Stable Locomotion in Unstructured Terrain using Curriculum Learning for Online Parameter Adaptation

Sam Shaw, Mateo Guaman

Abstract

We present a learning-based approach to optimize the gait of a hexapod robot for forward progression and stability. Using a central pattern generator (CPG) model for parameterized locomotion, we propose the use of reinforcement learning to learn and adapt parameters online to maximize the distance traversed by the robot in a stable fashion. We present a curriculum of different terrains of increasing complexity as a way of speeding up the learning of the robot and obtaining higher reward over non-learning and learning-from-scratch approaches. Our experimental results show that it is possible for the hexapod to learn to walk over the planks in the most stable way possible by keeping the step height of its gait as low as possible. Setting up a curriculum for the agent to successfully learn to walk over taller obstacles proved to be a challenging and time consuming task, and requires additional work.

1 Introduction

Legged animals use sensory feedback to inhibit or extenuate certain gait characteristics online, allowing them to subconsciously navigate extreme terrain with ease [2, 8]. Inspired by nature, legged robots can similarly traverse complex environments inaccessible to other systems, such as wheeled robots. However, two major challenges are presented: 1) the coordination of the many degrees-of-freedom (DOF) often present on legged systems to produce locomotion, and 2) determining suitable higher-level gait parameters for a given environment. In the context of robotics, central pattern generators [3] (CPG) aid in solving the first problem. Often modelled as a set of coupled oscillators [7], CPGs allow for the generation of parameterized gaits and provide smooth transitions when gait parameters are changed. The limit cycle of the CPG determines the step shape, and other numerical parameters control specific aspects (e.g., step length, step height, step speed).

When the path of the robot and surrounding environment is known *a priori*, it is sometimes possible to set static parameter values (e.g., set a step height larger than the highest obstacle) or encode simple rules [9]. First, such methods

are not suitable for general robot deployment, if the environment is not well-known. Second, such methods are purely reactive; a robot may locomote to a position in which it cannot appropriately react. Finally, these methods encode a significant amount of domain knowledge – which may or may not be optimal.

We are interested in ensuring stable, forward locomotion of a legged system on a variety of terrains – from flat ground to a flight of stairs. We employ a learning-based approach to adapt gait parameters online based on sensory feedback. By adapting gait characteristics such as step height and center of mass position based on this feedback, we will enable a legged system to more naturally navigate obstacles and ensure stable forward locomotion. We aim to apply a curriculum learning approach to facilitate learning – increasing the complexity of the locomotive challenge as learning progresses. For the purpose of this project, setting up an effective learning curriculum to improve learning performance to be prohibitively time consuming; this requires additional work. We devised different obstacles courses with ramps of varying slopes, planks, and debris. However, after trying multiple obstacle courses, we have developed a good intuition of which types of learning curricula to pursue in the future. Specifically, we think slow variations that increase the complexity of the terrain can be used as a curriculum to teach the agent to generalize the task of going over obstacles of varying heights.

2 Background and Related Work

In this section, we present background information on curriculum learning before detailing previous works on learning-based, full-robot control.

2.1 Transfer and Curriculum Learning

In reinforcement learning (RL) problems, we aim to reduce the amount of resources used in the learning process while maximizing the reward that the agent obtains. For complex tasks, transfer learning can be used to reduce the resource requirements for a learning agent. The idea behind transfer learning [11] is that what has been learned from experience for one task can be reused to learn another, but different, task. Transfer learning has been successful in reducing learning time and increasing reward achieved for complex tasks [12] compared to learning from scratch.

We are concerned with Curriculum Learning, a flavor of transfer learning where an agent learns from examples presented to it in a particular order, usually in order of increasing complexity. It has been shown that curriculum learning provides significant improvements in machine learning problems in terms of better generalization, faster convergence speed, and at finding better local minima [1]. In reinforcement learning, curriculum learning can be used to develop sub-tasks as components of a multi-step curriculum with the goal of obtaining better performance in a complex task [5].

