Learning Visual-Locomotion Policies that Generalize to Diverse Environments

Alejandro Escontrela[†§*], **George Yu**[§], **Peng Xu**[§], **Atil Iscen**[§], **Jie Tan**[§]

Georgia Institute of Technology[†], Robotics at Google[§]

aescontrela@gatech.edu, {georgeyu, pengxu, atil, jietan}@google.com

Abstract

We address the problem of learning locomotion controllers which generalize to a diverse collection of terrains that are often found in the real world. Prior works optimize policies to succeed in a single type of terrain with limited variations and depend on observations gathered from the ground-truth simulation state (i.e., ground-truth height maps, which are difficult to obtain in practice and noisy). We frame the challenge of learning generalizable locomotion controllers as a multi-task reinforcement learning problem and define each task as a type of terrain that the agent must traverse. We then propose an approach that trains an end-to-end visual locomotion policy to achieve high generalization performance. Our policies learn to navigate over many terrains that a legged robot may encounter in the real world, including stairs, rugged land, obstacles, office environments, rooms with humans, all without requiring manual preprocessing of vision inputs.

(a) Dynamic Environment (b) Office 1 (c) Office 2 (d) Mountainous

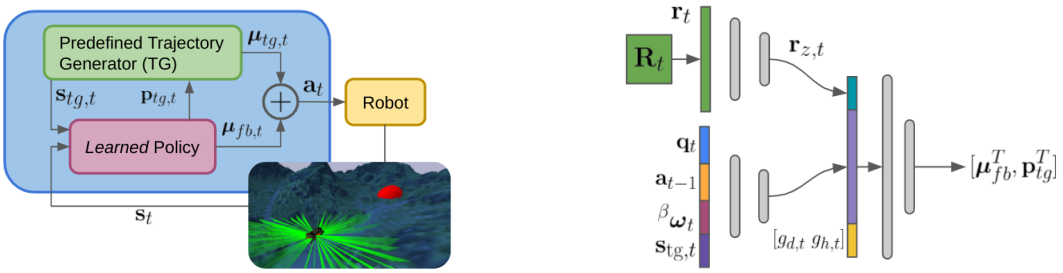
Figure 1: A Laikago robot navigating a variety of complex terrains not encountered during training.

1 Introduction

The ability to traverse unstructured terrains makes legged robots an appealing solution to a wide variety of tasks, including disaster relief, industrial inspection, and planetary exploration (1; 2). However, to successfully deploy robots in these settings, we must design controllers that work well across many different terrains. To this end, deep Reinforcement Learning (RL) (14) has proven itself capable of learning control policies that generate agile and robust behaviors which generalize to environments not encountered during training (6; 8). Additionally, recent work in multi-task reinforcement learning (MTRL) (3) has been successful in training policies which generalize to a wide variety of tasks. In (7), the proposed method learns a single policy which achieves state-of-the-art performance on 57 Atari games. (18) evaluated the performance of various RL algorithms on a grasping and manipulation benchmark and demonstrated that a single control policy is capable of successfully completing a variety of complex robotic manipulation tasks.

[*]Work performed during an internship at Google Brain.

NeurIPS 2020 3rd Robot Learning Workshop: Grounding Machine Learning Development in the Real World.



(a) Our Architecture (b) Learned Policy Component

Figure 2: Overview the policy architecture. a) Outputs $\boldsymbol{u}_{tg}$ of the TG are combined with the outputs of the learned policy network $\boldsymbol{u}_{fb}$ to produce the final action $\mathbf{a}_t$. b) The learned policy reads the state of the TG $\mathbf{s}_{tg}$ and the agent state $\mathbf{s}_t$ to produce the new TG params $\mathbf{p}_{tg}$ and residual action terms $\boldsymbol{\mu}_{fb}$.

For legged locomotion, many prior works (11; 15; 6) optimize policies to succeed in a single type of terrain with limited variations, thus limiting the agent's ability to generalize to many of the terrains the robot would encounter in the real world. Additionally, many of these works were evaluated in simulation, relying on unrealistic observations that are acquired from the ground-truth simulation state, such as height maps (12). In practice, these observations are difficult to obtain and noisy.

