Multi-Agent Quadruped Planning

Saudamini Ghatge
Robotics Institute, Carnegie Mellon University
sghatge@andrew.cmu.edu

Abstract: Significant research efforts are underway to develop local locomotion machine learning (ML) policies tailored for complex robotic systems that require sophisticated controllers. These local locomotion policies are designed to navigate obstacles effectively while also adapting to uneven terrain, enabling robots to successfully reach their designated local waypoints. This project builds upon such advancements by exploring the application of these local policies in scenarios involving multiple robots operating within a shared workspace. The primary focus of this course project is to integrate Multi-Agent Path Finding (MAPF) algorithms with pre-trained local locomotion policies. The goal is to generate collision-free paths for quadruped robots as they navigate and interact within the same environment, ensuring seamless coordination and efficient task execution. [Github Repository]

Keywords: Reinforcement Learning, Multi-Agent Path Finding, Locomotion

1 Introduction

In contemporary robotics applications, warehouse operations are primarily dominated by relatively simple robotic systems designed to perform repetitive and structured tasks efficiently. However, there is substantial potential for future warehouses to transition toward integrating more advanced robotic systems capable of handling complex, dynamic operations. Beyond structured indoor environments, the field of outdoor robotics presents even greater challenges. Tasks involving multiple robots navigating through demanding and unstructured environments—such as hiking trails, mountainous terrain, or dense forests—necessitate advanced coordination strategies. These settings often include natural corridors or bottlenecks, adding another layer of complexity to multi-robot systems.

Furthermore, disaster recovery and exploration scenarios represent critical domains where multi-robot teams play a pivotal role. Robots deployed in such situations are tasked with navigating unknown and hazardous environments, constructing detailed maps, and retrieving specific objects or samples. These scenarios emphasize the increasing importance of sophisticated multi-robot coordination, where adaptability and collaboration between robots are paramount for mission success. In these contexts, Multi-Agent Path Finding (MAPF) algorithms have been extensively utilized to generate solutions for coordinating multiple robots in shared workspaces. However, while MAPF algorithms can effectively plan collision-free paths, the solutions they generate often come with significant constraints that can limit their flexibility and applicability, particularly in dynamic or unstructured environments.

In the case of quadrupedal robots, recent advancements have focused on training local locomotion policies for velocity control. These policies enable quadrupeds to traverse uneven terrain and avoid obstacles effectively. Leveraging these pre-trained local policies in a multi-agent setting offers an exciting opportunity to address some of the limitations imposed by traditional MAPF solutions. By incorporating these policies, it is possible to reduce the constraints typically associated with multi-agent coordination, potentially leading to more adaptive and efficient solutions.

This report aims to explore and formulate the problem of multi-agent coordination specifically for legged robots, focusing on the integration of pre-trained local locomotion policies with MAPF algorithms. The objective is to demonstrate how these combined approaches can enhance the capabilities of multi-robot systems operating in complex environments.

2 Prior Work

Classical MAPF algorithms typically use heuristic based search to find solutions using centralised methods. This makes them difficult to scale when we have to run them with short timeouts. There are Machine Learning based approaches which learn policy for each agent, and are appealing as they could enable decentralized systems and scale well while maintaining good solution quality. But these policies are generally local policies that is, they only plan for a single timestep and have poor success rates and scalability. Recent work has been done on long horizon planning by using heuristic search methods [1]. The method can plan paths for agents in long horizon using a MAPF algorithm like Conflict based search (CBS)[2].

A recent work has been done on a similar application. This work done by [3] focuses on creating a framework which allows training such coordinating policies in a complex environment for quadrupeds. The authors performed experiments on tasks where two quadrupeds are trying to pass through a narrow gate by leveraging local locomotion policy. Although their higher level planner requires one robot to wait while the other passes the gate. There is also the scenario of two quadrupeds trying to cross a bridge where each robot tries not to fall off the bridge while crossing by pushing the other robot.