2.2 Full-robot Control via RL

The problem of using reinforcement learning to achieve full-body control of a robot has been explored in the past, using a variety of learning techniques. In [4], Kohl and Stone achieved full-robot control of a quadrupedal robot using policy gradient reinforcement learning. Their agent learned to search for gait parameters for locomotion on level, planar terrain that maximize the walking speed of the robot. Their approach significantly outperforms a variety of existing hand-coded and learned solutions. Reinforcement learning has also been used to achieve forward locomotion by learning motion patterns for different segments of a caterpillar robot [13].

Peng et al. [6] used Deep Reinforcement Learning to achieve full-robot control for terrain-adaptive locomotion. They used a mixture of actor-critic experts to achieve locomotion in multiple planar bio-inspired characters and terrain classes: gaps, steps, walls, and slopes. Specifically, their approach directly learns a policy for assigning joint angles and forces to simulated actors.

Sartoretti et al. [10] use an aynchronous advantage actor-critic (A3C) algorithm to learn a decentralized control approach for the locomotion of a modular, articulated snake. Specifically, using torque feedback, they manage to learn decentralized control policies that adapt shape parameters for different windows (segments of the snake). Their learned controller greatly outperforms a shape-based compliant control approach for forward progression in confined environments.

3 Problem Formulation and Technical Approach

We first define the learning problem precisely before presenting our learning approach.

3.1 Learning Problem

In this project, we learn a policy for adapting specific high-level gait parameters online based on the position and orientation of the robot's body for the stable, forward locomotion of a legged robot on unstructured terrain. Note: all of this data could be obtained from an external motion capture system targeting the robot's body. In the context of this project, we update desired step-height and center-of-mass position (along one dimension – in the direction of locomotion). We hold fixed two of the other gait parameters, step-length and step-speed, since for our specific attempted curriculum, adaptation of these parameters to ensure stability and forward progression is redundant with adapting the step-height and center-of-mass position.

3.2 Learning Approach

In this subsection, we detail our learning approach. Specifically, we specify our state and action representations, reward structure, and the learning algorithm.

3.2.1 Actions

As discussed in our statement of the learning problem, in this project, we seek to adapt the robot's step-height and center-of-mass position (single axis – in direction of locomotion) online. Although the robot's gait parameters – including step-height and center-of-mass position – are continuous variables, we only consider them to a finite resolution (e.g., step-height to the nearest 0.1cm). On a real robotic system, smaller difference in gait parameters will be absorbed by hardware tolerances. Therefore, we will avoid learning changes smaller than a specified finite increment.

Considering only finite increments to gait parameters greatly reduces the action space, which allows for easier learning. Thus, in this project, we consider updates to step-height and center-of-mass position as discrete positive, negative, or zero increments. As a result, we have nine possible actions: two possible gait parameters to adapt with three possible discrete increments to apply.

3.2.2 State Representation

Since we consider actions as discrete increments on gait parameters, we must include the true value of these gait parameters in the state. Doing so allows the agent to associate certain actions with current gait-parameter values (e.g., if the center-of-mass is pushed very far forward, the agent will see this in the state and learn not to shift the center of mass further forward). These features are internal to the system.

Additionally, we include features external to our system in the state representation. Since we care about both the robot's stability and forward progression, we include features that describe the robot's body orientation and position in the world. More specifically, we include Euler angles for the body's roll and pitch as well as the position of the body along an axis aligned with the direction of locomotion. The body-roll angle and body-pitch angle allow the robot to know how to shift its center-of-mass for stability. The body position will allow the robot to ensure that it continues to progress forward.

3.2.3 Reward Structure

We use several reward mechanisms to encourage forward progression and successful navigation of obstacles. The agent receives a positive reward of one each time it passes a progression checkpoint for the first time; these checkpoints are placed at 1cm intervals from the origin. Using checkpoints instead of the derivative of the position prevents the agent from obtaining reward by shifting the center-of-mass (from which the position of the robot is measured) of its body back and forth when it gets stuck behind an object. In order to promote getting over all the obstacles in the course, we assign a large positive reward for finishing an episode successfully. On the other hand, the agent received a large negative reward followed by episode termination if it has not moved forward enough after a certain number of steps. This negative reinforcement not only

reduces training time, since episodes are terminated, but also encourages the agent to navigate the obstacles in the most efficient way possible.

