Reinforcement Learning and Action Space Shaping for a Humanoid Agent in a Highly Dynamic Environment

Jyun-Ting Song[1], Guilherme Christmann[2], Jaesik Jeong[1] and Jacky Baltes[1]

***Abstract*— Reinforcement Learning (RL) is a powerful tool and has been increasingly used in continuous control tasks such as locomotion and balancing in robotics. In this paper, we tackle a balancing task in a highly dynamic environment, using a humanoid agent and a balancing board. This task requires continuous actuation in order for the agent to stay in a balanced state. We developed an implementation using a state-of-the-art RL algorithm and simulator that can achieve successful balancing in under 40 minutes of real-time with a single GPU. We sought to examine the impact of action space shaping in sample efficiency and designed 6 distinct control modes. Our constrained parallel control modes outperform a naive baseline in both sample efficiency and variance to the starting seed. The best performing control mode (PLS-R) is around 33% more sample efficient than the second-best, requiring 70 million fewer training timesteps to reach comparable performance.**

Our implementation is open-source and freely available at: https://github.com/NTNU-ERC/Robinion-Balance-Board-PPO.

I. INTRODUCTION

The use of reinforcement learning (RL) in continuous control tasks has been gaining more traction [1]. Recent works have begun favoring RL methods to enable real and simulated agents to solve challenging tasks [2] [3] [4]. Where more traditional control system approaches were previously preferred, now with modern deep learning, agents can learn emergent behaviors and solve dynamic tasks from scratch [5]. Specifically, in the field of robotics, this has pushed many researchers to tackle age-old problems such as locomotion and manipulation under the lens of deep reinforcement learning [6].

In RL, one of the most impactful design decisions is the choice of the action space [7]. The action space refers to the method which the actions inferred by the policy are applied to the agent in the environment. Different action spaces may differ in properties such as the number of actions, i.e., the dimensionality of the space, as well as how the action values are executed by the agent. Moreover, action spaces can be completely discrete [8], completely continuous [9], or a mix of discrete and continuous actions [10]. Furthermore, action spaces can also differ in the way of normalization ranges or other transformations. The idea of transforming the action space aims to enable more efficient learning, and it is known as *action space shaping* [7] in the literature.

The trend of growing RL is also observed in the sub-field of humanoid robotics. For a long time, balancing and locomotion problems were solved by using traditional control methods. For example, the work of [11] consisted of generating trajectories using a linear inverted pendulum model. Another example is the work by [12] which develops biped walking based on zero's moment point (ZMP). Whereas in recent works it's more common to see the use of deep RL to solve the same or similar problems. For example, [13] investigated the use of deep RL for human-like walking behavior, and explored action space shaping with two control methods: torque-based and muscle-based. Other works such as [14], [15], and [16] shows that with careful training and design RL policies for locomotion are robust to a large range of environments and disturbances. It also demonstrates that policies that are trained completely in simulation can be feasibly deployed to the real-world counterpart of the agent.

Although RL for locomotion is well-established in the field, for the specific case of explicit balancing tasks, it seems less so. We wish to highlight the difference of balancing as part of locomotion, and balancing in a highly dynamic environment, such as on top of a balancing board. In the first case, there are states in which the agent is more or less stable, for example during the stance phase of the commonly used locomotion cycles [17], [18]. In the second case, a highly dynamic environment means that constant actuation is necessary to remain in a balanced state, and failing to do so will inevitably cause a collapse. For control in highly dynamic environments there works such as [19], which developed a momentum-based controller that is robust to disturbances, where one of their experiments utilized a balance board. Even more similar to our proposed work, in [20] the authors designed a sensor filter and a predictive control algorithm and managed to balance a real humanoid robot on a balance board for a few seconds. The common line between these two works is that they both employ traditional control techniques, where in our work we were interested in investigating the problem using state-of-the-art deep RL techniques.

