Hierarchical Learning for Quadruped Locomotion and Path Planning

Edwin Lai | Nicole Nadim | Shengkai Peng | Weiqi Wang

Abstract

We are interested in applying hierarchical reinforcement learning to the legged robotic locomotion control framework described in a paper by Jain [1] . Here we implement multi-level policies to deconstruct complex goals and locomotion tasks into more primitive ones that can be applied to transfer learning.

We train a 12 DOF quadruped robot to walk and steer through a path to reach a goal, and show that the hierarchical policy is able to separate the goal reaching tasks into high and low level control tasks

Introduction

Designing controllers for legged robots is a tedious and challenging task, requiring fast communication protocols, precise calibration and careful tuning of control parameters. Additionally, traversing different environments requires significant changes in control methods. Despite the complexity of the problem, the highly coupled tasks of locomotion can easily be broken down in a hierarchical fashion: the high level decision making process examines the surrounding and sends the lower level controller more specific commands.

We designed a hierarchical policy which observes environment states at two levels and outputs motor control commands to the robot. We train this network to maximize a single reward function in order to generate high level controllable locomotion.

Goals

- Validate framework for separating complex locomotion and path planning
- Simultaneous training of general locomotion control and goal reaching
- Learn transferable low level controller with use of high level latent space

Environment Setup

Our environment is modified from the Open AI gym environment used in the open sourced project for imitation learning by Peng. et. at. [2] and simulates the inherited quadruped robot Laikago using pybullet3. We designed a trajectory generator to parameterize leg trajectories and overall gait of Laikago, which combined with the policy, creates a policies modulating trajectory generator (PMTG) [3] outputting target leg poses.

State observations include: position, roll, roll rate, pitch, pitch rate, yaw and motor angles of the robot for the 3 most recent steps.

The reward is designed to:

- Prioritizes improvement to goal at each time step and body stability
- Penalizes significant changes in policy and joint angle corrections

An episode ends when the goal is reached or if the robot leaves the path or falls over.

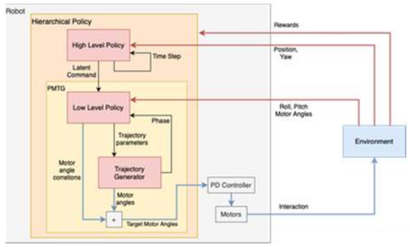
$$R_{Goal} = ||P_{goal} - P_{robot,t-1}||_2 - ||P_{goal} - P_{robot,t}||_2$$

$$R_{Stability} = w_1 * e^{-|r|} + w_2 * e^{-|p|} + w_3 * e^{-|\delta r|} + w_4 * e^{-|\delta p|}$$

$$R_{Offset} = w_5 * e^{-|C_t|} + w_6 * e^{-|C_t - C_{t-1}|}$$

$$R_{Parameters} = w_7 * e^{-|Pa_t|} + w_8 * e^{-|Pa_t - Pa_{t-1}|}$$

Policy Network



Hierarchical Network Structure:

- High level policy outputs a latent command while modulating its own command frequency (120ms to 1.2s between each command)
  - Linear policy weights: **3 x 12**
- Low level policy modulates TG and outputs correction angles at 6ms.
  - Linear policy w/bias weights: **48 x (88 + 1)**
- Total number of policy parameters: **4308**

Network structure for baseline model:

- **512 x 256** dense layer for policy,
- **512 x 256** dense layers for value function, Both used ReLu activation.

Constrained motor angle corrections to 0.2 Radian, allowed to modulate all TG parameters.

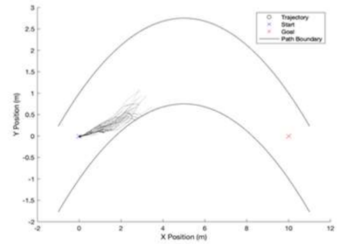
Policy Gradient Methods

Proximal Policy Optimization on PMTG

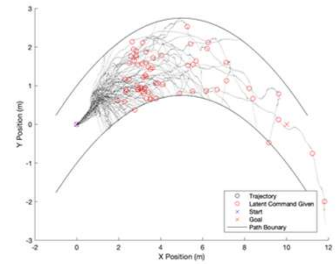
As a baseline to test the validity of the environment, we constructed a non-hierarchical network with the goal of moving forward at a steady velocity and trained it using proximal policy optimization (PPO)[4]. In this policy, the reward is set as the error of the robot's center of mass in following a straight line.

We also trained it with the full reward to follow the path and found that it had trouble navigating the turn.

Baseline Result - Path Trajectories

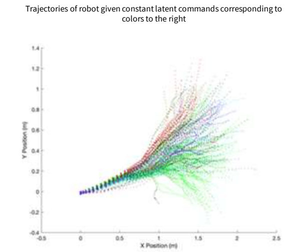


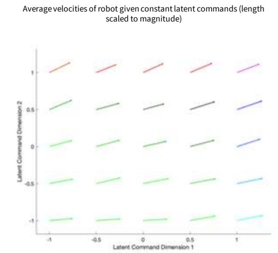
Hierarchical Result - Path Trajectories



Augmented Random Search on Hierarchical Policy

The hierarchical policy is trained using augmented random search (ARS)[5], a derivative-free gradient method, for 150 iterations. It learns a stable galloping gate that is able to navigate around the bend of the path. Sweeping through the latent command space, we see that the high level policy can control the curvature of the robot's trajectory while maintaining the same locomotion gait.





- The hierarchical structure successfully separates the goal reaching task into locomotion and steering for the low and high level policies respectively.
- The high level policy also learns when to apply a new latent command to steer the robot as it approaches the curve of the path. It continues to send new commands as the robot gets closer or deviates from the goal.
- With a more balanced path requiring a combination of tight and wide, left and right turns, we predict that the latent command space would reflect broader locomotion steering.

Conclusion

- Our hierarchical policy trained with ARS is able to learn locomotion and navigate the curved path simultaneously with a single reward function
- The high level control can steer the robot while low level control maintains stability
- Path shape is important to defining the latent space commands
- Low level policy may be transferable by simply training high level network to apply latent commands at learned points along new path
- The multi-level policy can make use of different state observations to make control decisions at different time scales depending on the task level

References

[1] Deepali Jain, Atil Iscen, and Ken Caluwaerts. "Hierarchical Reinforcement Learning for Quadruped Locomotion". In: (2019). arXiv: 1905.08926 [cs.LG].

[2] Xue Bin Peng et al. "Learning Agile Robotic Locomotion Skills by Imitating Animals". In: (2020). arXiv: 2004.00784 [cs.RO].

[3]. Atil Iscen et al.Policies Modulating Trajectory Generators. 2019. arXiv:1910.02812 [cs.RO].

[4] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

[5] Horia Mania, Aurelia Guy, and Benjamin Recht. "Simple random search provides a competitive approach to reinforcement learning". In: *CoRR* abs/1803.07055 (2018). arXiv: 1803.07055. URL: http://arxiv.org/abs/1803.07055.