# Beyond Locomotion for Legged Robots

Parth Patil *Purdue University*

***Abstract*—Quadrupeds are rapidly gaining popularity in various applications. Although locomotion of quadrupeds has been extensively researched, there has been limited exploration into enhancing their capabilities beyond walking and running. One such capability observed in nature is the use of legs as manipulators to interact with objects in the environment, such as pushing buttons or grabbing objects. This project aims to enable this capability in quadrupeds through reinforcement learning (RL) and then dividing the problem into smaller subproblems. Separate RL policies can be trained for specific subproblems, such as locomotion (moving from point A to point B) and manipulation (e.g., pushing buttons while standing still). A higher-level decision algorithm will then be employed to select between these policies, similar to the approach introduced by Cheng et al. [1]. IsaacGym [2] is used for the simulation platform to accelerate the learning algorithm**

***Index Terms*—Robotics, quadruped, Reinforcement learning, manipulation**

## I. Introduction

THE field of robotics is ever-growing in this era. Quadruped robots have received quite some attention because of their versatility and ability to perform a large array of tasks. You can mount numerous attachments on the back of a quadruped and use them in different environments. For e.g., one can attach a sensor array and use a robot to create a 3D map of the environment, or one can mount tools on the back and use them in military or industrial settings. This inherently led to significant advancements in the locomotion problem of these robots.

Fig. 1. A quadruped robot mounted with a 2-link manipulator and communication equipment. Reference: Photo from Google Photos $https$ : $//orisalessm.live/product_details/14420612.html$

Though physically these robots are much more capable than moving from one point to another. All four legs of the robot can be considered to be robotic arms and can perform all actions similar to an arm with two degrees of freedom. We will focus on this and lay down a framework to accomplish performing manipulation actions using a single arm.

## II. Related Work

### A. Legs as Manipulators [1]

In this paper, Xuxin Cheng et al. [1] the primary goal is to extend the capabilities of quadruped robots beyond conventional locomotion, aiming to achieve agility and versatility similar to that of biological counterparts like dogs and cats. This is achieved by a two-part approach. First, training an RL-based policy for sub-tasks like walking, climbing, and manipulation. Secondly, it t introduces a behaviour tree-based network to divide the task into sub-tasks and choose the appropriate policy to complete said sub-tasks. This project is heavily inspired by this research and tries to replicate their work using a Unitree B1 instead of Unitree Go1 used by the authors.

### B. Extreme Parkour [3]

Xuxin Cheng et al., in this paper, tries to expand the quadruped by enabling them to perform parkour. Historically, robots have always been limited in the actions they perform and are tied down to slow movements because of the algorithms and control techniques used. Even though, physically capable of performing impressive manoeuvres, it is difficult to devise a classical algorithm for the same. This paper uses an end-to-end RL-based network to achieve robot jumps twice its body length.

### C. IsaacGym [2]

IsaacGym is a high-performance platform for training various robots. This is a GPU-accelerated platform and provides both physics and policy training acceleration. With the recent boom in artificial intelligence, the outlook towards robotic algorithms has also seen a huge shift. A large portion of the research is now focused on a machine learning-based approach for control and planning. With the advent of a new platform like IsaacGym [2], which enables fast learning for large-sized networks, many approaches earlier thought to be impossible before have become feasible.

## III. Background

### A. Reinforcement Learning

Reinforcement Learning is a machine-learning technique used to control the actions of intelligent robots/actors in a dynamic environment. This algorithm tries to optimize the reward gain by performing actions while also exploring multiple paths to further optimize its actions [4].

*1) Problem Setup:* THe Reinforcement Learning problem is a Markov Decision Process (MDP) with the state given by $< S, A, P, R, \gamma >$ for any given time. Here,

- S: is a finite set of states.
- A: is a finite set of actions.
- P: is the state transition probability

$$P^a_{SS'} = P\left[S_{t+1} = S' | S_t = S, A_t = a\right] \tag{1}$$

- R: is the reward function (feedback signal)

$$R^a_S = E\left[R_{t+1} | S_t = S, A_t = A\right] \tag{2}$$

- $\gamma$: is the discount factor

We also define a few more functions,

- Return ($G_t$): it is the total discounted reward from step $t$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3}$$

- Policy ($\Pi$): A function that maps state to actions.

$$\Pi : S \rightarrow A \tag{4}$$

- Value Function ($V$): It is defined as the expected cumulative reward from following the policy $\Pi$ from State $S$

$$V^{\Pi}(S) = E\left[\sum_{t \geq 0} \gamma^t r_t \middle| S_0 = S, \Pi\right] \tag{5}$$

- Q-function: Also known as action-value function, is similar to value function. It is defined as s the expected cumulative reward from taking the action $A$ in State $S$ and then following the policy $\Pi$

$$Q^{\Pi}(S, A) = E\left[\sum_{t \geq 0} \gamma^t r_t \middle| S_0 = S, A_0 = A, \Pi\right] \tag{6}$$

Thus, the reinforcement learning problem can be defined as learning the Policy $\Pi$ for a given environment such that you maximize the cumulative reward.