In practice, it is important to limit the robot's step-height when possible. Large steps are accompanied with increased instability. Therefore, the robot should increase it's step height beyond a nominal value only when necessary to navigate an obstacle. To encode this in our reward structure, we modeled the source of reward as an asymmetric spring force, with the nominal step height at the spring's equilibrium position. If the step-height parameter remains at or below the nominal value, the agent receives a reward of 0. However, if the step-height parameter is increased by the agent above the nominal step-height value, the agent receives a negative, linearly proportional reward with respect to the height above the nominal step-height value. Therefore, the negative reward acts as a restoring force to bring the step-height parameter to or under the nominal value, ensuring stability.

3.2.4 Learning Algorithm

Since we have a continuous state representation and discrete actions, we apply a deep Q-learning agent to our problem. However, in the context of our problem, it is important to capture temporal relationships in the state. Relating to forward progression, our agent must be able to recognize a lack of change in body position, which would indicate that the robot is stuck on an obstacle. Relating to stability, it is important that the agent be able to distinguish sharp changes in body orientation (those caused by a step up or down) from a constant body orientation (associated with an incline). To capture the temporal changes in state features, we use a Deep Recurrent Q-Network to learn a state-action mapping described above.

4 Experiments and Results

In order to test the validity of our learning approach, we trained an agent to adapt the trainable gait parameters of a hexapod robot. We ran experiments in simulation using a curriculum of different obstacle courses of increasing complexity. in order to compare the speed of convergence to an optimal policy to that of non-learning and non-curriculum learning approaches.

4.1 Evaluation

To evaluate the performance of our gait-parameter control policy learned via a curriculum, we performed a simulated experiment with static gait parameters (no learning involved). We consider this project successful since the learned policy outperforms static parameter choices. We determine performance based on the effectiveness of the policy, which is measured based on cumulative reward.

4.2 Experimental Validation

4.2.1 Environment

For this project, we perform experiments in a Gazebo simulation environment. Learning in a simulation environment allows for faster learning, and avoids wear and tear of the hardware. The legged system that we use for experimental validation is an 18-DOF hexapod robot. Each of the robot's 18 DOFs has encoders for position and torque measurements as well as a 3-axis gyroscope and 3-axis accelerometer for orientation in the world.

We use ROS (Robot Operating System) as the communication infrastructure for the system. The learning agent communicates the learned gait parameters to a mid-level controller, which in turn translates the gait parameters to joint angles for the robot. These angles are then published to the Gazebo simulation, which directly uses them to simulate the movement of the robot. The new pose of the robot (body position and orientation in the world frame) is then obtained from the Gazebo simulation and passed as feedback back to the learning agent.

4.2.2 Learning Implementation

We are using a Deep Recurrent Q-Network to approximate a time-dependent state-action mapping. Our network has two fully connected layers connected to the input state vector. The input layer is connected to a 64-unit, fully-connected layer. This layer is in part connected to a 128-unit fully-connected layer. The output of this layer is the input to a 128-unit LSTM cell, which we use to keep track of changes in the pose of the robot over time. Finally, the output of the LSTM cell is connected to a 9-unit output layer without an activation function, which outputs the Q-values of each action given the input state vector. We found that using two hidden layers before the LSTM cell rather than using a single hidden layer approximated the state in a way that resulted in more efficient learning.

At the starting stage of our project, we were providing the network one example at a time, and optimizing our network using stochastic gradient descent. We soon found issues with this approach, so we experimented with training our network with mini-batches for more accurate gradient approximations. we began using four mini-batches per episode, but ultimately transitioned to one batch per episode. Using one batch per episode increased training speed and improved the quality of the training (in terms of loss).