3 Problem Statement

Similar to the work done in [3], we are focussing on the problem of quadrupeds trying to cross a narrow passageway. The main difference is in the formulation of the policy. Instead of allowing only one quadruped to pass by pushing the other down, we want to train a policy to allow both the quadrupeds to pass.

The problem is formulated in a Multi-agent setting. Formally, given in a graph $G(V, E)$, n agents a_i and $s_i \in V$, $g_i \in V$ being the start and goal location vertices, MAPF attempts to find a path π_i for each agent by searching through a constraint tree to solve conflicts. Two possible conflicts are considered, namely vertex conflicts and edge conflicts. Vertex conflicts happen when two agents try to go to the same vertex at the same timestep. Edge conflicts happen when two agents try to swap locations at the same timestep. This report focuses on Edge conflicts.

The core concept is to utilize a pre-trained policy whenever a constraint, such as an edge conflict, is encountered within the constraint tree of a multi-agent pathfinding algorithm like Conflict-Based Search (CBS) [2]. The basic idea of CBS is that it performs two searches - high level search which maintains a constraint tree Fig:1 and searches a optimal node in the tree, and a low level search which finds single-agent paths for all agents using A*. So if two agents are trying to pass through a corridor, CBS will possibly give a solution such that only one agent can pass through it while the other agent waits. However in the case of quadrupeds it could be possible for the two agents to squeeze through if they both try to move carefully around each other. So we try to train a policy for this scenario such that it can be used online whenever an edge conflict is found.

4 Approach

4.1 Scene Modeling

We use the Isaac Lab framework [4] to simulate a scene featuring two quadrupeds positioned across from each other in a hallway. The choice of Isaac Lab is driven by its library of readily available robot learning algorithms, which can be directly utilized to design a new policy. Fig:2 shows the

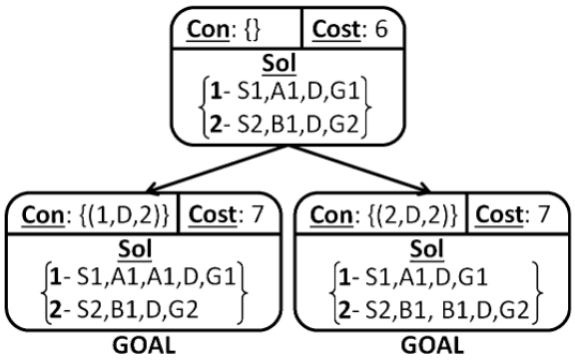

Figure 1: An example constraint tree.

scene of two quadrupeds standing in a hallways, where the walls are represented in red color primitives.