*2) Vanilla Reinforcement Learning Algorithm:* Using the terms defined above, a vanilla reinforcement algorithm is defined by Algorithm 1. Note that there are a lot of improvements that can be made to this algorithm to improve its performance, but the main idea remains the same.

## IV. Methods

The proposed methodology revolves around setting up an accelerated environment for training the Unitree B1 robot and training locomotion and manipulation policy in the said environment. Further, these policies can be extended to be used in conjunction as done by X. Cheng, et al. [1]

**Algorithm 1** REINFORCE Algorithm

```
Initialize policy π
Set learning rate α, discount factor γ
for each episode do
    Initialize state s
    while s is not terminal do
        Choose action a using policy π: a = π(s)
        Roll out actions using π:
            Perform actions a_1, a_2, ..., a_n using π, resulting in sequence of states s_1, s_2, ..., s_n
        Calculate total reward of rollout: R = Σ_{i=1}^{n} γ^i r_i, where r_i is the reward at state s_i
        Update policy π:
            Update policy parameters using gradient ascent: θ ← θ + α · ∇_θ log π(a|s) · R
        Update state: s ← s_n
    end while
end for
```

### A. Simulation Platform and Setup

As stated, this project uses the IssacGym [2] platform to utilize accelerated training on GPUs. The project first begins by setting up the environment and then loading **4096** B1 robots on a flat plane as shown in Figure 2

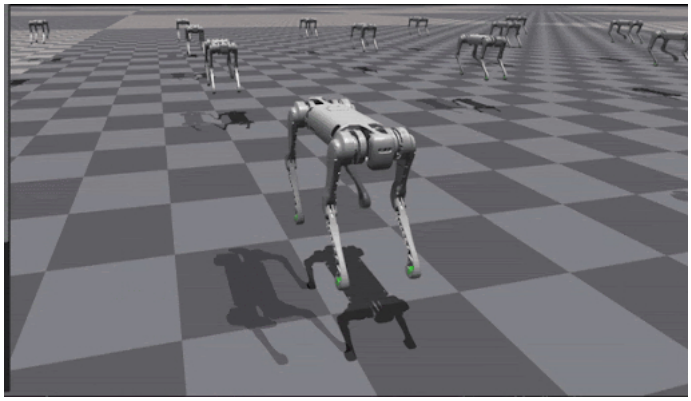

Fig. 2. Isaacgym environment loaded with the Unitree B1 Robot

### B. Environment and Action space

The observation space of the robot is 48-dimensional and contains not only the position and orientation quaternion but also joint acceleration, hip position, Z-velocity, etc to help with the smooth motion of the robot.

The environment consists of a 12-dimensional output space. The policy outputs the target joint angles $a_t \in R^{12}$ at 50Hz, as described in X. Cheng et al. [1] The action space contains 3 torque values for each of the four legs of the robot.

### C. Reinforcement Learning setup using IssacGym

Throughout this project, three main repositories were used to accomplish Reinforcement Learning within IsssacGym. All these repositories provide an increasing level of features for training a quadruped robot.

*1) IsaacGymEnvs [2]:* This repository situated at https://github.com/NVIDIA-Omniverse/IsaacGymEnvs provides the vanilla implementation of RL examples provided by the authors of IsaacGym [2]. Even though this repository consists of examples for the ANYMal robot, which is a similar quadruped, the code is not easily configurable.

*2) legged_gym [5]:* This repository tackles the problem in the previous one by offering a wide array of examples for a quadruped, with good initial configurations. This also consists of multiple configurations of different terrains, which are easy to set up.

*3) extreme-parkour [3]:* This is by far the most feature-rich repository, which builds upon both the repository mentioned above and provides very good configurable options which show promising results on different robots.

## V. Experiment Results

### A. Toy Problem setup and results

This project first started with a toy project, wherein isaacgym was used to train RL policy of a simple Cartpole problem. This took less than **6** minutes to train a policy using **4096** parallel environments

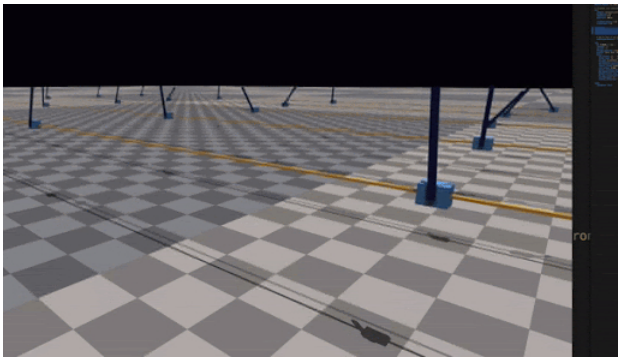

Fig. 3. Toy Problem: Cartpole RL training on 4096 environments

### B. Locomotion policy: using ANYmal robot

Locomotion policy on ANYmal robot was perfectly trained using the example provided in the legged_gym repository