The choice of our hyperparameters was empirical. We use a learning rate of 0.016, and a discount factor of 0.88. We decrease the exploration rate of the agent from 1 to 0.1 in 256000 steps. Additionally, we consider updates to the robot at 50Hz and allow for increments of 0.1cm to the step-height and center-of-mass position at each timestep.

4.2.3 Curriculum

In an attempt to facilitate learning, we planned to employ a learning curriculum. In order to develop a curriculum for the agent, we generated different scenarios in Gazebo with increasingly levels of complexity in terms of height of the obstacles in the environment. We believed that slowly increasing the difficulty of locomotive challenges would guide the agent to learn how to adapt gait parameters one-by-one for both stability and forward motion. With regard to the effectiveness of the learned policy, we expected that for a fixed number of episodes, the policy learned by the curriculum approach would outperform a policy that we attempted to learn directly.

Originally, we intended to develop a curriculum using ramps of varying steepness along with simulated debris, meant to parallel unstructured terrain for the real robot (Fig. 1). This environment resulted in interactions unlike those experienced during trials with the real robot, so we narrowed our focus to having the agent learn to go over obstacles of various heights by developing a curriculum with level terrain and obstacles of varying height. In this curriculum, smaller obstacles were presented before taller obstacles. This way, the agent first learns to go over obstacles without raising its step height to the maximum, and then it learns to raise its height only as necessary to get over the obstacle. With this intent, we developed levels with courses made of three horizontal planks of different heights.

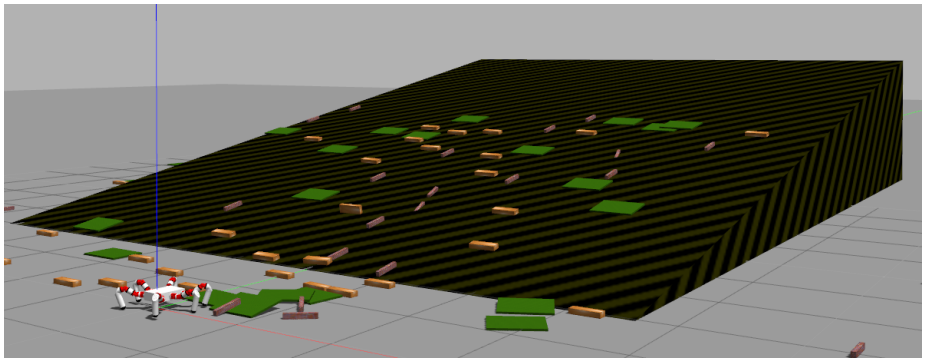


Figure 1: Series of plank obstacles.

All of the levels in the curriculum are structured in the same way. There are three planks that the robot has to go over to reach the finish line (Fig. 2). We slowly increase the height of the planks at each level of the curriculum. We expect that by doing this, the agent learns to adjust its step height and center-of-mass position to get to the finish line in the most stable way possible. The main challenge the agent faces at each level of the curriculum is to learn to go over taller obstacles while still remaining as stable as possible.

