Isaac Gym simulation, the actions automatically align with the constraints defined by the URDF file. Similar to the impact of the number of Degrees of Freedom (DoFs), the more constraints there are, the greater the difficulty in accomplishing complex missions. The limitations of each joint of our agent are listed in Table 3.

TABLE III
JOINTS LIMITATION OF THE AGENT

Limits			
Joints	***Rotation (rad)***	***Velocity (rad/s)***	***Effort (N-m)***
coxa	[0 - 1π]	3.8π	1.4
coxa (middle)	[-1 - 0.8]	3.8π	1.4
femur	[0.4 - 0.9π]	3.8π	1.4
tibia	opposite of femur	-	-

C. Reward Function

In this section, we delve into the specifics of our reward function. The design of the reward function is tailored to incentivize the agent to maintain balance on top of the board.

The complete reward function r_t is the summation of all the sub-rewards and sub-punishments. The hyper-parameters in (4) are shown in Table 4.

$$r_t = \alpha_1 r_l + \alpha_2 r_a + \beta_1 r_{lp} + \beta_2 r_{ap} + \upsilon r_j + \rho_1 r_{ar} + \rho_2 r_{pp} \tag{4}$$

TABLE IV
HYPER-PARAMETER SETTINGS FOR TRAINING OF PPO.

Hyper-paramaters	*Value*
α_1	2.5
α_2	2.0
β_1	0.5
β_2	0.25
υ	0.5
ρ_1	-0.002
ρ_2	-0.02

- In our study, the main components of our reward system are the linear velocity reward r_l and the angular velocity reward r_a. The agent earns these rewards upon achieving the desired linear and angular velocities. The reward magnitude increases as the velocities approach the predefined goals :

$$r_l = tanh(8 \cdot v_l') \tag{5}$$
$$r_a = tanh(2 * v_a') \tag{6}$$

  Here, the hyperbolic tangent (tanh) functions are utilized to guide the agent toward the goal linear velocity of 0.5 m/s and angular velocity of 1.5 rad/s. Through comparison with other common functions like exponential (exp), we observed that the tanh function is more effective in instructing the agent to walk. A crucial manipulation was applied before inputting v_l' and v_a' : we multiplied the original linear and angular velocities by our commands, which take values of -1, 0, or 1. This manipulation allows the direction of the velocities performed by our agent to be reflected effectively through the tanh function.

- In correspondence to the primary rewards, we also incorporate primary punishments. The r_{lp} and r_{ap} components represent the linear and angular velocity punishments, respectively, incurred when the agent deviates from the expected behavior dictated by the control commands. The functional expression of these rewards is as follows:

$$r_{lp} = c_l \cdot abs(v_l) \cdot r_a \tag{7}$$
$$r_{ap} = c_a \cdot tanh(abs(v_a^* + v_a)) \cdot r_l \tag{8}$$

  We introduced check variables c_l and c_a for linear and angular commands, respectively. These variables take a value of 0 when the control commands are not 0; otherwise, they are set to -1. In instances where the linear or angular velocity is non-zero while the command is set to 0 (hold), we impose a proportional penalty on the corresponding reward based on the agent's speed (absolute velocity) at that moment. It's worth noting that v_a* represents the angular velocity in the last action. We incorporated this into the function as we observed that the instantaneous angular velocity may not always closely match the average angular velocity.

- Several additional factors contribute to the final rewards. The perpendicular linear velocity punishment r_{pp} is introduced to guarantee our agent's capability to walk in a straight line. The joint acceleration penalty r_j and the action rate penalty r_{ar} are incorporated to promote smoother movements. This strategy prevents the agent from exploiting the physics engine and achieving high rewards through implausible locomotion in real-life scenarios. The functional expression of these rewards is as follows:

$$r_{pp} = tanh(25 \cdot v_x - 3) + 1.0 \tag{9}$$
$$r_j = \sum (v_j{}^* - v_j)^2 \tag{10}$$
$$r_{ar} = \sum (a^* - a)^2 \tag{11}$$

  v_x represents the perpendicular linear velocity of the agent. $v_j{}^*$ represents the last DoF velocity of one joint in the last timestep. $a*$ represents the last action of one joint.