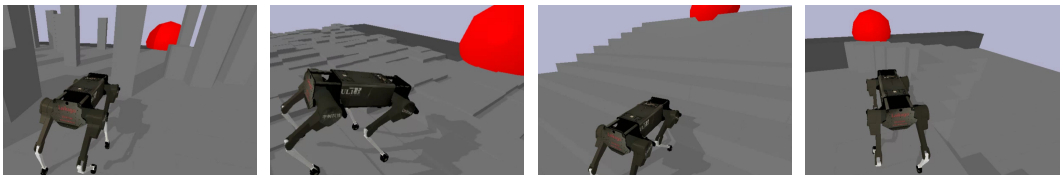
In this paper, we frame the challenge of learning generalizable locomotion controllers as a multi-task reinforcement learning problem and show that the agent learns a robust policy that works well across a wide variety of terrains (tasks). Using Policies Modulating Trajectory Generators (9), which consists of a trajectory generator and a learned policy network, we incorporate prior knowledge into our policies resulting in smoother leg movements. We then use Proximal Policy Optimization (13) to train the end-to-end policy that maps directly from the robot's proprioceptive and vision inputs to control actions. We demonstrate that this parameterization yields better generalization performance than a purely reactive policy. We also present a novel and efficient procedural 3D locomotion task generation technique. As a result, we learn a single end-to-end visual locomotion policy that learns to traverse more than 16 types of terrains. Additionally, the policy relies solely on sensors found on-board the robot and does not require any manual preprocessing of the sensor observations (i.e., to generate height maps). We evaluate the performance of our policy and competing approaches using a high-fidelity simulator (4) and virtual environments gathered from the real-world (16).

2 Methods

2.1 Problem Statement

In this work, we frame legged locomotion as a multi-task reinforcement learning problem (MTRL) and define each task as a type of terrain that the agent must traverse. Let us define a distribution of tasks, where each task is defined as a Markov Decision Process (MDP) $M_i \in \mathcal{M}$. Each MDP is defined as a 4-tuple, $M_i = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}_i, \mathcal{R} \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}_i : \mathcal{S} \times \mathcal{A} \times S \to \mathbb{R}_+$ is the transition probability function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. We utilize a common state space $\mathcal{S}$ and action space $\mathcal{A}$ such that we can use the same stochastic policy $\pi_\theta : \mathcal{S} \times A \to \mathbb{R}_+$ for all tasks. We employ the simple, yet flexible task of navigating the robot to a desired location, represented as a 3D vector $\mathbf{g} = (x_g, y_g, z_g)$ (see Figure 3). Once the agent's center of mass is within a sphere of radius r_g from the target location, the task is considered complete. We define the following reward function: $r_t^c = (d_t - d_{t-1})/\Delta t$, where d_t is the distance from the agent to the target location at timestep t, d_{t-1} is the distance to the target at timestep $t-1$, and Δt is the timestep duration. This reward can be interpreted as a finite differences approximation of the agent's velocity toward the target.

We use a simulated LiDAR sensor to provide the agent with information of the surrounding terrain (See Figure 2(a)). The 3D depth scan $\mathbf{R}$ is normalized to the range $[0, 1]$ and flattened to $\mathbf{r}$. Additionally, we make use of a simulated IMU sensor to gather the robot's angular rates ${}^{\beta}\boldsymbol{\omega} = (\dot{\theta}, \dot{\phi}, \dot{\psi})$ and motor encoders to observe the robot's 12 joint angles, $\mathbf{q}$. We now define the state for each timestep t as $\mathbf{s}_t = [\mathbf{r}_t^T, g_{d,t}, g_{h,t}, {}^{\beta}\boldsymbol{\omega}_t^T, \mathbf{q}_t^T, \mathbf{a}_{t-1}^T]$, where g_d and g_h are the distance and relative heading to the



(a) Obstacles (b) Rugged (c) Stairs (d) Cliff

Figure 3: A Laikago robot deployed in various procedurally generated training environments.

target, and $\mathbf{a}_{t-1}$ is the commanded action for the previous timestep. The action $\mathbf{a}_t$ from the policy specifies the target joint angles, which are tracked by PD controllers.

The goal is then to learn the parameters of a single stochastic policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ which maximizes the average expected return across all terrains from a terrain distribution $p(\mathcal{M})$, given by $\mathbb{E}_{M\sim p(\mathcal{M})}[\mathbb{E}_\pi[\Sigma_{t=0}^{T}\gamma^t\mathcal{R}_t(\mathbf{s}_t,\mathbf{a}_t)]]$. Prior work in MTRL requires that an encoding of the task ID be provided as an input to the policy (18; 17). However, in our work we omit this additional variable, as our goal is to produce a policy which generalizes to terrains beyond those encountered in training.

