With the help of Abhimanyu

Terrain Adaptation of Hexapod Robot via Central Pattern Generator and Policy Gradient

Abstract:
Recently model-free deep reinforcement learning methods have shown promise in learning complex locomotion skills for legged robots. However, these approaches suffer from low sample efficiency due to a large action space. To address this problem we propose a solution that parameterizes a robot's gait using a Central Pattern Generator (CPG). By performing policy gradient on CPG parameters instead of joint velocities and positions, we reduce the size of our action space to contain strictly periodic behaviors. We demonstrate how our method is able to perform online gait adaptation on a real-world hexapod robot in rugged terrain. Furthermore, we present a novel approach for terrain representation which allows for a smoother sim-to-real transfer. By combining gait parameterization and simplified terrain representation with deep reinforcement learning, our robot learns a sample-efficient way to traverse unstructured terrain.

Keywords: CPG, Robotics, Reinforcement Learning

1 Introduction

Developing a motion controller for a legged robot to traverse unstructured terrain is challenging because of difficulty modeling system dynamics and a high dimensional action space. In recent works, policy gradient methods for Reinforcement Learning have had success in combating these challenges to learn complex control strategies on legged robots [1]. These methods learn an optimal policy by allowing the robot interacts with a simulated environment over many trials. They directly learn a state-to-action policy that minimizes a user specified cost function without any knowledge of the system dynamics. In our work we frame the problem of legged robot motion planning as a Partially Observable Markov Decision Process (POMDP) that we solve using a model-free RL algorithm called Proximal Policy optimization (PPO) [2]. We reward our legged-robot for making forward progress though an unstructured terrain in simulation.

Some of the major downsides to model-free approaches are their lack of sample-efficiency and difficulty transferring to real world. In order to increase the sample efficiency of our RL agent and reduce the size of our action space. We decide to search only in space of periodic policies. We do this by introducing a Central Pattern Generator as the low level motion planner for our robot. Central pattern generators (CPGs) are biological neural circuits that produce rhythmic outputs in the absence of rhythmic input [3]. Typically CPG models are used in robotics to design open-loop gaits for articulated robots, such as crawling, swimming or legged robots. We look to close the loop on our CPG model by using reinforcement learning to learn the optimal parameters of the robots gait shape based on some sensory feedback.

In addition to using the CPG to reduce the size of the action space, a bulk of our contribution also focuses on reducing the size of our observations space. In our work we explore several methods for lower dimensional terrain representation including classical height maps and a trained variational auto-encoder. In the full system, we use this module to simplify perception task for our agent. All of the work that we present in this paper is applied to a hexapod robot with 18 degrees of freedom. We show that by effectively reducing the size of both our action and observation spaces, we are able to

4th Conference on Robot Learning (CoRL 2020), Zurich, Switzerland.

increase sample efficiency. Furthermore, we are able to transition form simulation to the real world much more easily.

2 Related Work

Many different approaches for automatic gait optimization have been suggested to date, such as grid search and evolutionary algorithms [3]. Gait optimization is a basic, yet challenging problem for quadrupedal robots. Various automatic gait optimization methods have been used in locomotion to design gaits, including gradient descent methods [4] [1] , evolutionary algorithms [5] , particle swarm optimization [5], and many others [6] [7] [8] [9].

2.1 Hierarchical Reinforcement Learning

There is extensive research exploring hierarchical approaches to reinforcement learning. Sutton et al. [10] proposed *options*, or temporarily extended actions. This is an early formulation of hierarchical RL, combining low-level options with high-level policies learning to choose appropriate options. [11, 12] used a policy gradient approach to learn termination conditions of options. In [13], the low-level policy is trained with an auxiliary reward to encourage value changes of non-proprioceptive dimensions of state observation, such as translation toward a specific orientation for legged robots. The high-level policy, thus, does not need to access many low-level states including joint torques and velocities. While most early works focus on learning policies at different levels separately, end-to-end framework learning multiple levels of policy [14, 15] have also demonstrated feasibility recently. Our work is different from these methods because we use the CPG rather than a low-level controlling policy. This reduces the effort to train another policy and allows continuous transitions between options. Our high-level policy thus doesn't have to choose from a fast gait or a big gait, but can rather adapt to terrains by "defining" appropriate speed and height of the gait. The smooth CPG action space also ensures the smooth transition between gaits.