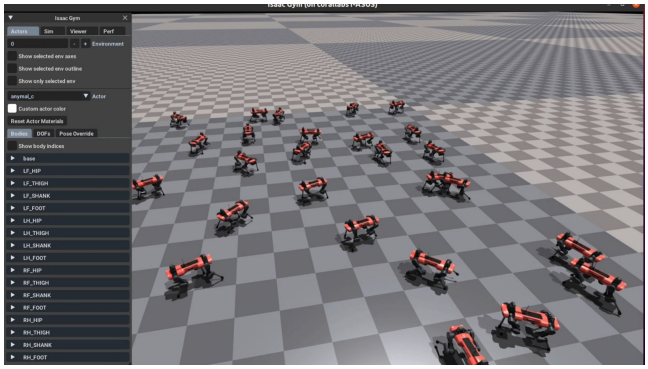

Fig. 4. Locomotion policy: using ANYmal robot

### C. Parkour Policy: using A1 Robot

The example from extreme_parkour for the A1 robot was trained with good results. Here, the robot is able to jump from one obstacle to another.

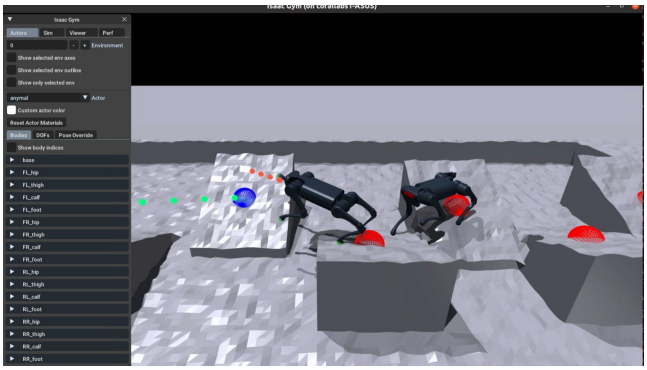

Fig. 5. Parkour Policy: using A1 Robot

### D. Locomotion Policy: using B1 Robot

Even though the example from this repository ran perfectly, significant difficulty was faced when trying to do the same with the Unitree B1 robot. Because of the different configurations and lengths of the robot, many changes were required to make it work. I was able to train a locomotion policy on B1, but it doesn't show a very good walking trait. As you can see, the legs are spread apart.

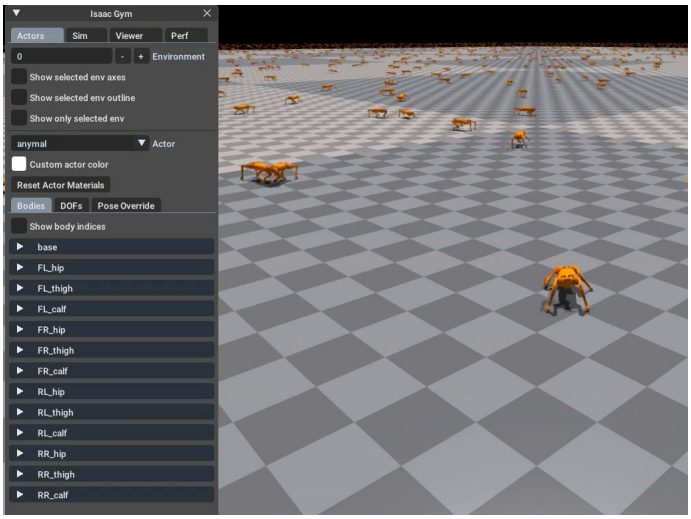


Fig. 6. Locomotion Policy: using B1 Robot

## VI. Conclusion

In conclusion, the project was able to complete two major milestones, which were setting up Isaacgym for RL training and getting a working locomotion policy on the Unitree B1 robot. Even though training on the manipulation policy was not completed, it can be built upon the resources provided in the project.

Quadruped are very capable robots. Using advanced reinforcement learning, we can achieve amazing feet of control using these robots.

## VII. Future work

The following is a natural extension of this project

- Training a Manipulation policy to control the leg of the robot.
- Try to improve the walking trait, which was trained on the B1, and achieve something similar to the ANYmal and A1 robots.
- Design complex tasks where both of these policies would be used in conjunction, for example, go to a point and then move a box out of the way.

## References

[1] X. Cheng, A. Kumar, and D. Pathak, "Legs as Manipulator: Pushing Quadrupedal Agility Beyond Locomotion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[2] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning," *arXiv preprint arXiv:2108.10470*, 2021.

[3] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme Parkour with Legged Robots," *arXiv preprint arXiv:2309.14341*, 2023.

[4] "Reinforcement learning," *Wikipedia*, Available: https://en.wikipedia.org/wiki/Reinforcement_learning

[5] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," *arXiv.org*, Sep. 24, 2021. [Online]. Available: https://arxiv.org/abs/2109.11978