Two-Policy Cooperative Transfer for Alleviation of Sim-to-Real Gap

Liangdong Wu
School of Artificial Intelligence
University of Chinese Academy of Sciences
Beijing, China
wuliangdong2018@ia.ac.cn

Fangzhou Xiong
Meituan
Beijing, China
xiongfangzhou@meituan.com

Zhiyong Liu
Institute of Automation
Chinese Academy of Sciences
Beijing, China
zhiyong.liu@ia.ac.cn

***Abstract*—The main difficulty of sim-to-real is the reality gap between the source domain and the target domain. In order to solve it, various methods where domain randomization is the mainstream have been emerged, whose essence is to make the single policy more robust. In contrast, we propose a novel transfer method, namely two-policy cooperative transfer, whose core is that one policy (task policy) is used to complete the task and another policy (gap policy) aims to assist the former to cover the gap, hence we can focus on the training of task and the overcoming of gap respectively. Based on this method, the setting of the learning objective of gap policy depends on the transfer situation of deploying task policy into real system, besides how to conduct the cooperation of the both lies in the threshold reflecting gap and the coupling of output actions of two policies. For the typical contact-rich gap in the dynamics field, we design an adaptive object pushing experiment based on UR3 robot, and verify the effectiveness of the proposed method.**

***Keywords*—Sim-to-real, Reinforcement learning, Robot control**

I. Introduction

In recent years, robot control based on reinforcement learning has gradually become an important direction in artificial intelligence field [1]. Its essence lies in training a policy achieving the maximum expected reward to complete the task through trial and error of large data samples [2]. Up to now, almost of all policy training is completed in virtual simulation, while direct training in the real robot system is faced with a series of problems of high cost, high risk and low efficiency [3]. Therefore, how to guarantee the effect of deploying the policy obtained via simulation training to the real system (sim-to-real transfer), specifically eliminating the impact of reality gap [4] between simulation and reality, has increasingly become a hot topic attracting wide attentions.

The difference between the simulated source domain and the real target domain leads to the gap [4, 5], which further render the application effect of the policy transferred into the real system is far less likely than that of the simulation. For this issue, some research advances have also emerged gradually, among which the more representative one is domain randomization (DR) [5], which was initially applied to cover sim-to-real gap in visual images, and later related studies employed this idea to narrow the gap in the field of dynamics [3, 6]. In this paper, our research focuses on the gap in dynamics, and a typical application scenario is the contact-rich [7] transfer experiment, in which the properties of the experimental object are difficult to accurately simulate.

In the initial phase of this research, we evaluated the performance of DR for gap bridging in dynamics, and found that it was effective but still seemed to fall short of the desired results. Additionally, some recent literature [8, 9] have raised doubts about DR. After more investigations, such as [3, 6, 10–12], we find that these methods including DR are basically single policy transfer, hence, the policy inevitably requires the two abilities at the same time, that is, completing the task and overcoming the gap. For now, there is no theoretical support for the coupling training of the two abilities to ensure that each ability can meet the established requirements. Meanwhile, the training complexity will also increase significantly [12]. Therefore, we tentatively propose a two-policy training mode, one policy is focused on the acquisition of task skills (task policy), another policy is focused on how to make up the gap (gap policy). The two policies will be deployed to the real system simultaneously, if there is no gap effect temporarily, the task policy will be executed; otherwise, the coupling action of the output actions of the two policies will be executed to eliminate the gap and continue the task at the same time. We define it as two-policy cooperative transfer.

For the proposed method, the setting of learning objectives of the gap policy depends on the transfer situation of deploying task policy into real system, that is, the pre-transfer of task policy. Although this method needs to train two policies, it can make more timely and reasonable adjustments to the impact of gap, so as to distinctly improve the effectiveness and success rate of sim-to-real transfer.

The main contributions in this paper are as follows: (1) We propose two-policy cooperative transfer to make up for the impact of gap. (2) Based on our method, we can focus on the training of task and the overcoming of gap respectively. (3) Experimental results show that the proposed method has a satisfactory covering effect for the contact-rich dynamic

This work is supported by Science and Technology Innovation 2030–"New Generation of Artificial Intelligence" Major Projects (2020AAA0108902), Strategic Priority Science and Technology Program of Chinese Academy of Sciences(Category B)(XDB32050100) and Dongguan City Core Technology Cutting-edge Project (2019622101001)

gap, and is significantly better than the domain randomization method.