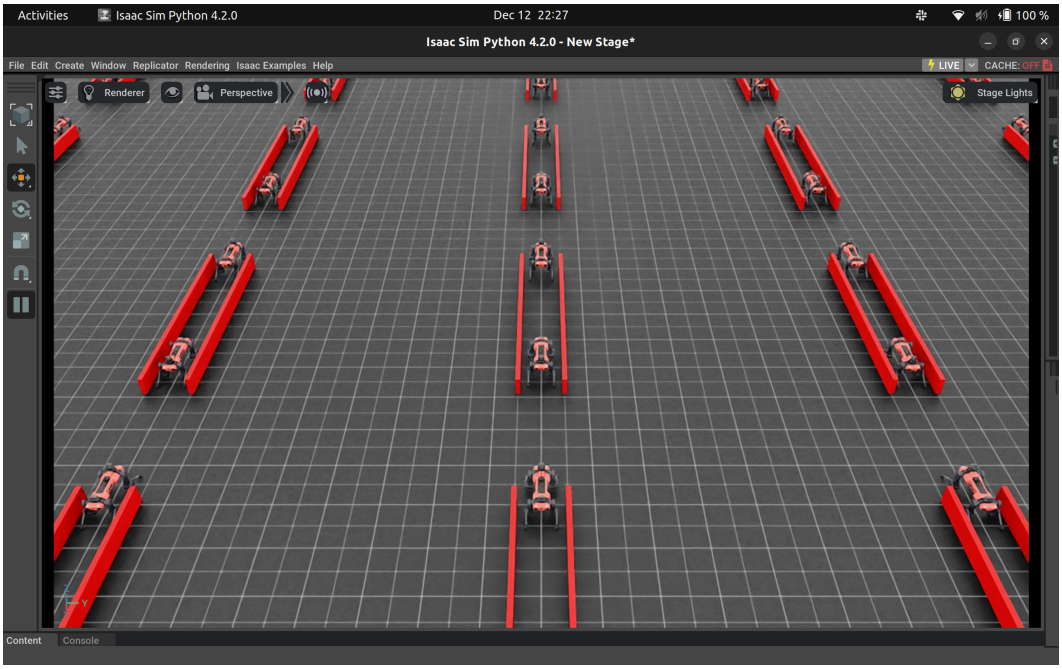

Figure 2: A snippet of scene created in Isaaclab representing two Anymal-C robots and primitives for walls.

4.2 Problem Formulation

We first define variables for a single agent. Let a_s be the action space, o_s be the observation space, s_s be the state space, r be the reward. Then for two robots in the same scene with a single policy, the action space will be $2 * a_s$, observation space will be $2 * o_s$, reward for both the agents will be $r1$ and $r2$ respectively, with total reward equals to $r = r1 + r2$.

The reward function is defined based on the distance to waypoint from each agent. So the goal of the two agents is to go to their waypoints. The episode terminates when the agents reach their respective goal locations.

We use SKRL to train our policy by using the standardized implementation of PPO.

5 Results

We began by conducting two foundational single-agent quadruped tests to establish baseline performance and validate the initial training setups. All the tests were performed on the ANYmal-C robot operating on a flat terrain. The first test focused on training a locomotion policy designed to achieve effective velocity control. For this test, the training environment was readily available through the Isaac Lab framework, simplifying the setup process and allowing us to quickly initiate the training phase.

The second test extended the first by training a navigation policy for a single quadruped tasked with reaching a specific goal location. This navigation policy was built on top of the previously trained locomotion policy, leveraging its capability for precise velocity control. To create a consistent training scenario, the spawning position of the quadruped and its goal location were fixed at each environment reset call. This ensured uniformity in training conditions and allowed for systematic evaluation of the policy's performance.

Figure 3 illustrates the reinforcement learning environment used to train the navigation policy. The task for the quadruped was to traverse to the opposite side of the passageway while maintaining trajectory fidelity. To accelerate the learning process, we deployed 50 parallel environments during training, which significantly reduced the time required for convergence. The policy successfully trained within 12,000 steps, completing the process in approximately 30 minutes.

The effectiveness of the trained policy is evident from the results. Figure 4 shows the policy loss curve for the Proximal Policy Optimization (PPO) algorithm used in this task. As the training progressed, the policy loss steadily decreased, eventually converging to a value close to zero at the final timestep. This convergence indicates that the agent effectively learned to perform actions that maximize its cumulative reward. In practical terms, it signifies that the quadruped successfully learned to reach its goal location while minimizing deviations.

Additionally, Figure 5 presents the reward curve for position tracking, which complements the loss curve by providing further insights into the agent's learning progress. The reward curve demonstrates a consistent upward trend, eventually stabilizing after approximately 8,000 steps, signifying the agent's ability to maintain a consistent trajectory toward the goal location. Both the loss and reward curves show the algorithm's steady improvement and eventual convergence, highlighting the effectiveness of the training process.

5.1 Challenges

After successfully setting up the environment for a single quadruped, I advanced to the more complex scenario of managing two quadrupeds within the same simulation. This step was crucial because our goal is to train a unified policy for scenarios involving tasks like swapping positions, which requires seamless coordination between multiple robots. To achieve this, I integrated two robot configurations into the same scene and ensured that all necessary API calls were made for each robot individually. This allowed the creation of a Reinforcement Learning (RL) environment designed specifically for two quadrupeds operating together.