2.2 Locomotion for Legged Robots

Locomotion skills are fundamentally important for legged robots to traverse in various terrain environments. One line of work falls into the trajectory optimization domain. Hemker et al.[6] propose a motion optimization approach for humanoid robots. Specifically, they parametrize the walking trajectory and apply sequential surrogate optimization to solve for a fast and stable walking gait. Lizotte et al.[16] applies Bayesian optimization to a quadrupedal robot. They take into account two optimization criteria for a quadrupedal robot: one with respect to the maximum walking speed and the other for the maximum gait smoothness. Another line of research takes CPGs as an efficient approach. CPGs are proved to be capable of producing rhythmic movements such as swimming and walking based on environment information[17, 18]. In the context of legged robot, Alexander et al.[19] demonstrates success in applying open-loop CPGs to a quadruped robot for fast locomotion on tough terrain. However, this work has limitation in generalizing to different terrains, and adapting its parameters according to environment becomes crucial. Efforts have been made on designing closed-loop CPG using sensory feedback[20, 21]. In this work, rather than close-loop control, we propose a reinforcement learning network that learns to adapt CPG parameters based on the robot's proprioceptive and external states.

2.3 Terrain Representation

Visual data (2D image, height map) is a common approach to acquire information of surrounding terrain. Many prior works on RL involve learning dense representations from visual data, both 2D[22, 23] and 3D[24]. DeepLoco[25] has shown the effectiveness of using a terrain map feature in learning high-level tasks. They use 2D heightmaps with a resolution of 32x32 as input and apply convolutions to obtain a 128 dimension terrain representation. [26] explores various approaches of image latent representation and demonstrates that VAE is one of the most efficient and stable

representation for image-based RL. Therefore, in this work, we pretrained a VAE and apply it to encode the image to get a more compact representation of the terrain.

3 Method

3.1 Parameterized CPG Gait

CPGs can be represented mathematically as a system of coupled oscillators. The equations and code that I used for my simulated system were adapted from [27]'s work in the Biorobotics Lab at CMU. Using this model, each of the six legs of the hexapod can be modelled as an individual oscillator drawing out an ellipse in joint space. Each oscillator is governed by the following equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \omega n + (1 - H(x(t), y(t))t \tag{1}$$

$$H(x, y) = (\frac{x - c_x}{a})^2 + (\frac{y - c_y}{b})^2 - 1 \tag{2}$$

$$n = [-\frac{\partial H}{\partial y}, \frac{\partial H}{\partial x}]^T \; t = [\frac{\partial H}{\partial y}, \frac{\partial H}{\partial x}]^T.$$

In these equations, H is the limit cycle function, a and b are the major and minor axis of the limit cycle ellipse, c_x and c_y represent the center of the limit cycle, and w is the desired angular frequency. The first term in the equation 1 is pushing the leg along an elliptical cycle, while the second term is pulling the leg back onto the limit cycle if it gets off. By changing the values of a and b (the major and minor axis of our joint space ellipse) as well as w (the desired angular frequency) we were able to shape our gait to adapt to different terrains.

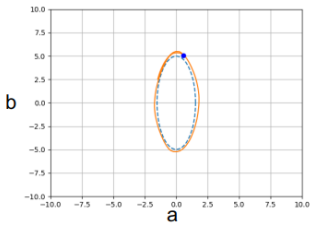

Figure 1: Plotting the Limit Cycle Shape for One Foot

3.2 Proximal Policy Optimization

We describe the problem of selecting gait parameters for our CPG model as a POMDP in which the state of our robot is some estimate of where it is in the environment and a corresponding rgb camera image. The actions our agent can take are modifications to the parameters of our CPG gait, and we specify that the reward for our agent is quick forward progress in unstructured terrain. We use reinforcement learning to learn the optimal policy for our agent by simulating thousands of trials in a simulation environment called py-bullet.