II. Related Work

Sim2real transfer has gradually become a topic of widespread concern, and the main difficulty is how to bridge the gap. For this issue, there have been a lot of research progress, which can be roughly divided into three categories [13]: system identification, domain adaptation, and domain randomization (DR).

The purpose of system identification is to accurately obtain the corresponding simulation model through the identification of real system and objects, so as to minimize the differences between simulation and reality. Hwangbo et al [14] trained the deep network model to conduct the mapping from motor instructions to torque based on the data of the actual system, and the transfer of walking and running on the quadruped robot is realized.

Domain adaptation is originated from computer vision, which aims to study how the visual-based model trained by the source domain adapts to the target domain that has never been encountered. The idea is also used to solve reality gap. Christiano et al [10] trained the inverse dynamics model through the data of the actual system to make the simulation model closer to the actual system, thus improving the transfer effect of the policy.

DR aims to experience various simulation environments as much as possible in the simulation training process (such as all kinds of simulation images, simulation objects with various dynamic characteristics in a certain range, etc), and try to include the approximate actual situation in them, so as to achieve a strong robust policy. Sadeghi et al [11] trained the visual-based quadrotor control policy by using random composite rendering scenes. In the paper [5], the authors used various images to train the policy for realizing the grasping task of the robot.

In order to solve the gap of dynamics, Andrychowicz et al [12] trained the policy of controlling object rotation by randomly setting physical parameters such as friction and delay, and transfer it to the Baxter robot without additional fine-tuning. In the paper [6], the authors explored how to reduce the parameter space of DR to improve training efficiency. In view of how to solve the dynamics gap effectively, Valassakis et al [9] comprehensively evaluated several main methods at the present stage, expressing that the result of DR is not ideal enough. The paper [8] also raised doubts about domain randomization.

III. Method

For now, the policy transfer from simulation to reality is basically the transfer of a single policy. In view of the gap reflected in the transfer procedure, most of scientists are devoted to the research on how to make the single policy more robust and more generalized, so as to cover the gap. In contrary to this idea, we propose a concept which is the two-policy cooperative transfer, specifically meaning the cooperation of task policy and gap policy, with the former focusing on task skills and the latter focusing on overcoming gap. Further, we clarify the method about how to train the two policies and transfer them to the real system cooperatively. Figure 1 shows the main logical framework of our proposed method, and more details are illustrated in Figure 2.

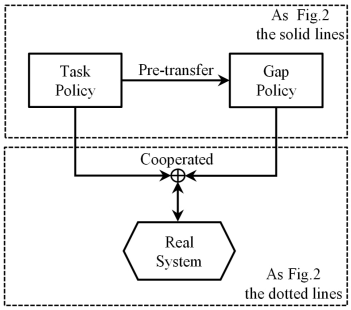


Fig. 1. The main logical framework of the two-policy cooperative transfer

A. Task Policy

In order to make the agent possess a certain skill or be able to complete a certain task, the relevant algorithm of reinforcement learning is used to train a policy that is inputted environmental states and output the actions, so as to realize the maximum reward return $J(\theta) = E_\theta[\sum_{t=1}^{\infty}\gamma^t r_t]$. We define this kind of policy as task policy $\pi_{T\theta}(a_{Tt} \mid s_t)$, and its action is further defined as a_{Tt}. T is the symbol representing task. The training of task policy is the same as the policy training of reinforcement learning in general sense. It is not be expanded here. For more details, refer to the relevant literature [3, 5].

For the training of task policy, we do not take consideration into narrowing the gap. Therefore, in the training process, we only focus on whether the policy can acquire the required task capability.

B. Gap Policy

In this section, we illustrate how to train a specific policy for the gap, namely gap policy $\pi_{G\theta}(a_{Gt} \mid s_t)$. First, we hold the opinion that it is sufficient for task policy if its test results perform well in simulation. Further, we regard the state of task policy at each step in the simulation test as the reference state, and the deviation state obtained when the task policy is deployed into the actual system as the gap state. Based on these two different classes of states and possible modification actions, new learning objective are designed to train gap policy. G is the symbol representing gap.