III. RESULTS

In this section, we provide comparisons of various training settings and present the experimental results of the proposed training method.

A. Simulation Environment Setup

The agent employed in our experiments is a hexapod robot modified from the Scorpi, which was developed by the ROBOTIS company. A hexapod consists of six legs, each of which is typically divided into three sections: coxa, femur, and tibia, as illustrated in Figure 3.

This hexapod has 12 degrees of freedom (DoF). Due to the presence of a mimic joint in this robot—a structure currently unsupported in the Isaac Gym simulation—we excluded the mimic link during simulation. To address the mimic joint issue, our approach involves directly defining the action in the joint on the tibia instead of assigning control to it from our policy.

We have added a plate to the robot's base for the next step of our experiment. The file format of our robot is URDF, encompassing crucial data such as mass, collision shape, inertia, and more. The boxes we use are generated directly in the Isaac Gym environment, providing physical information such as length, width, height, and density. The relative details

Fig. 3. Left: The hexapod robot; right: one of legs of the hexapod

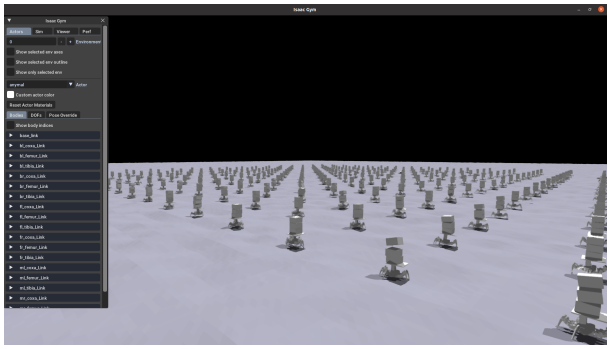

Fig. 4. Training environment in Isaac Gym.

TABLE V
SPECIFICATIONS OF THE HEXAPOD AGENT, THE PLATE, AND BOXES.

info	***hexapod***	***plate***	***box***
weight (g)	675	15	50-100
length (cm)	20-30	25	15
width (cm)	20-40	25	15
height (cm)	2-15 (base to the ground)	1 + 15 (plate and pillar)	10
actuator	Dynamixel 2xl430-w250-t	-	-

of our agent and each box's information are presented in Table 5.

Our simulation environment is constructed using Isaac Gym, a physics simulation environment designed for reinforcement learning (RL) research by NVIDIA. Isaac Gym notably offers robust support for large-scale parallel training. This platform can create a physics simulation scene directly on the GPU. Furthermore, an API enables simultaneous access to read and write the state of multiple environments, with the flexibility to vary the number of environments. In our device, we can easily establish 4096 parallel environments. However, considering the substantial increase in memory usage when deploying terrain in the simulation environments, we opted to train our policy with 256 environments. Each environment comprises an instance of the hexapod robot agent and boxes. Figure 4. provides a screenshot of the training scene with multiple parallel environments in Isaac Gym. The GPU used for our experiments was an RTX 3080 laptop version with 8 GB of VRAM.

The terrain is also generated in Isaac Gym. We utilize the random uniform terrain from the Isaac Gym example they released [25], featuring a maximum height of 0.1 m and random hills with slopes ranging from 0.01 to 0.2 m.

B. Comparison and Analysis

The policy is optimized using samples collected in the Isaac Gym simulation environment. The total training time for all sub-processes is less than 1 hour on a single Nvidia RTX 3080 Laptop GPU. We experimented with various activation functions, including Elu, Selu, Relu, Tanh, and Sigmoid. Figure 5. illustrates the diverse rewards for one substep of training using different activation functions. The bright color curves represent the average reward across all environments, while the translucent curves depict the maximum reward.