We investigated the effects of action space shaping in the efficiency of training in a highly dynamic environment. We ran several experiments using PPO [21] to train an agent based on a humanoid robot [22] in a balancing task. The goal of the agent consisted of staying on top of a balancing board for as long as possible. As a humanoid, the agent counts with as many as 23 degrees of freedom (DoF) capable of actuation. We designed 6 distinct action spaces, called control modes, that shape the execution of actions in different manners. Each control mode differs in the number of outputted actions and constrains the propagation

[1]The authors are with the department of Electrical Engineering at National Taiwan Normal University (NTNU), 10610, Taipei, Taiwan 61075001H@ntnu.edu.tw, jslvjh@gmail.com, jacky.baltes@ntnu.edu.tw

[2]Guilherme Christmann guichristmann@gmail.com

Manuscript 92 submitted to 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids). Received July 15, 2022.

of the actions to the joints. The most free control mode (**FA**) counts with 23 actions, and controls each joint independently. On the other hand, the most constrained control mode (**PL**) outputs only 6 actions, which are mapped to the joints using a scheme that ensures the feet remain parallel to the ground. Our results show that the action space has a significant impact on the sample efficiency of training. In the best case, our implementation using *Isaac Gym* is capable of training an agent capable of balancing for the whole duration of an episode in under 40 minutes on a single GPU. We make our implementation open-source and freely available for all at: https://github.com/NTNU-ERC/Robinion-Balance-Board-PPO.

II. Methodology

We aim to show how the choice of action space affects the sample efficiency and final performance of RL. We designed a balancing problem solved by our agent in a simulation environment. The goal of our agent is to remain stable on a balance board while avoiding letting itself or the board touch the ground. In the following section, we describe the complete setup for our experiments, including the simulator and RL algorithms. We provide the details of our observation features, actions and reward function design. For action space shaping, we define and describe in detail 6 distinct control modes. For each control mode, we train a new agent using 5 different seeds. Finally, for improved policy stability and generalization, we also provide the parameters used for domain randomization.

A. Simulation Environment Setup

The agent used in this study is the humanoid robot *Robinion2* [22]. The humanoid agent counts with 23 degrees of freedom (DoF). The goal of the agent is to stay balanced on top of a balance board. The balance board is composed of two separate rigid body objects, a board and a roller, which are a modelled as cuboid and a cylinder, respectively. We used 3D CAD software to design the model of the agent and the balance board, compute its physical properties such as mass and inertia matrices, and import them into the simulation environment for training. The specifications of the agent and balance board are shown in Table I.

We design our simulation environment using *Isaac Gym* [23]. *Isaac Gym* is a physics simulation environment for RL research developed by NVIDIA with support for large-scale parallel training. With *Isaac Gym* we can create physics simulation scene directly in the GPU and an API provides access to read and write the state of multiple environments simultaneously. In all of our experiments, we created 4096 parallel environments. Each environment contains an instance of the humanoid agent and the balance board. Figure 1 shows a screenshot of the training scene with many parallel environments in *Isaac Gym*. The GPU used for our experiments was an RTX 2080 Super with 8 GB of VRAM.

TABLE I: Specifications of the humanoid agent, the board and the roller.

Specification	Agent	Board	Roller
Dimensions (mm)	300x112x920	730x280x37	113x113x430
Weight (kg)	7.5	2.3	1.5
DoFs	23	-	-
Actuators	XH540, XM540 XM430	-	-

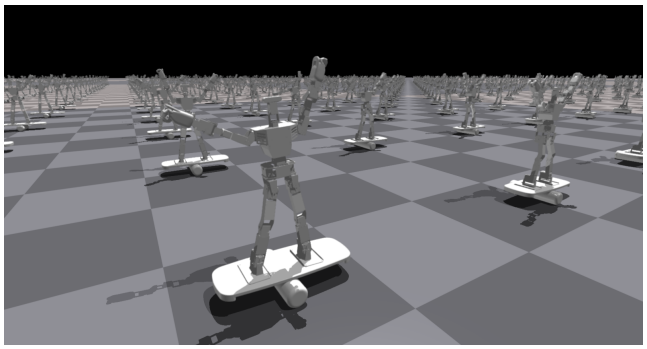


Fig. 1: Training environment in *Isaac Gym*. Our training loop simultaneously reads and writes the states of 4096 parallel environments.