While I was able to establish the RL environment, I encountered some challenges related to initialization. These issues stem primarily from the limited support currently available for Multi-Agent Reinforcement Learning (MARL) examples in the Isaac Lab framework. Since MARL is still under active development, the documentation and examples are not as comprehensive as those available for locomotion tasks. For instance, while locomotion examples are well-documented, the MARL documentation currently provides only a single example environment with characteristics similar to high-degree-of-freedom robots.

Despite these challenges, I view this as an exciting opportunity to deepen my understanding of the MARL framework. It has been an engaging process to explore and experiment with the various methods available in the MARL class, uncovering their functionality and potential applica-

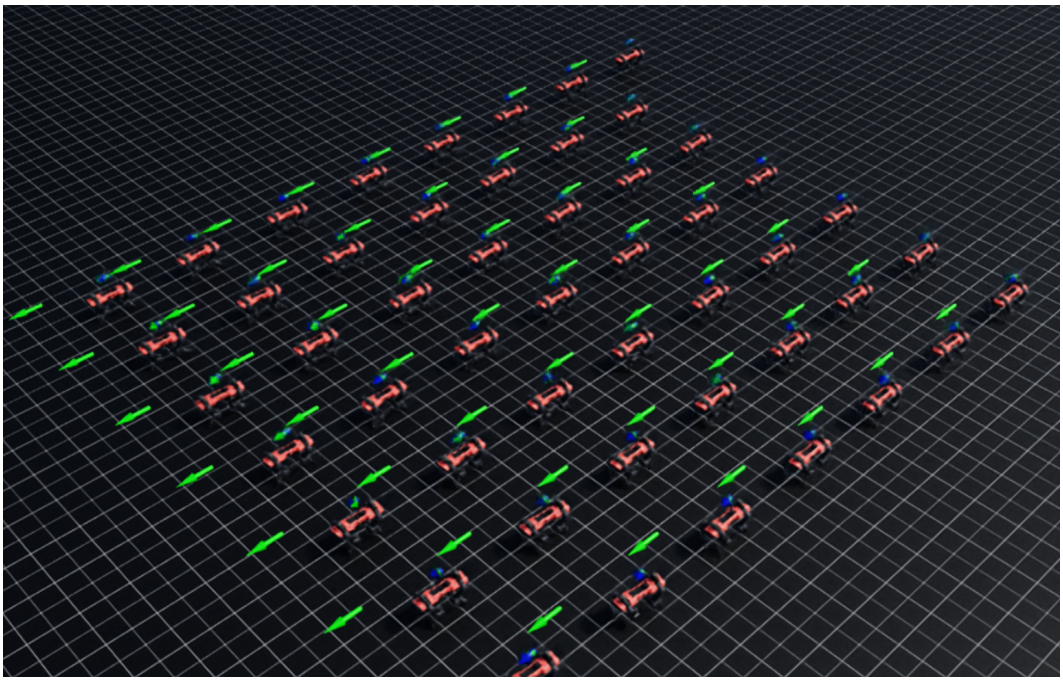

Figure 3: A visualization of the scene created for training the Navigation policy, where green arrows represent the direction of the goal location.

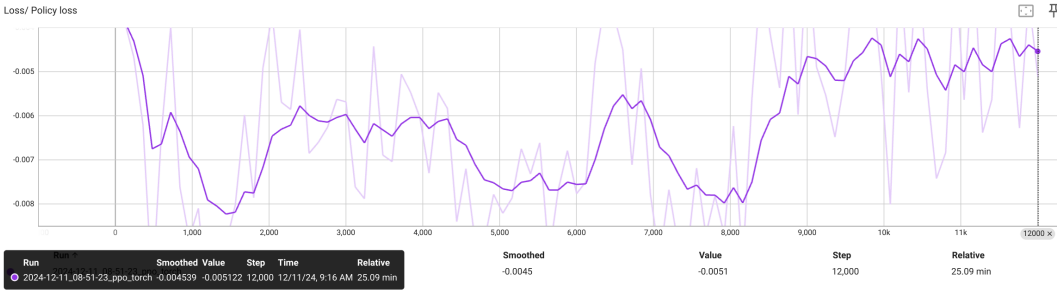

Figure 4: The policy loss curve of the PPO algorithm.