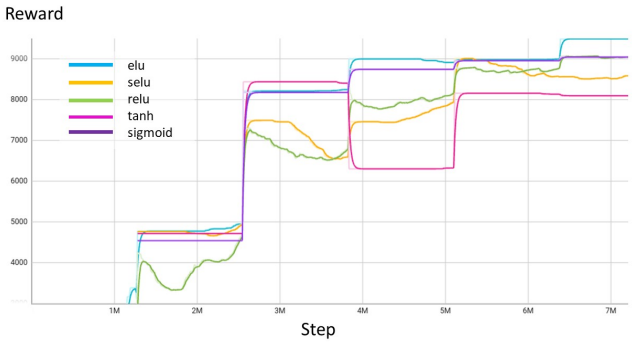

Fig. 5. Rewards for each activation function.

We also assessed the effectiveness of Long-Short Term Memory (LSTM) networks. Contrary to some related research, such as [9], which integrated LSTM layers into their neural networks and observed improved performance, our findings did not show a similar outcome. We attribute this to the fact that our goal involves the opposite behavior of our policy, making the model more challenging to train. This aligns with issues observed in [26], [27] for LSTM networks. The comparison between the regular model and the model with LSTM layers is depicted in Figure 6.

Another notable observation is that we discovered our training process to be more efficient when employing a separated actor-critic network in the PPO algorithm. In recent years, many researchers have leaned towards choosing an actor-critic network with some shared layers in the network structure. This approach allows the actor to benefit from the state representation learned by the critic. However, it also introduces more instability in the learning process because two components of the gradient are being backpropagated simultaneously through the network. [28] In our case, this instability made it more challenging for the agent to achieve higher results and increased difficulty finding the right behavior. Figure 7 illustrates the comparison between the separated and non-separated networks.

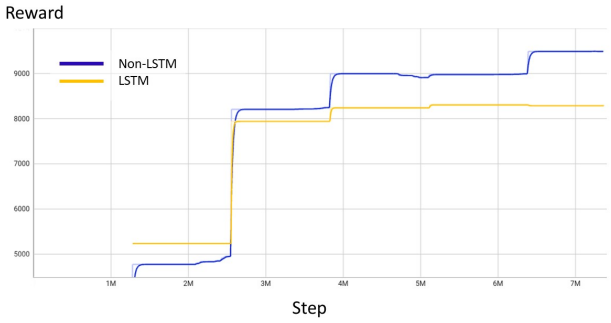

Fig. 6. The regular-LSTM networks comparisons.

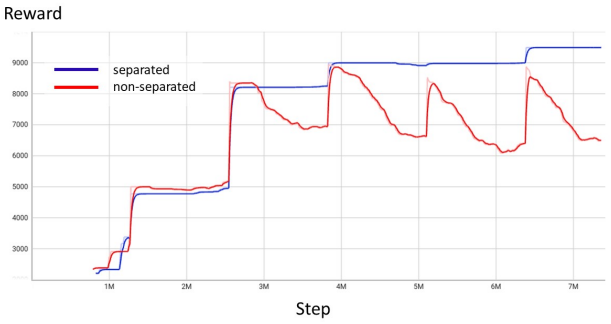

Fig. 7. The separated network versus the non-separated network.

IV. Conclusion

In this paper, we introduce a reinforcement learning (RL) training architecture based on Proximal Policy Optimization (PPO) applied to a hexapod robot, enabling it to follow control commands (Forward, Backward, Left, Right) and concurrently carry objects on uneven ground. We devised a comprehensive reinforcement learning (RL) framework for the hexapod robot and implemented it in the Isaac Gym environment. Additionally, we conducted experiments to evaluate different training settings involving various activation functions and neural network structures in the physics simulation. Remarkably, the total training time to achieve our result can be accomplished within an hour using a single GPU on a laptop.