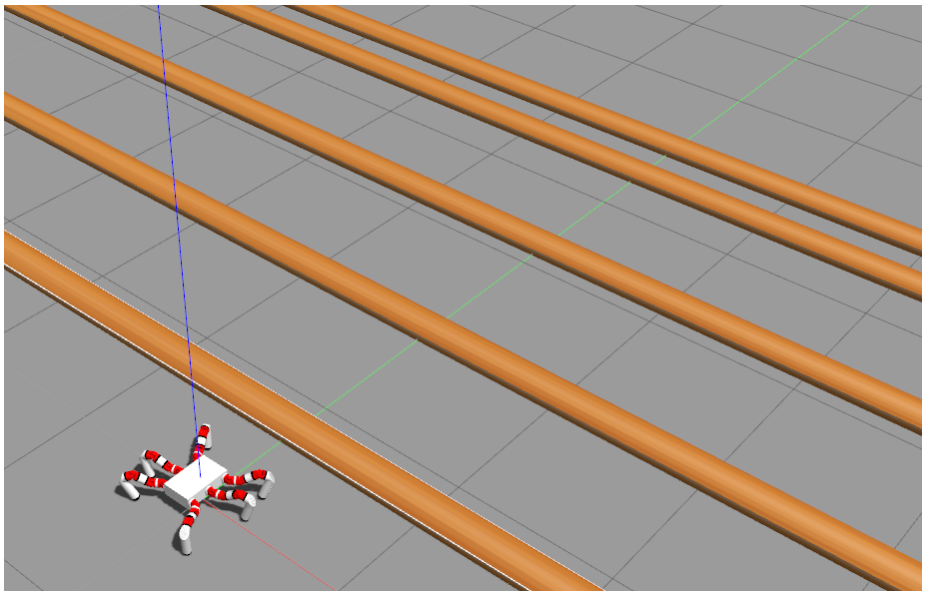


Figure 2: Series of plank obstacles. Finish line shown as two planks in the distance.

4.3 Experiments

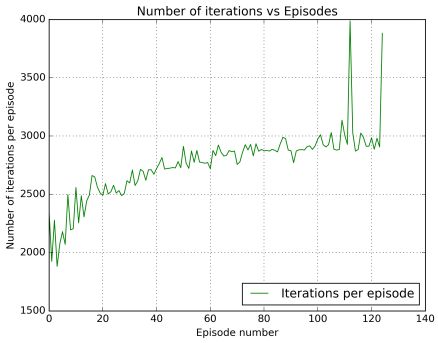
Step Height	Reward	Steps
Learned (Dynamic)	439.7	2884
5 cm (Static)	450	2625
5.1 cm (Static)	184.4	2656
5.5 cm (Static)	-747	2394

Table 1: Comparison between reward and steps to complete episode for learned vs static parameters for step-height.

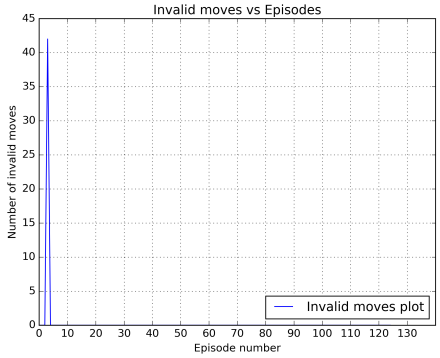
We began by training the agent to navigate the first level of our curriculum. This level featured a course of three 5cm-tall planks, spaced 1m apart. The

figure below shows the best results achieved on this level of the curriculum, after fine-tuning the reward structure and learning parameters.

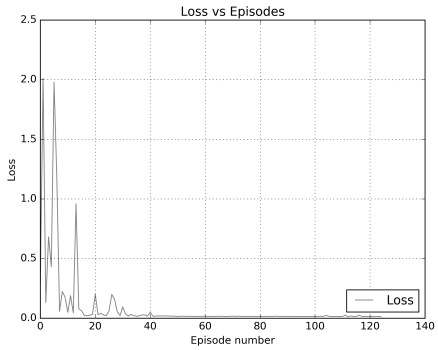
(a) Reward vs Episode. Reward converges to a value close to 400. A stable policy has been achieved.



(b) Iterations per Episode vs Episode. If the robot is stuck for too long, the episode terminates; with more training episodes, the robot's progression improved.



(c) Number of invalid inverse kinematics (IK) moves vs Episode. A move is considered invalid if the foot position requested is outside the robot's workspace – no joint configuration exists.



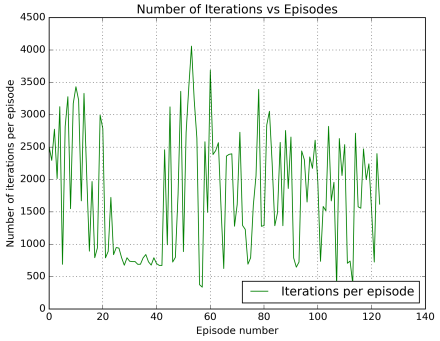
(d) Loss vs Episode. Loss steadily decreases as episodes progress.