After the task policy $\pi_{T\theta}(a_{Tt} \mid s_t)$ is obtained through training under simulation, it is first tested in simulation, and the state s_t of the experimental object in each step is recorded and collected as a state reference set s_{traj}:

$$s_{traj} = (s_1, s_2, s_3, ...s_k, ...s_n) \tag{1}$$

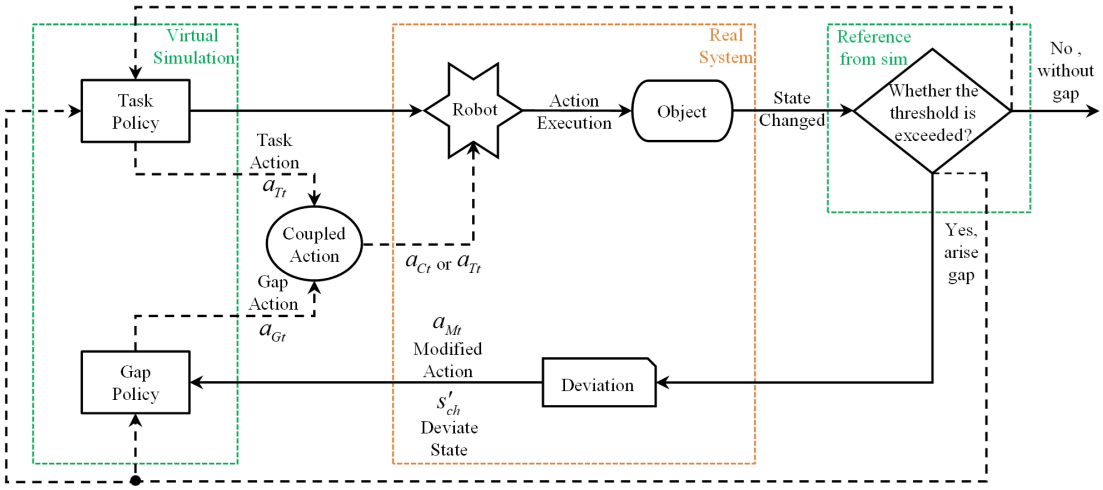


Fig. 2. The solid lines represent how to acquire the prior knowledge about designing the learning objective of gap policy through the task policy pre-transfer, and then train it in simulation. The dotted lines represent the operation logic and action selection of the two-policy cooperative transfer. The solid lines of robot and object are collinear with the dotted lines

Then the task policy is transferred to the actual system, and the state s'_t of the actual object is recorded at each step to obtain the actual state set s'_{traj}:

$$s'_{traj} = (s'_1, s'_2, s'_3, ...s'_k, ...s'_n) \tag{2}$$

And we can get the set of deviations e_{traj} between s_{traj} and s'_{traj}, as:

$$e_{traj} = (e_1, e_2, e_3, ...e_k, ...e_n) \tag{3}$$

Each deviation e_t is compared with the threshold c in chronological order. Based on experience, we set the threshold c at $\pm 5\%$ of the corresponding reference state. The first deviation exceeding the threshold is set as e_c. And then the simulation state and the actual state at the corresponding time is respectively set as s_c, s'_c. The reason for adopting the first deviation is that it is most timely to make adjustments to overcome the impact of gap at this moment. Otherwise, the accumulation of errors will become larger, and the difficulty of adjustment will increase significantly. Due to the influence of the gap, each actual trajectory is likely to be different, assuming that k practical experiments are carried out, the state-deviation trajectory τ_{es} can be obtained, as:

$$\tau_{es} = (e_{c1}, s_{c1}, s'_{c1}, ...e_{ck}, s_{ck}, s'_{ck}) \tag{4}$$

The three elements with the same subscript in τ_{es} are classified as one class, and the class h is arbitrarily taken out of it, with time t. Suppose there is an action a_{Mt}, render that:

$$\widehat{s}'_{ch} \sim P_R(\widehat{s}'_{ch} \mid s'_{ch}, a_{Mt}) \tag{5}$$

$$0 \leq \widehat{s}'_{ch} - s_{ch} < e_{ch} \tag{6}$$

After the execution of the action a_{Mt}, the updated state $\widehat{s}'_{ch}$ is obtained from the actual system transition probability distribution P_R, which is closer to the simulation state s_{ch}, so as to realize the state modification of the object. Actions a_{Mt} can be designed with reference to specific system and task, and symbol R and M represents reality and modification respectively.