For future work, our plans include refining the current training algorithm to achieve improved gait and locomotion for our hexapod robot. Additionally, while our policy was trained in flat ground and random uniform terrain environments in this work, we aim to train the agent in a more diverse set of terrains to enhance the robustness of our policy. Lastly, we intend to tackle the sim-to-real problem, with the ultimate goal of deploying the policy learned in simulation onto the real-world robot, ensuring it performs at the same level as it does in simulation.

Acknowledgment

Special thanks to Dr. Jaesik, Mr. Roux, Mr. Afriza, and Mr. Song for their helpful suggestions.

References

[1] Zhihua Chen, Shoukun Wang, Junzheng Wang, Kang Xu, Tao Lei, Hao Zhang, Xiuwen Wang, Daohe Liu, and Jinge Si. Control strategy of stable walking for a hexapod wheel-legged robot. *ISA transactions*, 108:367–380, 2021.

[2] Yong Gao, Wu Wei, Xinmei Wang, Yanjie Li, Dongliang Wang, and Qiuda Yu. Feasibility, planning and control of ground-wall transition for a suctorial hexapod robot. *Applied Intelligence*, pages 1–19, 2021.

[3] Jacky Baltes, Guilherme Christmann, and Saeed Saeedvand. A deep reinforcement learning algorithm to control a two-wheeled scooter with a humanoid robot. *Engineering Applications of Artificial Intelligence*, 126:106941, 2023.

[4] Saeed Saeedvand, Hanjaya Mandala, and Jacky Baltes. Hierarchical deep reinforcement learning to drag heavy objects by adult-sized humanoid robot. *Applied Soft Computing*, 110:107601, 2021.

[5] Qiao Sun, Feng Gao, and Xianbao Chen. Towards dynamic alternating tripod trotting of a pony-sized hexapod robot for disaster rescuing based on multi-modal impedance control. *Robotica*, 36(7):1048–1076, 2018.

[6] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[7] Yudi Isvara, Syawaludin Rachmatullah, Kusprasapta Mutijarsa, Dinara Enggar Prabakti, and Wiharsa Pragitatama. Terrain adaptation gait algorithm in a hexapod walking robot. In *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 1735–1739. IEEE, 2014.

[8] Salil S Bidaye, Till Bockemühl, and Ansgar Büschges. Six-legged walking in insects: how cpgs, peripheral feedback, and descending signals generate coordinated and adaptive motor rhythms. *Journal of neurophysiology*, 119(2):459–475, 2018.

[9] Teymur Azayev and Karel Zimmerman. Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification. *Journal of Intelligent & Robotic Systems*, 99(3-4):659–671, 2020.

[10] 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.

[11] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[12] Zhiying Qiu, Wu Wei, and Xiongding Liu. Adaptive gait generation for hexapod robots based on reinforcement learning and hierarchical framework. In *Actuators*, volume 12, page 75. MDPI, 2023.

[13] Jyun-Ting Song, Guilherme Christmann, Jaesik Jeong, and Jacky Baltes. Reinforcement learning and action space shaping for a humanoid agent in a highly dynamic environment. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2022-Winter*, pages 29–42. Springer, 2023.

[14] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[15] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018.

[16] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

[17] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

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

[19] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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)*, pages 23–30. IEEE, 2017.

[20] Qiyue Yang, Yue Gao, and Shaoyuan Li. Terrain-adaptive central pattern generators with reinforcement learning for hexapod locomotion. *arXiv preprint arXiv:2310.07744*, 2023.

[21] Zixian Zang, Maxime Kawawa-Beaudan, Wenhao Yu, Tingnan Zhang, and Avideh Zakhor. Perceptive hexapod legged locomotion for climbing joist environments. 2023.

[22] David N Perkins, Gavriel Salomon, et al. Transfer of learning. *International encyclopedia of education*, 2:6452–6457, 1992.

[23] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[24] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese control and decision conference (CCDC)*, pages 1836–1841. IEEE, 2018.

[25] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[26] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[27] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[28] Laura Graesser and Wah Loon Keng. *Foundations of deep reinforcement learning*. Addison-Wesley Professional, 2019.