Figure 3: Results for Curriculum Level 1.

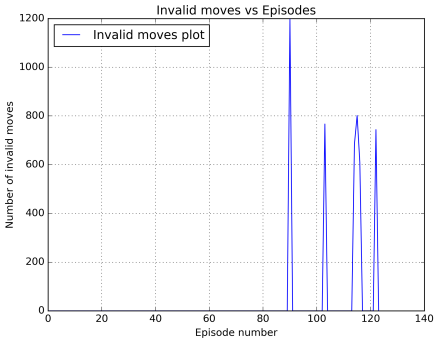
Once the agent had arrived at a stable, effective policy for the first level of the curriculum, we introduced it to the second level. This level featured a course of three 7cm-tall planks, spaced 1m apart. We reset the episode to this new level manually, by loading a saved model of the weights of the network obtained by training in the previous level. We select the model at an episode soon after the reward has already converged. When we load the model obtained in the previous level, we needed to tweak the reward structure and exploration rate of the agent. The proportions of the rewards obtained in level 1 was not the same as that

in level 2, which induced undesired behavior such as the agent getting more reward for getting stuck than for completing an episode. Therefore, we needed to adjust the reward scales in order to ensure the right behaviors were rewarded and punished. Additionally, whenever we changed the level to a more complex one, we increased the exploration rate back to 1 and decreased it linearly to 0.1. However, the rate of decrease of epsilon was twice as fast as the one for level 1, to keep the knowledge obtained from level 1. The figure below shows the best results achieved on this level of the curriculum, after fine-tuning the reward structure and learning parameters.

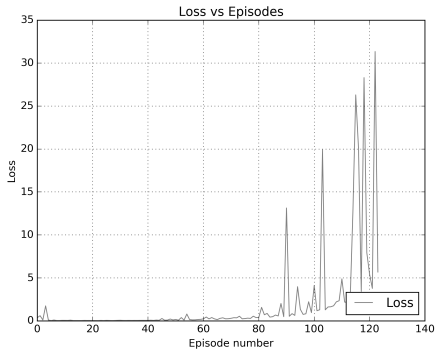
(a) Reward vs Episode. Reward seems to converge to a value close to 0, but then oscillates. A stable policy has not been achieved.



(b) Iterations per Episode vs Episode. For episodes with iterations less than 2500, the robot got stuck.



(c) Number of invalid inverse kinematics (IK) moves vs Episode. A move is considered invalid if the foot position requested is outside the robot's workspace – no joint configuration exists.



(d) Loss vs Episode. Loss increases as episodes progress.

Figure 4: Results for Curriculum Level 2.

5 Discussion

Fig. 3 highlights the performance attained after training the learning agent on the first level of our curriculum; overall, these results are successful. Since we choose a nominal step-height of 5cm, these planks are not tall enough to completely block the robot in most cases, but do regularly slow it down. The intent of this step was to teach the agent to raise its step height only when necessary, rather than maximizing its step-height so that it can step onto and over any obstacle, since doing this induces instability. Sub-Fig. 3b, which shows the episode length over time, demonstrates that as the agent's policy improved it became stuck on the planks less; we know this, because getting stuck results in episode termination. Sub-Fig. 3a, which shows the reward over time, demonstrates that the agent ultimately learned to increase its step-height over the nominal value only when necessary. Early on, the agent can cross all obstacles by prematurely setting a high step-height, but this results in a large reward penalty for lack of stability. Finally, Sub-Fig. 3c, which shows the number of invalid actions – those which attempted to command a leg outside of the robot's workspace – we're quickly rejected. These actions cause significant instability, and hinder forward progression.