B. Reinforcement Learning

The goal of reinforcement learning learn an optimal policy π that maximizes the expected return $J(\pi)$,

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)}[\sum_{t=0}^{T} \gamma^t r_t] \quad (1)$$

where $p(\tau|\pi)$ denotes the likelihood of a trajectory τ under the given policy π and $\sum_{t=0}^{T} \gamma^t r_t$ is the total reward collected in one trajectory, with r_t being the reward collected at time t, T denoting the length of each episode, and $\gamma \in [0, 1]$ is the discount factor that determines the weight of future rewards. We employ Proximal Policy Optimization (PPO) [21] to train our policies because it effectively optimizes Equation 1 and has demonstrated remarkable success in several continuous control tasks [14] [24]. Our network consists of three hidden layers with 256, 128, and 64 neurons respectively, with ELU activation function [25]. Our policy architecture is presented in Figure 2. Table II shows the hyper-parameter settings of training with PPO.

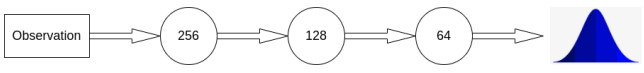


Fig. 2: Network Architecture. The network consists of three fully connected hidden layers of size 256, 128 and 64, respectively. Each hidden layer is followed by an ELU activation function.

At each timestep, the agent receives the latest observations from the environment. The observations are the inputs of the policy network, which models each output action as an

Manuscript 92 submitted to 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids). Received July 15, 2022.

TABLE II: Hyper-parameter settings for training of PPO.

Parameter	Value
Discount factor	0.99
Learning rate	3 x 10^{-5}
PPO clip threshold	0.2
PPO batch size	32768
PPO epochs	2250
KL threshold	0.008
Entropy coefficient	0
λ for generalized advantage estimation	0.95

independent normal distribution. The observations include information about the current state of the agent such as the current position and velocity of each joint, the agent's current pose and angular velocity and the position of its feet relative to its center-of-mass (CoM). Additionally, to facilitate convergence of training we include extra information such as the actions taken at the previous timestep, the angle of the balance board and position of the roller, distance to the ideal position of the feet on the board's surface, and a projection of the upwards the robot torso, which indicates how vertical the agent currently is. Table III presents the complete list of observations and their dimensionality. For the actions, we realized action space shaping with 6 different control modes, presented in detail in the next Section II-C.

To improve the robustness and generalization properties of the policy, as well as to introduce some variance to the training loop, we employed domain randomization [26]. During training, we randomly sample a diverse set of physics parameters for each environment. Table IV presents the parameters that are randomized as well as the ones that are kept fixed. Having the policy optimized on trajectories from a diverse set of physics parameters forces the agent to learn a behavior that can perform well across all of them.

TABLE III: Observations. The dimensionality of position and velocity of controlled joints and previous actions are vary from 6 to 23 depending on the control mode applied to the agent. The board angle is computed relative to the ground. The upwards projection is the orthogonal projection of the robot's orientation onto the z-axis.

Observations	
Feature	**Dimensionality**
Position of Controlled Joints	[6-23]*
Velocity of Controlled Joints	[6-23]*
Previous Actions	[6-23]*
Agent's Position	3
Agent's Angular Velocity	3
Agent's Orientation	3
Left and Right foot position (Relative to CoM)	6
Distance of Left Foot to Ideal Position	1
Upwards Projection of Robot Torso	1
Board Position	3
Board Angle	1
Roller Position	3

C. Action Space Shaping – Control modes

We train our agent with six different control modes to show the effect of action space shaping and choice of the control scheme can affect sample efficiency and final performance. In our experiments, we have two primary schemes: **free** and **parallel**. In free mode, the agent can control its joints freely, without any constraints. In parallel mode, the agent's feet are constrained to remain parallel to the ground. In the agent's mechanical configuration the thighs and calves have the same length. Thus, we can ensure the feet remain parallel to the ground by rotating the ankle and hip joints by the same amount, and simultaneously rotating the knee joints twice the target angle while moving the leg's pitch joints. Figure 3 shows the agent's lower body mechanics for the parallel type of control modes.