Policy gradient methods directly optimize a state-action policy instead of learning a value function. These methods are preferred in the field of 3D locomotion because it is often intractable to learn a value function. Recently Proximal Policy Optimization (PPO) has emerged as a new class of popular model-free reinforcement learning algorithms that performs comparably or better than state-of-the-art approaches [28]. PPO has shown popularity specifically because of its ease to implement and tune in practice.

Typically vanilla policy gradient methods have been shown to have a convergence problem that can be addressed by the natural policy gradient. In practice implementing natural policy gradient is difficult because it requires a second order derivative based optimization. PPO is unique because is

able to approximate this second order with a simple first order gradient decent scheme and a soft constraint on its objective function. Because of this simplicity, PPO has shown promise on a variety of challenging tasks even with relatively limited computational resources. In our work, we use the stable-baselines implementation of PPO relatively like a black box to solve our formulated POMDP.

3.3 Terrain Representation

Dealing with high dimensional states in the form of images is not a sample efficient way to do reinforcement learning. We effectively reduce the dimensionality of the state using representation learning. We train an encoder-decoder architecture use the encoder to generate latent variables which can be passed to our RL agent. Specifically, we choose to use a Variational Autoencoder (VAE) because it has been shown to result in disentangled factors in the latent space. During training we try to minimize the following objective:

$$-D_{KL} = (q_\phi(Z|X_i))||p_\theta(z) + E_{q\theta(z,xi)}[logp_\theta(x_i|z)] \tag{3}$$

In this equation, DKL represents the Kullback-Leibler (KL) Divergence and Eq(z—x)[logp(x—z)] represents the reconstruction loss. We chose to pretrain our VAE on RGB images collected from an RGBD sensor while our robot is walking with a random policy in simulation. By pretraining our VAE and freezing its weights during RL policy optimization we hope to introduce a more stable training.

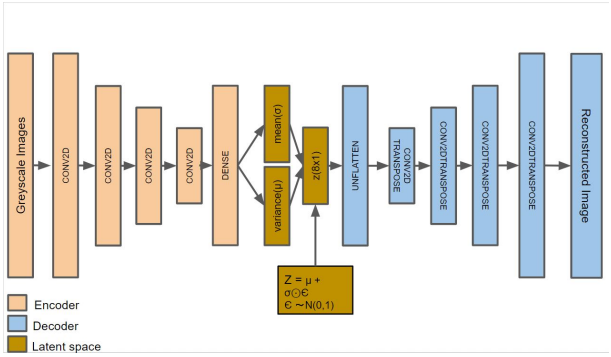

Figure 2: VAE Pipeline Architecture

The architecture of our VAE is shown above. We reduce the dimension of our input image from (220x320) matrix to an 8 dimensional embedding. This latent vector is then fed to our RL agent along with other state information.

3.4 Putting it All Together

When put together, our pipeline is able pre-process raw sensor data and learn to output optimal changes to CPG gait parameters as policy actions. Our agent's state vector is comprised of position, orientation, velocity, angular velocity, current cpg parameters (a,b,w), and an 8 dimensional image embedding.

In order to achieve more stable transitions between gaits, our trained policy outputs the second derivative of the desired cpg parameters. We use these actions to smoothly increment 'a' the major axis of limit cycle, 'b' the minor axis of the limit cycle, and 'ω' the angular velocity of the limit cycle. All values are also bounded by a manually set upper and lower bounds to avoid infeasible gaits. The reward function for our agent is simply formulated as $r_t = 100 * v_{tx}$, and our episode will end after the robot moves 3m from its origin or it gets stuck.

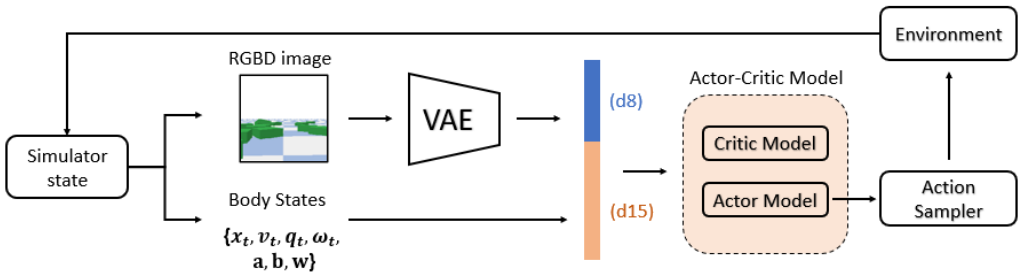

Figure 3: Overview of RL Pipeline

4 Results

All of our experiments were done in simulation, using the open-source pybullet environment [29]. First, we show the results for training our VAE to generate terrain embeddings. After that, we show the results for training our full reinforcement learning pipeline on a simulated hexapod for one million timesteps. Finally, we also show some of the limitations of our method by showing the outcome of running our trained policy on unseen environments.

