Learning to Walk without Dynamics Randomization

Jeremy Dao, Helei Duan, Kevin Green, Jonathan Hurst, Alan Fern
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, Oregon
{daoje, duanh, greenkev, jonathan.hurst, alan.fern}@oregonstate.edu

***Abstract*—The data-heavy nature of reinforcement learning often dictates that control policies must be learned in simulation and then transferred to hardware. This sim-to-real transfer is not trivial and the model differences between simulation and reality usually means that naively trained policies will not transfer. Almost all of the recent robotics reinforcement learning work has relied on some form of dynamics randomization or model adaptation in order to overcome the challenge of sim-to-real. However, we find that we are able to achieve sim-to-real transfer without relying on these techniques. Furthermore, we accomplish this for the difficult and highly dynamic task of bipedal locomotion, transferring high performance policies capable of walking at 2.6 m/s directly to the bipedal robot Cassie. We hypothesize and discuss several possible reasons we do not require dynamics randomization and identify possible new research directions that our results raise.**

I. Introduction

A large part of reinforcement learning robotics research focuses primarily on learning control policies in simulation [2, 8, 6, 12]. The practical constraints of physical robots can make learning on hardware difficult, time-consuming, and even unsafe. However, this presents the challenge of transferring simulation trained policies onto the physical hardware, where differences between the simulation model and real world can cause the policy to behave differently than expected. For example, if the physical robot is heavier than expected, a simulation trained policy would undershoot the control and could fail in completing the desired task.

The most common ways to solve this issue is to use dynamics randomization [5, 2, 8, 3, 1] or some form of model adaptation [4, 11]. However, we present real hardware results using simulation trained policies without any dynamics randomization or model adaptation. Only a single, unchanging model is used during training and the resulting policy is run directly on hardware. Furthermore, our model is sufficient enough to achieve the fastest walking speed ever recorded for the bipedal robot Cassie.

II. Methods

Our methods build off of the work presented in Xie et. al [10] with a few key differences.

A. Policy Input

While [10] included an expert reference trajectory in the input to the neural network policy, further research has found this to be ultimately unnecessary. Instead of the reference trajectory we directly input a clock signal and a commanded speed. This, along with the estimated robot state, makes our state input be of size 49.

The clock signal consists of sine and consine functions and is used to synchronize the motion and produce a cyclic output. In order to control the speed of the robot, we input a single floating point value into the policy that corresponds to the desired center of mass speed of the robot. This desired speed is also used the reward calculation, detailed in section II-C.

B. Training Procedure

Similar to [10] we first train to match an expert reference trajectory, then use it as a starting point to bootstrap the learning and optimize the policy with a more "natural" reward function like matching a desired center of mass speed. However, we expand upon this process by subsequently training on larger and larger ranges of desired speeds and stepping frequencies, eventually allowing us to learn policies capable of walking at speeds between 0 and 3 m/s and stepping frequencies between 1.19 and 2.38 Hz. In total, we train in 4 separate stages: (1) to match the reference trajectory at speeds between 0 and 1 m/s, (2) to match a commanded forward velocity between 0 and 1 m/s at a stepping frequency of 1.19 Hz, (3) we expand the commanded speed range to $[0, 2]$ m/s and the stepping frequency range to $[1.19, 1.76]$ Hz, and (4) we expand the commanded speed range to $[0, 3]$ m/s and the stepping frequency range to $[1.19, 2.38]$ Hz.

C. Reward

The first reward function used is a reference trajectory matching reward as described in [10].

After a policy is trained using the reference trajectory matching reward, we use it as the initial policy to subsequently train using a "speed matching" reward function, This reward incentivizes walking forward in a straight line at the commanded speed while facing forward. It consists of 4 terms: a center of mass velocity term (absolute value of the difference between the pelvis x velocity and the desired x velocity), an orientation term (difference between the current pelvis orientation and forward facing orientation), a "straight line" term (magnitude of pelvis y position), and a "drift" term (magnitude of pelvis y velocity). Similar to the reference trajectory reward, each term is normalized and a weighted sum is taken.

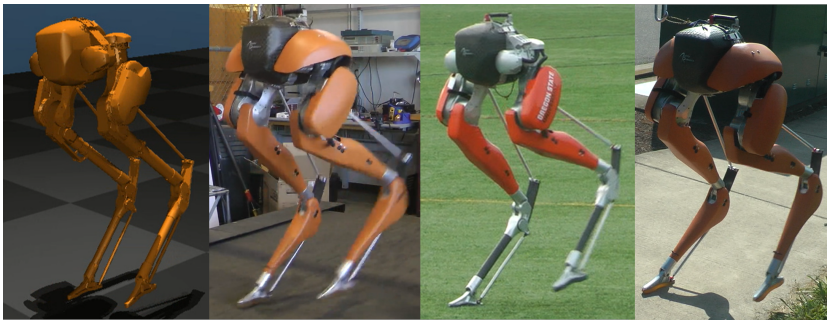

Fig. 1. The same simulation trained policy is shown in the 4 above still frames. The policy is able to produce the same motion when transferred directly to hardware and can do so over different unseen terrains (rubber treadmill, artificial turf, and concrete asphalt are shown here).