TABLE IV: Parameter ranges for domain randomization. Additive noise are added to the original value, while scaling noises are multiplied with the original value.

Parameters	Range	Distribution	Type
Actions	[0.0, 0.003]	Gaussian	Additive
Mass	[0.85, 1.15]	Uniform	Scaling
Stiffness of DoF	[0.75, 1.25]	Uniform	Scaling
Damping of DoF	[0.9, 1.1]	Uniform	Scaling
Initial DoF Position	[-0.03, 0.03]	Uniform	Additive
Initial DoF Velocity	[-0.1, 0.1]	Uniform	Additive
Gravity	9.81	Constant	-
Static friction	1.0	Constant	-
Dynamic friction	1.0	Constant	-

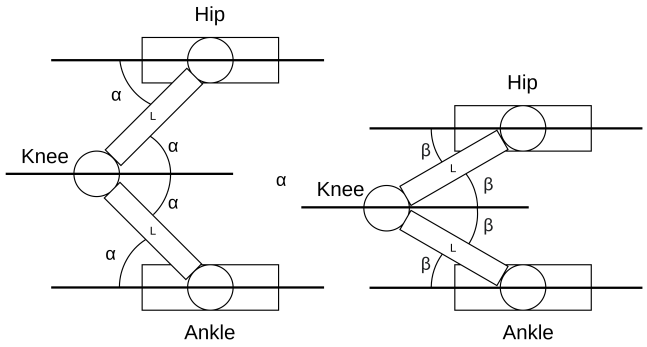


Fig. 3: Side view of the agent's lower body mechanics while moving in the parallel type of control modes. The 3 horizontal lines show that the knee angles will always be twice the angle of the hip and ankle joints.

When designing our experiments and choosing the action space we had two main questions: 1. How much impact does action space shaping have on the sample efficiency of training, and; 2. How critical is a humanoid's upper body to solve balancing problems? To answer both of these questions, we carefully designed 6 control modes. In the most free mode the agent can control all of the joints independently, where in the most constrained mode it can only control 6 joints. Some control modes allow for control of only the lower body joints and others allow for varying degrees of upper body control. The 6 control modes and their detailed description are presented below:

- **FA** – *Free Mode + All Joints*: The agent can control all of its joints without any limitation. Since there are no

constraints in this mode, this is considered the baseline control mode. The total number of actions in this mode is 23.

- **FL** – *Free Mode + Lower Body*: The agent can control any of its lower body's joints independently. The upper body's joints are set to a fixed pose and not controlled. The total number of actions in this mode is 10.
- **PA** – *Parallel Mode + All Joints*: The agent's feet are constrained to remain parallel to the ground. It can control the joints from both lower and upper body. The lower body actions are propagated using the parallel scheme. The total number of actions in this mode is 19.
- **PL** – *Parallel Mode + Lower Body*: The agent's feet are constrained to remain parallel to the ground. The agent can control any of its lower body's joints independently. The upper body's joints are set to a fixed pose and not controlled. The total number of actions in this mode is 6.
- **PLS-R** – *Parallel Mode + Lower Body + Shoulder Roll Joints*: The agent's feet are constrained to remain parallel to the ground. The agent can control any of its lower body's joints independently. Additionally, the agent can also control its shoulder roll joints. The remaining upper body's joints are set to a fixed pose and not controlled. The total number of actions in this mode is 8.
- **PLS-RP** – *Parallel Mode + Lower Body + Shoulder Roll Joints + Shoulder Pitch Joints*: The agent's feet are constrained to remain parallel to the ground. The agent can control any of its lower body's joints independently. Additionally, the agent can also control its shoulder roll and shoulder pitch joints. The remaining upper body's joints are set to a fixed pose and not controlled. The total number of actions in this mode is 10.

The dimensionality of both the action and observation spaces differ depending on the control modes. Table V shows the dimensions of the action space and observation space for the different control modes. To ensure reproducibility and robust results, for each control mode we train a policy from scratch using 5 random seeds. The graphs and tables presented in Section III are averaged over the 5 random seeds.