4.1 Training our VAE

We trained our VAE implementation on a data set of around 2.5k images collected from the robot's point of view. The test results of the trained VAE are shown below in the left image. Within this image we are able to see that the input image and reconstructed image are almost indistinguishable. The loss, which is a combination of reconstruction loss and the KL divergence loss, is 20.2. The image on the right shows how images look in the embedding space. Since there are clear decision boundaries between images with varying amounts of blocks in this space, we can further confirm that our latent space appropriately represents our terrain.

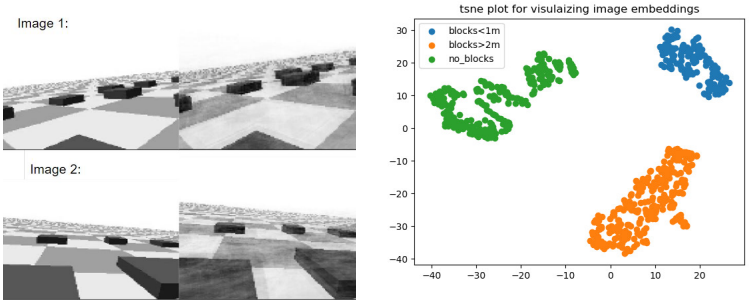

Figure 4: Left: Input/Reconstructed, Images Right: Feature Space

4.2 RL Training Case

We trained our robot to make the quickest forward progress in a simple scenario with only one obstacle in front of it.

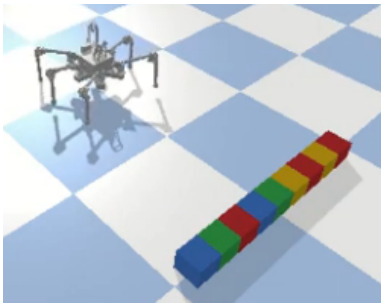

Figure 5: Training Scenario. Video: https://www.youtube.com/watch?v=RvvIhSrE5YA

We found that our agent was able to learn an optimal policy for navigating over this obstacle within the first 250,000 timesteps. Below we show a plot of our reward signal over the full 1,000,000 timestep training process. In the plot the reward signal starts off unstable, sometimes making it over the blocks and sometimes failing. In the next section of the plot the policy reaches a local optimum, where the robot learns to take large,fast steps toward the obstacle. Finally, a quater of the way into the training process, the robot learns to combine these fast steps with high steps when it sees an obstacle to achieve the maximum reward.

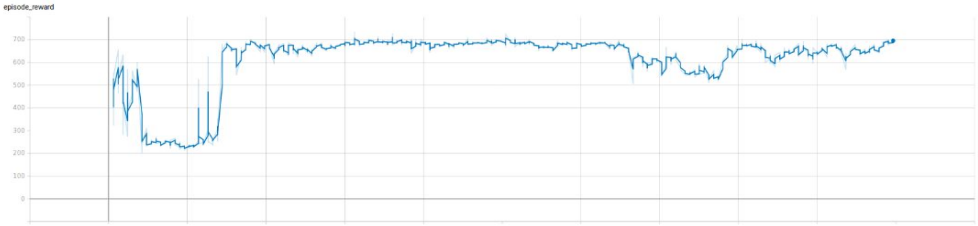

Figure 6: Episode Rewards vs. Time

Below we show how the cpg gait paramters adapt over the course of the robot's journey in this simple environment. In the leftmost image the robot has an initial set of cpg paramters that are sub optimal. In the next image the policy shapes the parameters by increasing the length of the ellipse major axis 'a parameter' to take larges steps close to the ground. Once the robot comes close to the obstacle, in the last image the agents policy increases the ellipse minor axis 'b parameter' to take a higher step and overcome the obstacle.