III. Results

Using our method, we are able to train policies that are capable of walking at speeds between 0 and 3 m/s and at stepping frequencies between 1.19 and 2.38 Hz in simulation. We are then able to directly transfer these policies to hardware where we see a very similar motion and can get our Cassie robot to walk at a max speed of 2.6 m/s. Attempting to walk at speeds higher than this causes the robot to become unstable. We believe that this is because the robot physically doesn't have enough inertia to oppose the momentum of the leg swinging at higher speeds. Using this same policy we run hardware tests on three different types of surfaces: (1) the rubber tread of our treadmill, (2) artificial turf, and (3) concrete asphalt. We find that the policy can generalize and produce very similar behaviors on the varying terrains.

Furthermore, our policies are able to generalize to unseen stepping frequencies. A policy trained on only a single stepping frequency of 1.19 Hz is able to handle unseen larger stepping frequencies. Unfortunately, due to the Coronavirus pandemic, we are currently only able to show this in simulation.

We believe that these generalizations show the validity of our results and are indicative of successful sim-to-real transfer. Videos of these experiments can be seen in at https://www.youtube.com/watch?v=_YAmx3xFMz8.

IV. Discussion

Though we are achieve great hardware results, we are unsure of exactly why we can do direct sim-to-real transfer. However, we hypothesize several possible reasons and discuss them below.

A. Modeling

Extra care was taken to model certain aspects like actuator delay and reflected inertia that other works tend to leave out [1, 3, 4, 11]. The reflected inertia of the motors is included in our MuJoCo [9] simulation model (defined as "armature" in MuJoCo) and actuator delay is modeled by delaying when torque commands are actually executed. Desired motor torques are stored in a buffer first, before actually being applied 6 times (0.003 seconds) after being "sent" by the simulator. It is possible that these modeling aspects are extremely important for sim-to-real and are the reason for our success. Actuator delay is undoubtedly present in all hardware systems and taking this into account can help the policy be more robust, by encouraging it to not require a specific command to happen a specific time. This is especially important in contact-heavy tasks where relying on a contact to happen at a specific time can make the control very brittle.

However, previous works have also observed these modeling aspects to not be enough for sim-to-real transfer. [2] utilized a learned actuator model from hardware data and [8] included latency in their model, but both works still required dynamics randomization to achieve successful transfer.

We plan to test this hypothesis in the future by taking these features out of our model and retraining policies using the same training procedure. Whether or not these polices can successfully transfer to hardware will provide some evidence for whether or not reflected inertia and actuator delay are important for sim-to-real transfer.

B. Hardware

One major difference between our work and other works that show successful sim-to-real transfer is that our hardware has in built compliance through the use of springs. Springs have been shown to make a system robust to contact timing [7] and it could be possible that this extra compliance can make the policy inherently more robust to model differences. Ideally, we would test this by swapping out the springs in Cassie with rigid plates and seeing if we can still achieve sim-to-real transfer. Unfortunately, due to the design of the hardware, this can not be done without damage to the robot. Our plan is to test this with sim2sim experiments. We could train multiple policies each with different fixed inaccurate models and then test them on a soft spring model and a rigid spring model. If more policies work on the soft spring model than the rigid spring, it might indicate that hardware compliance through springs allow the policy to get away with a more inaccurate model.

Acknowledgment

We thank Zhaoming Xie and Michiel van de Panne for their help and collaboration in starting this work. We thank Pedro Morais for his help in setting up our training framework. This work is supported by NSF Grant No. IIS-1849343, DGE-1314109 and DARPA Contract W911NF-16-1-0002.

REFERENCES

[1] Van Baar. Sim-to-Real Transfer Learning using Robustified Controllers in Robotic Tasks involving Complex Dynamics. 2019.

[2] 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):1–20, 2019. ISSN 24709476. doi: 10.1126/scirobotics.aau5872.

[3] Ofir Nachum, Michael Ahn, Hugo Ponte, Shixiang Gu, and Vikash Kumar. Multi-Agent Manipulation via Locomotion using Hierarchical Sim2Real. pages 1–14, 2019. URL http://arxiv.org/abs/1908.05224.

[4] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–17, 2019.

[5] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3803–3810, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8460528.

[6] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. *IEEE International Conference on Intelligent Robots and Systems*, pages 2219–2225, 2006. doi: 10.1109/IROS.2006.282564.

[7] Siavash Rezazadeh, Christian Hubicki, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Andy Abate, and Jonathan Hurst. Spring-Mass Walking With ATRIAS in 3D: Robust Gait Control Spanning Zero to 4.3 KPH on a Heavily Underactuated Bipedal Robot. Dynamic Systems and Control Conference, 10 2015. doi: 10.1115/DSCC2015-9899. URL https://doi.org/10.1115/DSCC2015-9899. V001T04A003.

[8] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.010. URL http://www.roboticsproceedings.org/rss14/p10.html.

[9] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. *IEEE International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. ISSN 21530858. doi: 10.1109/IROS.2012.6386109.

[10] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel Van De Panne. Learning Locomotion Skills for Cassie: Iterative Design and Sim-to-Real. *Conference on Robotic Learning*, (CoRL), 2019.

[11] Wenhao Yu, Visak CV Kumar, Greg Turk, and C. Karen Liu. Sim-to-Real Transfer for Biped Locomotion. 2019. URL http://arxiv.org/abs/1903.01390.

[12] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J. Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:12861–12874, 2019.