D. Reward function

In this section we present the details of our reward function. The reward function was designed in such a way to encourage the agent to remain balanced on top of the board.

The reward function is conditioned on board's angle to the ground, the orthogonal projection of the agent's orientation to the z-axis (upwards direction), the angular velocity of the board, and the distance between the agent's left foot and an optimal position on top of the board. Figure 4 shows the information components used in the formulation of the reward function.

The complete reward function r_t is a summation of four reward components:

$$r_t = r_t^b + r_t^p + r_t^a + r_t^d \tag{2}$$

The first component r_t^b encourages the agent to maintain the board parallel to the ground. It is computed according to the following equation:

$$r_t^b = 1 - 20 \cdot {\theta_t}^2 \tag{3}$$

where θ_t represents the board's angle relative to the ground at timestep t. The angle is computed relative to $\vec{x}$ which is orthogonal to the upwards vector of the word and runs parallel to the ground and is shown in Figure 4.

The second component r_t^p encourages the agent to maintain its body vertical to the ground and is defined as:

$$r_t^p = -(1 - proj_{\mathbf{o}}\mathbf{z}) \tag{4}$$

where $proj_{\mathbf{o}}\mathbf{z}$ represents the projection of the upwards vector from the agent's orientation against the world's z axis vector (upwards). Figure 4 shows that the value of $proj_{\mathbf{o}}\mathbf{z}$ is smaller the more tilted the agent becomes.

The third component r_t^a encourages the agent to maintain the board as stable as possible:

$$r_t^a = -\|\omega_t\|_2 \tag{5}$$

where ω_t represents the board's angular velocity at timestep t.

The fourth and last component of the reward function r_t^d encourages agent to place its foot in an ideal position on top of the board. It is defined as follows:

$$r_t^d = 1 - 20 \cdot {D_{lf}}^2 \tag{6}$$

where ${D_{lf}}^2$ is the distance between the current position of the left foot and the ideal position on the board.

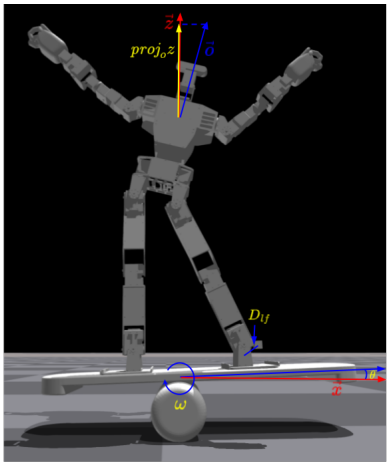
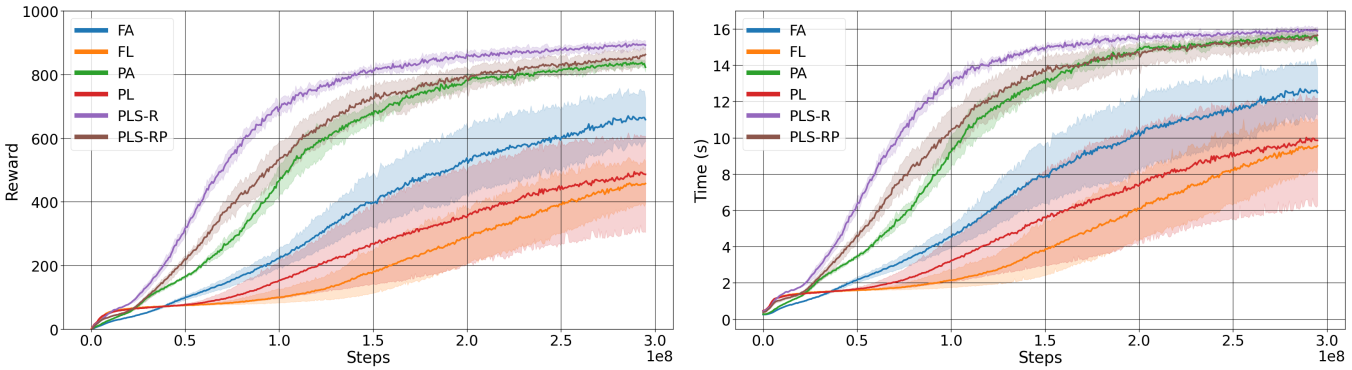


Fig. 4: Vector and angle components used by the reward function. $\vec{x}$, $\vec{z}$ are unit vectors in the direction of the world's x-axis and z-axis, respectively. The symbols in yellow are used in the reward function.