2.2 Visual Locomotion Policy Architecture

We now introduce the policy architecture we used to generate smooth, realistic behaviors that help the agent generalize to different terrains. Policies Modulating Trajectory Generators (PMTG) (9) is a policy parameterization which consists of a trajectory generator (TG) for the legs that can be modulated by a learned control policy $\pi_\theta(\cdot)$. The learned policy observes the state of the TG, $\mathbf{s}_{tg}$, and the agent's state $\mathbf{s}_t$. The policy then outputs parameters of the TG, $\mathbf{p}_{tg}$, such as gait frequency, swing height, and stride length. The learned policy also outputs residual action terms $\boldsymbol{\mu}_{fb}$ which are combined with the TG actions $\boldsymbol{\mu}_{tg}$ to produce the final action for the agent to execute $\mathbf{a}_t = \boldsymbol{\mu}_{tg} + \boldsymbol{\mu}_{fb}$. We adapt this policy architecture to our work via two modifications. First, we augment the observation space with vision, which we obtain from a simulated 3D LiDAR sensor. Second, we modify the policy to solve navigation problems by providing the distance and heading to the target, g_d, g_h. See Figure 2 for a block diagram representation of the policy architecture. We use two encoders to process the proprioceptive and exteroceptive inputs before passing them into the policy. As detailed in (6), this architecture achieves a separation of concerns between the basic locomotion skills and terrain perception and navigation. We train our policy using a distributed version of the Proximal Policy Optimization (PPO) (13) online RL algorithm, implemented using TF-Agents (5).

2.3 Terrain Parameterization and Procedural Task Generation

To train generalizable policies, we develop a procedural terrain generator that can create a large variety of 3D locomotion tasks efficiently. The environment is composed of $m \times n$ pillars, each pillar having cross-sectional dimensions of l, w. During training, a task is selected and the heights of each pillar are adjusted to reflect the chosen task. For the obstacles task (Figure 3(a)), a subset of the pillars is chosen at random and their heights adjusted to introduce barriers which the agent must navigate around. To generate a rugged terrain (Figure 3(b)), the height of all pillars is chosen at random from a uniform distribution. We have found that this terrain parameterization allows us to generate over 10 different types of 3D Locomotion tasks which an agent may encounter in the real world. This provides the agent with a rich set of training data from which to learn.

3 Results and Discussion

The Impact of MTRL on Generalization Table 1 shows the generalization performance of our visual-locomotion policy trained on different types of terrains (rows) to terrains not encountered during training (columns), including a maze (Maze), a steep and rugged mountain (Mountainous), two indoor scenarios (Office 1 and Office 2) generated using the Gibson simulator (16), an office space with moving humans (Dynamic Env), a forest scenario with rugged terrain and obstacles (Forest), a winding cliff (Cliff), and a randomly-generated continuous mesh (Continuous). Figure 1 shows a subset of these testing environments. We evaluate multiple policies using the Task Completion

Table 1: The following table demonstrates the generalization performance of our visual-locomotion policy. Our policy was trained in a MT using prodecural 3D locomotion task generation. The average t_{cr} and standard deviation were computed over 20 episodes for the top five policies of each method.

Train Description	**Evaluation Environment Task Completion Rate**							
	Maze	Mount -ain	Office 1	Office 2	Dynamic Env	Forest	Cliff	Contin -uous
Flat	0.39 ±0.17	0.22 ±0.21	0.54 ±0.12	0.51 ±0.16	0.50 ±0.12	0.43 ±0.13	0.32 ±0.18	0.57 ±0.12
Rugged	0.57 ±0.18	0.28 ±0.11	0.64 ±0.12	0.60 ±0.15	0.59 ±0.16	0.65 ±0.18	0.36 ±0.16	0.73 ±0.17
Holes	0.33 ±0.18	0.22 ±0.12	0.41 ±0.15	0.38 ±0.16	0.38 ±0.13	0.39 ±0.12	0.52 ±0.15	0.57 ±0.13
Obstacles	0.70 ±0.19	0.23 ±0.04	0.67 ±0.19	0.69 ±0.15	0.68 ±0.17	0.59 ±0.09	0.55 ±0.13	0.53 ±0.16
Stairs	0.39 ±0.18	0.28 ±0.11	0.52 ±0.14	0.49 ±0.13	0.43 ±0.16	0.60 ±0.19	0.38 ±0.18	0.68 ±0.10
Ours	**0.72** ±0.13	**0.67** ±0.12	**0.84** ±0.11	**0.83** ±0.13	**0.79** ±0.14	**0.78** ±0.10	**0.70** ±0.17	**0.72** ±0.19

Rate t_{cr}, which measures how close the agent gets to the target relative to its starting position. The first five rows of this table correspond to policies trained on a single terrain. They achieve low generalization performance due to a lack of diverse training data. Meanwhile, our approach achieves much higher generalization performance. For instance, our method on average achieves a t_{cr} of 67% on the mountain task, while policies trained in a single type of terrain only achieve 28% at best. In fact, our approach achieves significantly better generalization performance across all evaluation tasks.

