Sim2Real Transfer for Quadrupedal Locomotion

Jeremiah Coholich

Abstract

Transferring policies learned in simulation via reinforcement learning (RL) to the real world is a challenging research problem in robotics. In this study, the sim2real transfer method of three papers is examined. In [2], the RL agent learns a robust policy by limiting the observation size and using domain randomization. The sim2real method in [11] learns an adaptive policy conditioned on a latent space that implicitly encodes the physics parameters of its environment. Samples must be collected on the robot to learn a latent space corresponding to the physics of the real world. In [6], the authors also employ a learned latent space, but constrain the mutual information between the latent variables and the input. This "information bottleneck" prevents the latent space from overfitting to the simulation physics parameters. Finally, I propose using the same information bottleneck approach on policy observations to learn a robust policy more effectively.

1. Introduction

Wheeled robots, which are relatively simple control, are mostly limited to flat ground or paved roads. Legged robots, on the other hand, have the ability to go wherever humans may go. These platforms allow embodied intelligence to be applied in a much wider variety of situations. For example, legs allows robots to go up and down stairs, which would be critical for a home robot in a two-story house. The focus of this paper will be on four-legged robots.

Unfortunately, controlling quadrupeds is a difficult challenge. Designing planning and control algorithms typically requires expert knowledge of multi-body dynamics and quadrupedal locomotion. Controllers may require extensive gain tuning and accurate contact modeling. Legged robots are hybrid dynamical systems, which means planners must optimize over continuous and discrete variables, which is computationally difficult [3, 10] .

Model-free reinforcement learning (RL) promises to be a way to learn locomotion skills without expert domain knowledge. However, these algorithms are generally sample inefficient. Fortunately, we can acquire samples quickly using physics simulators, such as Mujoco [9] or Pybullet [1]. Training in simulation also protects the robot, as reinforcement learning algorithms often take random or unpredictable actions during training. However, transferring policies trained in simulation to the real world is challenging.

Often, deep RL algorithms are tested on the same environment they are trained on. One example of such environment is the Atari games, a popular RL benchmark [5]. The state space of Atari games (raw pixels) is so high dimensional that it is impossible to sample every state, which means RL algorithms need to generalize across the state space after being trained on a finite number of samples. Fortunately, neural networks are good at doing this.

In sim2real transfer, the test environment is different from the training environment. If the locomotion problem is viewed as a partially-observable Markov Decision Process, the conditional observation probabilities and the state transition dynamics both shift when moving from sim to real. The following papers each employ a unique strategy handle this shift.

2. Literature Review

2.1. Policies Modulating Trajectory Generators [2]

The paper's title refers to the policy architecture designed by the authors; a neural network outputs the parameters for a gait generator instead of directly outputting robot joint targets. However, the details of this gait generator are not important to the sim2real transfer and are omitted for brevity. The policy architecture enables the RL agent to receive only 4D IMU data and still successfully learn a locomotion policy. This is in contrast to most learned policies which receive IMU data as well as the state of each robot joint. The small policy input size serves to "regularize" the agent and prevent it from overfitting to simulated environments.

The second strategy the authors use for sim2real transfer is domain randomization and perturbations. This makes the agent robust to a wide variety of physical robot parameters. The learned policy is directly transferable to the real robot. However, as shown in Fig. 1, robust policies sacrifice performance. [8]

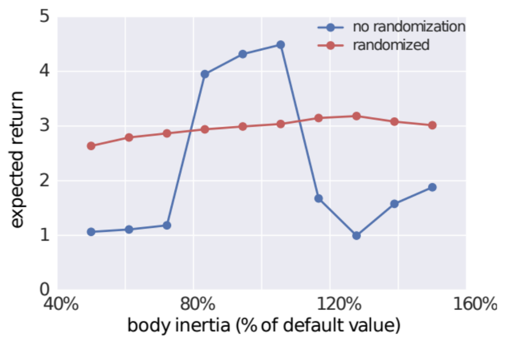

Figure 1. Comparison of a policy trained with body inertia randomization to one trained without. Using randomization prevents the agent from obtaining the optimal policy for any particular value of body inertia. [8]

2.2. Learning Fast Adaptation with Meta Strategy Optimization [11]

The sim2real strategy in this work is conditioning the policy on a low-dimensional latent space which implicitly encodes the physics of the environment. During training, many different simulated environments are generated with different physics parameters. The learning algorithm alternates between optimizing the environment-specific latent space and optimizing the policy. Both the latent space and the policy are optimized via Augmented Random Search [4].