(a) Average reward versus training steps.

(b) Average time versus training steps.

Fig. 5: Training curves for the six different control modes, averaged over 5 random seeds. The top performing parallel control modes achieve much better sample efficiency than the control modes.

TABLE V: Size of the observation and action space for the different control control modes. In the parallel mode, the lower body joints are constrained to keep the feet parallel to the ground, which reduces the number of actions needed from the network.

Mode name	Dimension	
	Observation space	Action space
FA	93	23
FL	54	10
PA	93	19
PL	54	6
PLS-R	60	8
PLS-RP	66	10

TABLE VI: Evaluation trials for each control mode. The metrics were collected from 2048 validation episodes with maximum duration of 16 seconds and averaged for 5 different seeds for each control mode.

Control Mode	Reward	Steps	Tolerance
FA	765±62	852±66	90±11
FL	470±101	612±133	37±4
PA	866±11	954±14	94±6
PL	533±203	665±255	37±9
PLS-R	**922±14**	**974±13**	104±3
PLS-RP	900±16	967±16	**105±10**

III. Results

A. Experimental Evaluation

Using PPO, the policy is optimized with samples collected from 300 million timesteps in *Isaac Gym*. This process finishes in under 40 minutes in a single GPU. Figure 5 presents the training curves for the 6 control modes defined in Section II-C. Each line of the graph is averaged over 5 distinct random seeds.

After training is finished we run 2048 validation episodes and measure the performance of the policies in three metrics: average reward, average time to stay alive, and tolerance to disturbances. To measure tolerance to disturbances, we apply a force at the root of the agent in a random direction, starting from 0 Newtons and increase by 10 Newtons every 2 seconds until the agent collapses. For each control mode the data is averaged over the 5 random seeds and standard deviation is calculated, presented in Table VI.

B. Analysis of the Control Modes

Our results shows that action space shaping, i.e. the choice of control modes, had a significant impact in the sample efficiency and final performance of the RL policies. The training curves in Figure 5 show a large gap between the **parallel** mode methods **PLS-R**, **PLS-RP** and **PA** and the other control modes. Additionally, the variance for different starting seeds is lower which implies that the training is more stable and has a larger guarantee of convergence. The first question we sought to answer was to quantify the impact of action space shaping in sample efficiency. Using the reward value of 800 as a threshold reference, Figure 5 shows that the most efficient method is **PLS-R** crossing the reference line at about 140 million timesteps. The second best-performing method **PLS-RP** crosses the reference line after about 210 million timesteps. From these, we can infer that **PLS-R** can reach a good level of performance with 70 million fewer timesteps; it is 33% more sample efficient than the second best control mode.

The second question we sought to answer was determining if control of the humanoid agents upper body was critical to solve the balancing problem. Figure 5 shows poor sample efficiency and final performance, as well as huge variance to initial random seed for the control modes constrained only to lower body **PL** and **FL**. Similarly, the evaluation trials presented in Table VI also demonstrate sub optimal performance for these control modes. On the other hand, the effectiveness of the parallel mode constraint in combination with upper body control is notable. The top 3 constrained control modes **PL**, **PLS-R**, and **PLS-RP** substantially outperformed the naive baseline **FA** where all joints are controlled independently.

IV. Conclusion

In this paper, we presented a state-of-the-art RL method with a humanoid agent to tackle a balancing problem. Our

implementation can train a successful balancing policy in under 40 minutes on a single GPU. We explored the effect of action space shaping on the sample efficiency of training. We designed 6 distinct control modes, with varying constraints on the propagation of actions to the agent's joints. For each control method we averaged the results over 5 different random seeds. Our results showed the control modes had a significant impact on sample efficiency and variance of training. The constrained parallel control modes outperformed the naive baseline by a wide margin, and our best performing control method required 70 million fewer timesteps than the second-best method to reach a comparable performance.