4.3 Generalizing to Unseen Environments

To verify that our agent was actually using information from the images to make informed actions, we decided to change the location and size of blocks in our environment. In one setting we added an additional set of blocks that the robot had never seen before. The robot was able to adapt to the unseen set of blocks successfully by lifting its legs when it approached the block.

In a second setting we decided to make the block we were trying to overcome wider. Unfortunately, in this setting the robot got stuck when trying to traverse over the block. This could be due to physical constraints, but it is more likely due to an inability of our policy to generalize to this unseen environment. We believe that we may, however, be able to overcome these kind of unique test environments by introducing some domain randomization during our RL training process.

5 Conclusions and Future Work

We successfully implemented a deep reinforcement learning pipeline that uses a CPG for low level controls and modifies parameters of the CPG to adapt a robot's gait to an environment. Additionally,

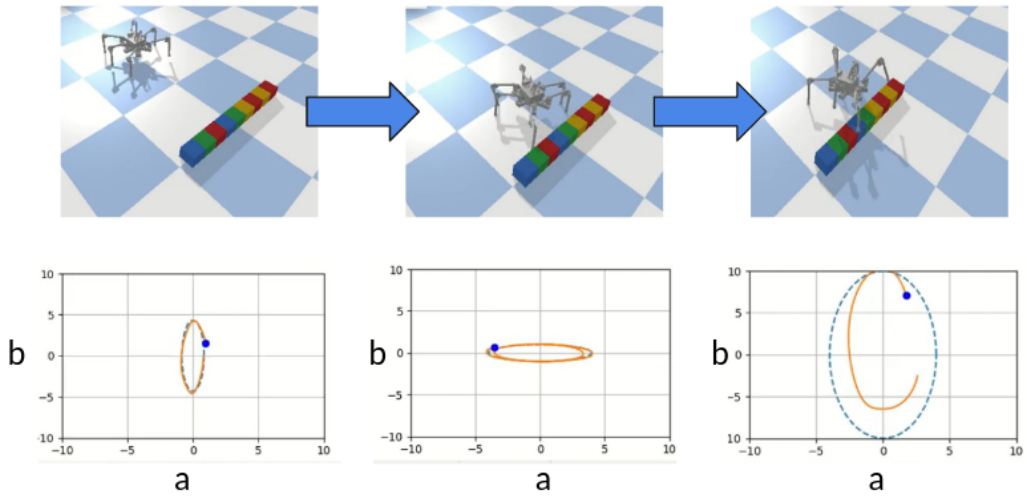

Figure 7: Trained Policy

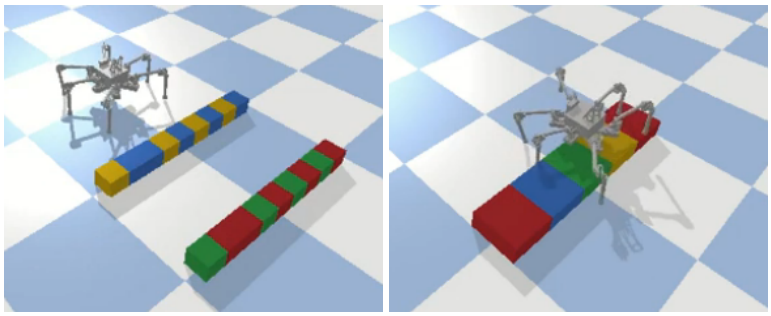

Figure 8: Left: adding additional blocks, Right: making block wider

we were able to learn how to encode sensory information into a lower dimensional latent space by training a VAE. We hope that by introducing this structure within model-free learning, our agent will be able to learn more sample efficiently and be able to transfer to real world more easily.

Our method is limited because it is not able to generalize well to unseen environments. Our robot also gets caught in local minimum when training and requires tuning of hyperparamters to achieve optimal performance. In the future we wish to get rid of some these problems by training in randomized domains so that are robot learns general policies. We have implemented a contious terrain enviornenent that we are looking to train our robot on in the near future. Furthermore, we wish to increase the amount of learnable parameters in our CPG formulation so that the robot can execute more complex locomotive behavior.