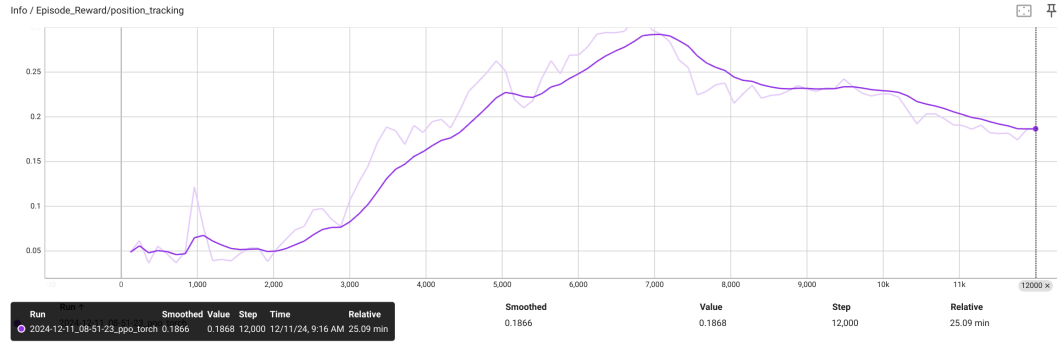

Figure 5: Reward functions for position tracking of the quadruped.

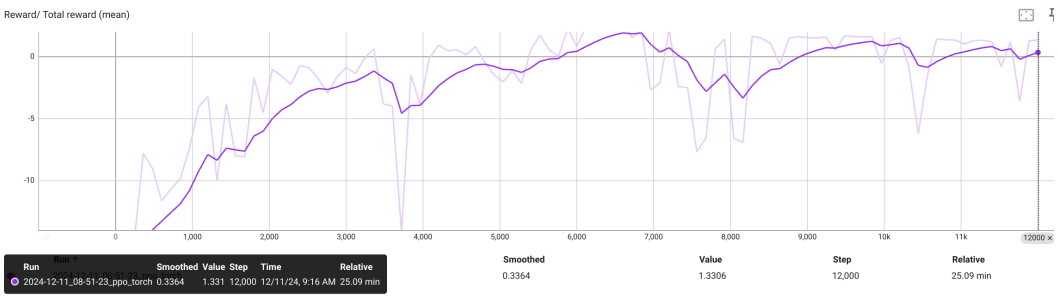

Figure 6: Total rewards for the task.

tions. While the lack of direct examples and guidance poses difficulties, it also encourages creative problem-solving and a deeper engagement with the framework's capabilities. This exploration has been a rewarding learning experience, and I am optimistic that continued experimentation will lead to innovative solutions for multi-agent coordination using the Isaac Lab framework.

6 Summary

In summary, our work tries to create benchmark RL environments in Isaac Lab framework, which can further be used by people working on Multi-Agent path finding algorithms for complex robots. The advent of such robots in warehouses and other applications gives us more incentive to train more such complex policies which can be leveraged to reduce constraints in a MAPF algorithm. Throughout our work, we managed to accomplish the following:

1. Creating a Reinforcement Learning environment capable of adding two high dof robots in one scene.
2. Leveraging a pre-trained locomotion policy to train a high-level policy.
3. Training a single control policy for two cart-pole pendulum configurations in the same scene. This was done prior to setting up environment for two quadrupeds. Since the complexity of the robots in this case is less, it was a trivial task to setup unlike the quadruped environment. Fig 7 shows the RL environment with two cart-pole pendulums placed together for a single velocity control policy.