In our future work, we plan to extend the findings from this work into the sim-to-real domain. Our goals are to deploy the policies learned in simulation with the real world robot. The biggest challenge to overcome will be handling the observation features that can't be easily estimated in the real world but were used in simulation of this work.

ACKNOWLEDGMENT

This work was financially supported by the "Chinese Language and Technology Center" of National Taiwan Normal University (NTNU) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan, and Ministry of Science and Technology, Taiwan, under Grants no. MOST 111-2918-I-003-003-, MOST 110-2923-E-003-001-MY3, and MOST 110-2221-E-003-023. We are grateful to the National Center for High-performance Computing for computer time and facilities to conduct this research.

REFERENCES

[1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, 2015.

[2] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke et al., "Scalable deep reinforcement learning for vision-based robotic manipulation," in Conference on Robot Learning. PMLR, 2018, pp. 651–673.

[3] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017, pp. 3389–3396.

[4] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray et al., "Learning dexterous in-hand manipulation," The International Journal of Robotics Research, vol. 39, no. 1, pp. 3–20, 2020.

[5] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," arXiv preprint arXiv:2004.12570, 2020.

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

[7] A. Kanervisto, C. Scheller, and V. Hautamäki, "Action space shaping in deep reinforcement learning," in 2020 IEEE Conference on Games (CoG). IEEE, 2020, pp. 479–486.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[9] X. B. Peng and M. 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, 2017, pp. 1–13.

[10] M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller, "Continuous-discrete reinforcement learning for hybrid control in robotics," in Conference on Robot Learning. PMLR, 2020, pp. 735–751.

[11] S. Kajita, O. Matsumoto, and M. Saigo, "Real-time 3d walking pattern generation for a biped robot with telescopic legs," in Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. no. 01ch37164), vol. 3. IEEE, 2001, pp. 2299–2306.

[12] S. Kagami, K. Nishiwaki, T. Kitagawa, T. Sugihara, M. Inaba, and H. Inoue, "A fast generation method of a dynamically stable humanoid robot trajectory with enhanced zmp constraint," in Proc. of IEEE International Conference on Humanoid Robotics (Humanoid2000), 2000.

[13] A. S. Anand, G. Zhao, H. Roth, and A. Seyfarth, "A deep reinforcement learning based approach towards generating human walking behavior with a neuromuscular model," in 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids). IEEE, 2019, pp. 537–543.

[14] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," Science Robotics, vol. 7, no. 62, p. eabk2822, 2022.

[15] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," Science robotics, vol. 5, no. 47, p. eabc5986, 2020.

[16] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in Conference on Robot Learning. PMLR, 2022, pp. 91–100.

[17] F. Lacquaniti, Y. P. Ivanenko, and M. Zago, "Patterned control of human locomotion," The Journal of physiology, vol. 590, no. 10, pp. 2189–2199, 2012.

[18] K. Yin, K. Loken, and M. Van de Panne, "Simbicon: Simple biped locomotion control," ACM Transactions on Graphics (TOG), vol. 26, no. 3, pp. 105–es, 2007.

[19] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014, pp. 981–988.

[20] J. Baltes, C. Iverach-Brereton, and J. Anderson, "Sensor filtering for balancing of humanoid robots in highly dynamic environments," in 2013 CACS International Automatic Control Conference (CACS). IEEE, 2013, pp. 170–173.

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

[22] J. Jeong, J. Yang, and J. Baltes, "Robot magic show: human–robot interaction," The Knowledge Engineering Review, vol. 35, 2020.

[23] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa et al., "Isaac gym: High performance gpu-based physics simulation for robot learning," arXiv preprint arXiv:2108.10470, 2021.

[24] 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). IEEE, 2018, pp. 3803–3810.

[25] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in 2018 Chinese control and decision conference (CCDC). IEEE, 2018, pp. 1836–1841.

[26] 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). IEEE, 2017, pp. 23–30.