Ablation Studies We compare our method to a reactive policy (i.e., no TG), a PMTG policy without vision input, and a PMTG policy trained in a sequential fashion. Table 2 demonstrates the impact of each design decision on the resulting generalization performance of the policy. Our policy generalizes better than a purely reactive policy. This demonstrates that providing the policy with a strong prior over the space of possible gaits not only leads to smoother actions, but also results in improved generalization. The comparison with a policy that lacks vision makes it clear that our 3D LiDAR sensor plays an essential role in learning locomotion policies which can adapt to wide variety of terrains. We also compare our approach to a sequential training regime, whereby the policy is trained to complete each task in succession. The policy trained in a sequential fashion performs poorly due to *catastrophic forgetting* (10)

Our preliminary results on a suite of simulated environments show that treating legged locomotion as a MTRL problem leads to increased generalization performance. In future work, we plan to evaluate our work on a real-world robotic system and further increase generalization performance.

Table 2: This table compares our proposed method to other policies deployed in a MTRL training regime. The performance decreases when different components are removed from the system.

Train Description	**Evaluation Environment Task Completion Rate**							
	Maze	Mount -ain	Office 1	Office 2	Dynamic Env	Forest	Cliff	Contin -uous
Reactive	0.67 ±0.13	0.65 ±0.16	0.76 ±0.16	0.77 ±0.19	0.56 ±0.12	0.75 ±0.14	0.68 ±0.10	0.64 ±0.14
No Vision	0.56 ±0.17	0.44 ±0.12	0.54 ±0.16	0.51 ±0.15	0.47 ±0.17	0.57 ±0.16	0.32 ±0.08	0.54 ±0.13
Sequential	0.60 ±0.18	0.29 ±0.15	0.63 ±0.17	0.62 ±0.14	0.59 ±0.19	0.61 ±0.13	0.57 ±0.14	0.61 ±0.17
Ours	**0.72** ±0.13	**0.67** ±0.12	**0.84** ±0.11	**0.83** ±0.13	**0.79** ±0.14	**0.78** ±0.10	**0.70** ±0.17	**0.72** ±0.19

References

[1] K. Albee, A. C. Hernandez, O. Jia-Richards, and A. T. Espinoza. Real-time motion planning in unknown environments for legged robotic planetary exploration. In *2020 IEEE Aerospace Conference*, pages 1–9, 2020.

[2] C. Dario Bellicoso, Marko Bjelonic, Lorenz Wellhausen, Kai Holtmann, Fabian Günther, Marco Tranzatto, Péter Fankhauser, and Marco Hutter. Advances in real-world applications for legged robots. *Journal of Field Robotics*, 35(8):1311–1326, 2018. doi: 10.1002/rob.21839. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21839`.

[3] Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, ICML'93, page 41–48, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1558603077.

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

[5] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`, 2018. URL `https://github.com/tensorflow/agents`. [Online; accessed 25-June-2019].

[6] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments, 2017.

[7] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart, 2018.

[8] 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): eaau5872, Jan 2019. ISSN 2470-9476. doi: 10.1126/scirobotics.aau5872. URL `http://dx.doi.org/10.1126/scirobotics.aau5872`.

[9] Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators. volume 87 of *Proceedings of Machine Learning Research*, pages 916–926. PMLR, 29–31 Oct 2018. URL `http://proceedings.mlr.press/v87/iscen18a.html`.

[10] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C): 109–165, January 1989. ISSN 0079-7421. doi: 10.1016/S0079-7421(08)60536-8.

[11] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, 36(4), 2017.

[12] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL `http://doi.acm.org/10.1145/3197517.3201311`.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[14] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[15] Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning, 2020.

[16] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[17] Tianhe Yu, Saurabh Jumar, Abhishek Gupta, Sergey Levine, Karol Hausmann, and Chelasea Finn. Multi-task reinforcement learning without interference, 2019.

[18] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019.