Thus, we get two important prior knowledge of the training setting of gap policy $\pi_{G\theta}(a_{Gt} \mid s_t)$: preliminarily reflecting various states s'_{ch} of the gap, and each action a_{Mt} to modify these states. Further exclude possibly similar states and actions, retain representative ones (S'_{ch}, A_{Mt}), and construct corresponding learning objective based on them. After training, the gap action a_{Gt} should be equivalent to the modified action a_{Mt}. The corresponding logic is shown as the solid lines in Figure 2, and the setting of simulation training environment refers to the training situation of task policy.

C. Two-Policy Cooperative Transfer

Task policy $\pi_{T\theta}(a_{Tt} \mid s_t)$ and gap policy $\pi_{G\theta}(a_{Gt} \mid s_t)$ are transferred into the real system to overcome the gap and complete the task. In this process, when the state of the experimental object does not reach the threshold c, we deem to that the gap does not appear at the moment, and the task action a_{Tt} given by the task policy can be performed continually. When the threshold c is reached, the gap appears, then the gap action a_{Gt} given by the gap policy is coupled with the task action a_{Tt} to obtain the coupled action a_{Ct}, which is executed to modify the object state and make it drop below the threshold c.

The coupling of actions can be simply expressed as:

$$a_{Ct} = a_{Tt} \oplus a_{Gt} \tag{7}$$

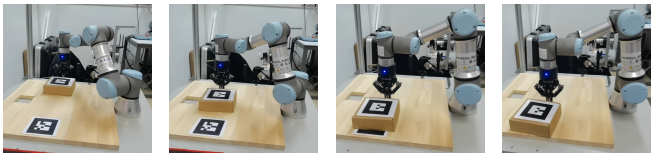
In certain situation it might be adding vectors. The logic of coupled actions execution is shown in Figure 2 with the dotted lines.

IV. Experiments

We set up a simulated and realistic UR3 robot object pushing experimental platform respectively, as figure 3 (a)(b), then conduct policy training through the former and policy transfer

(a) (b) (c) (d)

Fig. 3. The simulation (a) and real experiment platform (b) for UR3 robot object pushing, besides the box loaded 1000g iron block into the upper part of the center (c) and the lower part of the center (d) respectively



(a) (b) (c) (d)

Fig. 4. The experiment process of object pushing by UR3

through the latter. Furthermore, we illustrate the experimental process based on the proposed method, also compare and evaluate it with domain randomization.

A. Experimental Setup

Virtual simulation: Our simulation environment is composed of Mujoco physics engine [15] and OpenAI Gym library [16]. We use a UR3 robot equipped with two-finger gripper at the end, which will be closed during the experiment to push the experimental object. The simulation step size is 0.002s, and each episode contains 50 steps. The output of the policy is the Cartesian coordinate at which the end of the robot gripper should reach. The input of the policy includes the Cartesian position coordinates of the end of the robot gripper, target position and the position attitude of the object.

Real system: The system communication is set up through the robot operating system (ROS). The Kinect2 camera is to get the target pose and position attitude of the experimental object, and UR3 robot is equipped Robotiq two-finger gripper with closed state. The end of the robot gripper is set perpendicular to the motion plane and the height is fixed to ensure that the end can avoid bumping.

The experimental object and process in reality: The object used in our experiment is a box with QR code, being $(0.15m * 0.15m * 0.15m)$ and $60g$. For comparison, we further load $1000g$ iron block into the non-geometric center position of the box, such as the upper part of the center and the lower part of the center, shown as Figure 3 (c)(d), to significantly increase and change the whole box mass, friction force and center of gravity position. We set the mark of the success of the experiment as that the distance between the box and the target pose is less than $0.02m$, meanwhile the deflection angle is not more than $5°$. The initial position of object is $(x, y) = (0.36m, 0.15m)$, and the target point is $(x, y) = (0.11m, 0.40m)$. The object pushing experiment process is shown as Figure 4.