After pretraining the agent on the first level of the curriculum, we introduced the second level of the curriculum – a new challenge with higher obstacles. Fig. 4 highlights the performance attained after training the learning agent on the second level of the curriculum. As shown, the agent is unable to adapt to the new challenge presented and performance decreases rapidly from what was learned during the first level of the curriculum. Originally, after completing each level, we increased the height of the planks by 2 cm. After attempting to train multiple agents using this curriculum, we realized that this curriculum was too hard for the agent to complete. We changed the height-increase of the planks to be 0.5 cm, but we found that this task was still to hard for the agent to learn. We believe there could be a number of causes, but failed to completely isolate the issue. When starting a new challenge with a pretrained agent, it is important to initially increase exploration in order to allow the agent to adapt to the new challenge. However, if too much exploration is permitted, then it is possible that the agent's learned policy (from earlier stages of the curriculum) could be destroyed. It is possible that we did not find the right balance between policy adaptation and preservation when transitioning to the new level of the curriculum.

6 Conclusion

6.1 Limitations

We faced significant challenges working on this project, mainly regarding setup overhead and training speed of our agent. Much of the first few weeks were spent on setup. First, we configured the simulation environment for the robot in Gazebo and built the different courses for our curriculum. Additionally, it was necessary to prepare a mid-level CPG-based control mechanism to parameterize the robot's gait, and incorporate this controller in a custom gym environment for interfacing with our learning agent. In figuring out a reasonable curriculum for the scope of this project, we designed different levels with ramps of various steepness angles, debris of different sizes and shapes, and finally, long planks of different dimensions.

After setting up the simulation environment for the robot and designing the learning agent, we faced issues with training the robot. We were unable to speed up the Gazebo simulaton by any amount. We are unsure whether this issue was generated by an inaccurate robot model or simply by the nature of the problem. After heavy optimization of our task at hand, we still ended up having episodes that took anywhere between 1 and 2 minutes to finish. In total, each run of our learning agent took approximately four hours, which meant that there was not much room for experimenting around with different values for all the hyperparameters, different network structures, different state representations, or different reward structures. Therefore, running multiple learning agents to average their performance was prohibitively expensive. The high training time for our agent meant that designing a more complex curriculum was not possible given the short amount of time to complete this project.

6.2 Future Work

Future work should involve continued development of a suitable learning curriculum. Beyond this, future experiments could implement the final simulated approach using the robot's on-board IMUs. Additionally, future experiments could augment on-board IMU sensing (e.g., with lidar) with additional feedback on the environment's structure, necessary for anticipatory control. Doing so would allow the robot to both react to its environment, and also anticipate future parameter changes (e.g., an increase in step height to navigate a large obstacle). Anticipatory control is necessary to ensure that the robot avoids locomoting into a position/situation where it is unable to react.

7 Permissions

We have permission from Professor Howie Choset to use the "Snake Monster" Gazebo simulation environment for this project, and have proposed robot experiments if the simulated results appear promising.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[2] Philip Holmes, Robert J Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *SIAM review*, 48(2):207–304, 2006.

[3] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[4] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624. IEEE, 2004.

[5] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 566–574. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[6] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):81, 2016.

[7] Ludovic Righetti and Auke Jan Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 819–824. IEEE, 2008.

[8] Serge Rossignol, Réjean Dubuc, and Jean-Pierre Gossard. Dynamic sensorimotor interactions in locomotion. *Physiological reviews*, 86(1):89–154, 2006.

[9] Guillaume Sartoretti, Samuel Shaw, Katie Lam, Naixin Fan, Matthew Travers, and Howie 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.

[10] Guillaume Sartoretti, Yunfei Shi, William Paivine, Matthew Travers, and Howie Choset. Distributed learning for the decentralized control of articulated mobile robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6. IEEE, 2018.

[11] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[12] Matthew E Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(Sep):2125–2167, 2007.

[13] Ryota Yamashina, Masafumi Kuroda, and Tetsuro Yabuta. Caterpillar robot locomotion based on q-learning using objective/subjective reward. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 1311–1316. IEEE, 2011.