Finally, we want eventually transfer our learned policy to a real hexapod robot. We believe that because we are using a stable CPG controller for our low level commands this should be a bit easier. The system that we are looking to transfer to can be seen below:

We believe that our perception pipeline will transfer more easily to real world if we further process our images before being fed into our VAE. To this end, we propose using height maps instead of RGB images in the hope for better sim-to-real transfer.

Figure 9: Hebi Robotics Daisy Robot

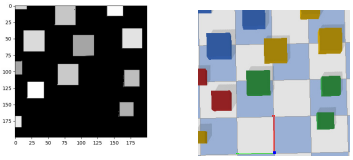

Figure 10: Height Maps from Simulated World

References

[1] A. Kumar, N. Paul, and S. Omkar. Bipedal walking robot using deep deterministic policy gradient. *arXiv preprint arXiv:1807.05924*, 2018.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[3] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 76(1):5–23, 2016. ISSN 1573-7470. doi:10.1007/s10472-015-9463-9.

[4] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2849–2854. IEEE, 2004.

[5] S. Chernova and M. Veloso. An evolutionary approach to gait learning for four-legged robots. pages 2562 – 2567 vol.3, 01 2004. ISBN 0-7803-8463-6. doi:10.1109/IROS.2004.1389794.

[6] S. M. v. S. O. S. H. Hemker, T. Efficient walking speed optimization of a humanoid robot. *International Journal of Robotics Research (IJRR)*, pages 303–314, 02 2009.

[7] T. Geng, B. Porr, and F. Wörgötter. Fast biped walking with a sensor-driven neuronal controller and real-time online learning. *I. J. Robotic Res.*, 25:243–259, 01 2006.

[8] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[9] K. Yamane. Geometry and biomechanics for locomotion synthesis and control. In *Modeling, Simulation and Optimization of Bipedal Walking*, pages 273–287. Springer, 2013.

[10] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[11] G. Comanici and D. Precup. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 709–714, 2010.

[12] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[13] K. Marino, A. Gupta, R. Fergus, and A. Szlam. Hierarchical rl using an ensemble of proprioceptive periodic policies. 2018.

[14] K. Y. Levy and N. Shimkin. Unified inter and intra options learning using policy gradient methods. In *European Workshop on Reinforcement Learning*, pages 153–164. Springer, 2011.

[15] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[16] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with gaussian process regression. pages 944–949, 01 2007.

[17] M. MacKay-Lyons. Central pattern generation of locomotion: a review of the evidence. *Physical therapy*, 82(1):69–83, 2002.

[18] M. R. Dimitrijevic, Y. Gerasimenko, and M. M. Pinter. Evidence for a spinal central pattern generator in humans a. *Annals of the New York Academy of Sciences*, 860(1):360–376, 1998.

[19] A. Sproewitz, L. Kuechler, A. Tuleu, M. Ajallooeian, M. D'Haene, R. Moeckel, and A. Ijspeert. Oncilla robot: a light-weight bio-inspired quadruped robot for fast locomotion in rough terrain. In *5th International Symposium on Adaptive Motion of Animals and Machines*, number CONF, 2011.

[20] L. Righetti and A. J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *2008 IEEE International Conference on Robotics and Automation*, pages 819–824. IEEE, 2008.

[21] G. Sartoretti, S. Shaw, K. Lam, N. Fan, M. Travers, and H. Choset. Central pattern generator with inertial feedback for stable locomotion and climbing in unstructured terrain. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–5. IEEE, 2018.

[22] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016.

[23] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

[24] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013.

[25] X. B. Peng, G. Berseth, K. Yin, and M. van de Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, 36(4), 2017.

[26] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[27] G. Sartoretti, S. Shaw, K. Lam, N. Fan, M. Travers, and H. Choset. Central pattern generator with inertial feedback for stable locomotion and climbing in unstructured terrain. *International Conference on Robotics and Automation (ICRA)*, 01 2018.

[28] P. Hämäläinen, A. Babadi, X. Ma, and J. Lehtinen. PPO-CMA: proximal policy optimization with covariance matrix adaptation. *CoRR*, abs/1810.02541, 2018.

[29] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation in robotics, games and machine learning, 2017.