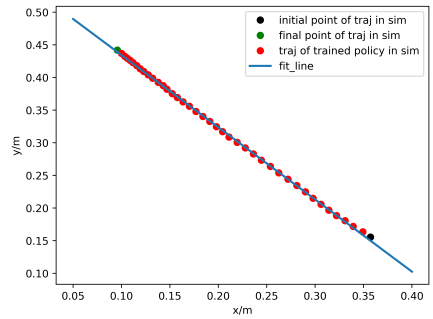


Fig. 5. The reference trajectory and fitting line of the pushed object in the simulation test

(a) (b)

Fig. 6. (a) and (b) illustrate the two types of deflection

B. The Application of Proposed Method

Task policy: For the task policy training of UR3 robot object pushing experiment, the corresponding simulation environment is designed as described in 4.1 section. The reinforcement learning algorithm used in training is SAC [17], combined with the use of HER [18]. The neural network Settings and hyperparameters of the algorithm program are set using the corresponding default Settings in the Stable-baselines library [19]. For the design of the reward function, we take the distance function between the object and the target position, as:

$$r(s_t, a_t) = -d_t/d_0 \tag{8}$$

Where, d_t and d_0 represents the distance of the current moment and the initial moment respectively. Additionally, we fixed the initial position and target position of the experiment to reduce the training difficulty and generalization ability of the task policy, so as to highlight the impact of gap and the effectiveness of the proposed method.

After training, we test the task policy in simulation, and record the states of each step of the object moving on the flat surface location, then gather into a reference trajectory and infer the fitting line, as shown in figure 5.

Gap policy: The task policy is deployed into the UR3 robot system, due to the uncertainty about the physical properties of the actual object, reality gap appears. Specifically, the position and deflection of the actual object at each moment will gradually deviate from the corresponding state of the reference trajectory, namely the deviation e_t. From analysis, there are actually no more than two types of deflection at the initial phase of arising deviation, that is, deflection along clockwise

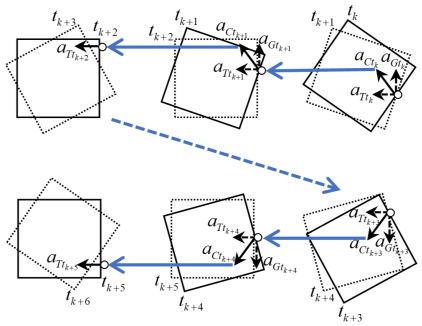


Fig. 7. the actions of the two policies are coupled and executed

(a) (b)

Fig. 8. Schematic diagram of the learning objective of gap policy

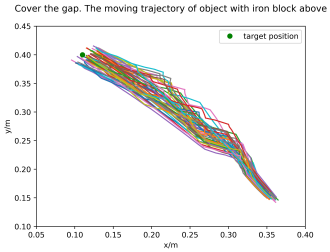
or counterclockwise direction of original object plane, with different specific degrees, as shown in Figure 6 (a)(b).

For the two direction deflections, obviously, the corresponding modified adjustment is the action from the oblique direction, defining it as coupled action a_{Ct}. While the action a_{Tt} given by task policy is along the direction of motion and is continuous, we can reverse infer that the action a_{Gt} given by gap policy should be perpendicular to the direction of motion up or down, as figure 7. Our output action is the coordinate of the gripper end, hence the coupling of actions in this experiment refers to the addition of coordinate vectors.

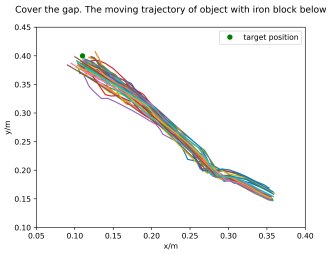
Furthermore, based on the deviation states and the modified actions inferred from the analysis, we can design new learning objective and obtain the gap policy through simulation training. The vertical view of learning objective is shown in Figure 8. When clockwise deflection or counterclockwise deflection occurs as shown in Figure 8 (a) or Figure 8 (b), the learning objective is how to move the end from the starting point A to the target point B. The AB distance in this experiment is set as 0.10m.