When the policy is deployed, data from the real robot is collected in order to find a latent space that corresponds to the physics of the real world. This only takes about 4000 samples, or 75 seconds worth of data. Additionally, the learned latent space enables the policy to adapt to a wide variety of scenarios not seen during training, such as walking with a weakened motor or walking up a slope.

2.3. Learning Agile Locomotion Skills by Imitating Animals [6]

In this paper, the authors train a robot to mimic motion capture clips from quadrupedal animals. The training is done in simulation, and the authors use a latent space method to transfer the learned policies to the real robot.

In contrast to the previously discussed method, the latent space here has explicit access to simulation physics parameters. A two-layer encoder compresses physics parameters into a latent space during training. The encoder is regularized with a loss term that minimizes the mutual information between the simulation physics parameters and the latent space. Equation (1) contains the full objective function that the policy is trained with.

$$\begin{aligned}\arg\max_{\pi,E} \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})} \mathbb{E}_{\mathbf{z}\sim E(\mathbf{z}|\boldsymbol{\mu})} \mathbb{E}_{\tau\sim p(\tau|\pi,\boldsymbol{\mu},\mathbf{z})} &\left[\sum_{t=0}^{T-1}\gamma^t r_t\right] \\ &- \beta \mathbb{E}_{\boldsymbol{\mu}\sim p(\boldsymbol{\mu})}\left[\mathrm{D}_{\mathrm{KL}}[E(\cdot \mid \boldsymbol{\mu})\|\rho(\cdot)]\right]\end{aligned} \quad (1)$$

The benefit to this approach is that the amount of information coming from the physics parameters can be continuously tuned. When the hyperparameter β is zero, the policy degenerates to a purely robust method. When β is high, the policy may overfit to simulation parameters.

In the real world, the agent does not have access to ground-truth physics parameters. When the policy is deployed, a latent space is learned with Advantage Weighted Regression, a sample-efficient off-policy RL algorithm [7].

3. Conclusion and Future Work

The method from [11] uses a small latent space to minimize overfitting to the simulation physics parameters. The method in [6] avoids overfitting through a regularization term in the objective function. I argue that the latter method is more powerful because it allows a continuously-tunable tradeoff between robustness and adaptability. Additionally, the former method uses random search to find the latent space, which is computationally inefficient for high dimensional spaces. Therefore, the method in [11] may scale poorly to more complex tasks.

The first method discussed from [2] limits observation to 4D IMU data to prevent overfitting. However, this is a crude way to regularize the policy, as researchers must decide which parts of the state space to exclude and data from the quadruped's sensors goes unused. I propose using the information bottleneck from [6] on the observation space. This allows robustness to be continuously tuned and the policy to learn what to include and exclude.

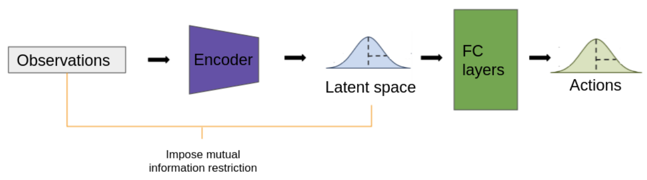

Figure 2. Proposed robust policy architecture. The mutual information constraint allows tunable robustness and enables the encoder to learn what to exclude or include.

References

[1] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[2] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke.

Policies modulating trajectory generators. In *Conference on Robot Learning*, pages 916–926. PMLR, 2018.

[3] Donghyun Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

[4] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[6] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[7] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[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. *arXiv preprint arXiv:1804.10332*, 2018.

[9] 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.

[10] Guiyang Xin, Wouter Wolfslag, Hsiu-Chin Lin, Carlo Tiseo, and Michael Mistry. An optimization-based locomotion controller for quadruped robots leveraging cartesian impedance control. *Frontiers in Robotics and AI*, 7:48, 2020.

[11] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2):2950–2957, 2020.