7 Future Directions

Although the task seemed trivial at start, it poses significant challenge with respect to creating the RL environment, as described in section 5.1. Looking ahead, there are several key directions we aim to pursue to extend and enhance the work conducted so far. These future tasks are designed to build upon the foundational efforts described and address some of the current limitations, while also exploring new possibilities to further optimize and refine the system. By addressing these goals, we hope to advance the capabilities of our RL framework and achieve more sophisticated and robust solutions for complex multi-agent scenarios. The following are the tasks we want to work on -

1. Completing the RL environment setup for two quadrupeds.
2. Training a single RL policy for the velocity control of the two quadrupeds. This is essential if both the quadrupeds are trying to walk in the same constraint space, they need to take into account each others action space.
3. Using the above single locomotion policy for training higher-level policy of reaching to a goal location.

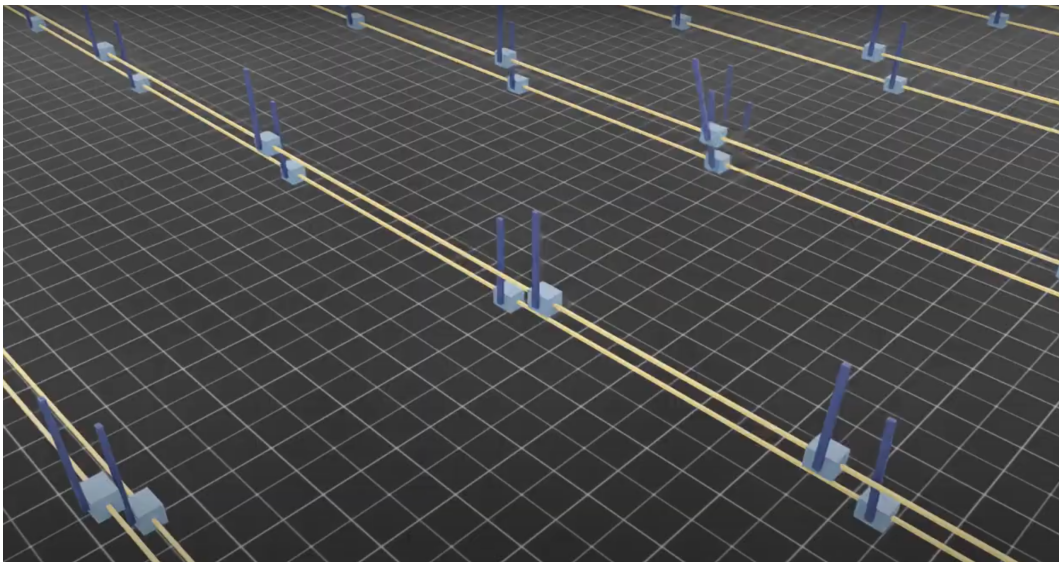

Figure 7: two Cart-pole pendulum configuration RL environment

4. Incorporating these pre-trained higher-level policies during the online phase of a Multi-Agent path finding algorithm, wherein the idea would be to look for a suitable policy whenever a collision is found.
5. Introduce more robot configurations in the same scene.

Acknowledgments

This work is in collaboration with Rishi Veerapaneni who is a PhD student at the SBPL lab advised by Prof. Maxim Likhachev and Prof. Jiaoyang Li. Moving forward we are also collaborating with LeCAR Lab's member Nikhil Sobanbabu who has significant experience of working with the Nvidia Omniverse Framework.

References

[1] Veerapaneni, R., Wang, Q., Ren, K., Jakobsson, A., Li, J., Likhachev, M. (2024). Improving Learnt Local MAPF Policies with Heuristic Search. Proceedings of the International Conference on Automated Planning and Scheduling, 34(1), 597-606. https://doi.org/10.1609/icaps.v34i1.31522

[2] https://doi.org/10.1609/aaai.v26i1.8140scan.pdf

[3] @miscxiong2024mqe, title=MQE: Unleashing the Power of Interaction with Multi-agent Quadruped Environment, author=Ziyan Xiong and Bo Chen and Shiyu Huang and Wei-Wei Tu and Zhaofeng He and Yang Gao, year=2024, eprint=2403.16015, archivePrefix=arXiv, primaryClass=cs.RO

[4] https://isaac-sim.github.io/IsaacLab/main/index.html