For simulation training of gap policy, its relevant Settings, such as reinforcement learning algorithm, hyperparameters, network setting and reward function, are the same as those of task policy, only the learning objective is different.

Two-policy cooperative transfer: The task policy and the gap policy are transferred to the actual system together. After each execution step, the state is compared with the threshold. If the threshold is not exceeded, only execute the task action at the next step; otherwise, execute the coupled action. The entire process can also be referenced in Figure 7 and the dotted lines of Figure 2.



(a)



(b)

Fig. 9. The motion trajectories of the box loaded iron block through our method

TABLE I
DYNAMIC PARAMETERS AND THEIR RANGES IN SIMULATION

Parameters	Range
Mass	[0.05, 1.10] kg
Sliding Friction Coefficient	[0.2, 2.0]
Torsion Friction Coefficient	[0.01, 0.1]
Rolling Friction Coefficient	[0.005, 0.05]

C. Results of Comparisons

We use the single transfer of task policy as the baseline and domain randomization [5, 9] as the comparison method to conduct a comprehensive evaluation of the object pushing experiment with our proposed method.

We employ domain randomization for object's parameters to train a policy with some generalization. Details are shown as Table I.

Each group of experiments is conducted for 50 times, and the corresponding success rate is recorded. The specific results are shown in Table II.

The results show that the baseline method has certain task completion ability only for the empty box, but it can't complete the experiment after loading iron block. For domain randomization, although it can enhance the generalization and improve the success rate partly, the overall effect is not ideal.

TABLE II
THE SUCCESS RATE OF THREE TRANSFER METHODS FOR THREE TYPES OF BOXES

Three Types of Boxes	The Baseline	Domain Randomization	Ours
Empty Box	56%	68%	**92%**
Iron Block into The Upper	0%	10%	**84%**
Iron Block into The Lower	0%	8%	**82%**

In contrast, our proposed method is robust to the changes of physical properties of box and achieves relatively high success rate. As Figure 9 (a) and (b), the motion trajectories of the box loaded iron block reflect that based on our method, there are adaptive real-time adjustments in the pushing process to eliminate the influence caused by gap.

It should be pointed out that for the task policy of single transfer and domain randomization, we did not retrain targeted, but carried out experiments on three types of boxes with the same policy and set of procedures. In the experiment based on our method, we did not make any secondary adjustment to the task policy and the gap policy, and just ran the same set of programs based on the same policy, which also reflected the strong robustness of the proposed method, and there was no need to retrain the policy for the change of the physical properties of the experimental object.

V. Conclusion

In order to solve the gap between simulation and reality, specifically dynamics field, we propose two-policy cooperative transfer, which is different from the previous other methods around single policy transfer. The basic intention of the proposed method is that the task policy is responsible for the acquisition of task skills and the gap policy is used to assist the former to cover reality gap, with cooperation of the two policies to achieve strong robust transfer. Consider that the setting of the learning objective of gap policy depends on the transfer situation of task policy, it embodies the characteristics of single policy pre-transfer. This method provides a new way of thinking for alleviating sim-to-real gap. The adaptive object pushing experiment based on UR3 robot verifies the effectiveness of the proposed method. In the future, we will research how to apply this method to more complex tasks and higher dimensional robots.

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[2] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839, 2020.

[3] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[4] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995.

[5] 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.

[6] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[7] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Florian Golemo, Melissa Mozifian, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. Perspectives on sim2real transfer for robotics: A summary of the r: Ss 2020 workshop. *arXiv preprint arXiv:2012.03806*, 2020.

[8] Zhaoming Xie, Xingye Da, Michiel van de Panne, Buck Babich, and Animesh Garg. Dynamics randomization revisited: A case study for quadrupedal locomotion. *arXiv preprint arXiv:2011.02404*, 2020.

[9] Eugene Valassakis, Zihan Ding, and Edward Johns. Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics. *arXiv preprint arXiv:2008.06686*, 2020.

[10] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

[11] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[12] 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.

[13] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[14] 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), 2019.

[15] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[18] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[19] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.