Quadruped Robot Navigation: A Classical Planning and RL Approach

Cristiano Battistini, Javier Tapia

April 2, 2025

1 Introduction and Motivation

Autonomous navigation in cluttered environments remains a challenge for legged robots. This project explores motion planning and reinforcement learning (RL) for the Unitree A1 quadruped in PyBullet, aiming to compare classical planning against RL-based policies.

2 Project Structure

The project consists of an environment manager, a motion planner, and a controller:

- **Environment:** Managed by `A1Simulation`, handling robot loading, stepping, and visualization. A `MazeGenerator` generates obstacle-rich scenarios.
- **Path Planning:** Implements RRT and RRT*, encoding robot position and heading, with collision checking via bounding boxes.
- **Controller:** A static stance controller is implemented using MPC and `SciPy.optimize.minimize` to optimize the control inputs over a defined horizon while minimizing a cost function. A gait planner is under development to enable locomotion.

3 Implemented Experiments

RRT and RRT* were tested in various maze configurations. Paths were visualized using debug lines, and validity was confirmed by teleporting the robot along computed paths. The static controller is able to maintain a stance, with motion control in progress.

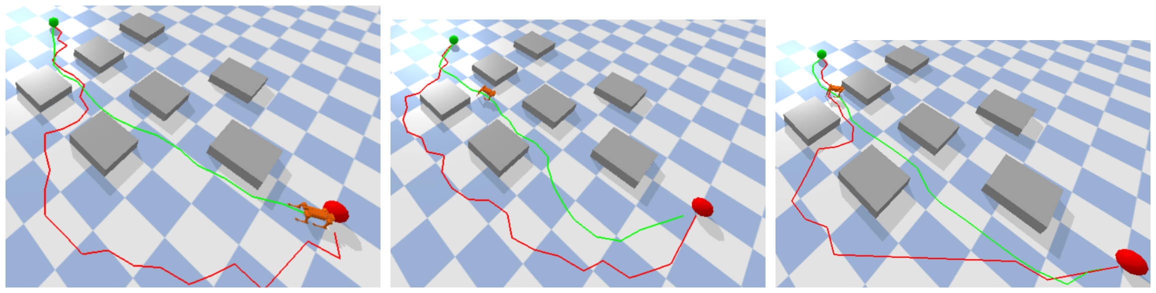
3.1 Comparison of RRT and RRT*

RRT grows a tree randomly, ensuring feasibility but not optimality. RRT* introduces a rewiring step, leading to shorter paths. Results show RRT finds solutions faster, while RRT* yields better paths at increased computation cost.

Table 1: Performance comparison of RRT and RRT*

Metric	**RRT**	**RRT***
Path Length	15.16	12.76
Planning Time (s)	0.017	0.111

It is possible to observe the results visually with the pictures below: the red line is the RTT path, while the green line is the RTT*.



4 Connection to Existing Work

Rapidly-exploring Random Trees (RRT) and its optimal variant RRT* have long been recognized as foundational algorithms in classical motion planning. In our project, we will use these classical planners not just as baselines, but as active tools to generate expert demonstrations and benchmark performance. This will be especially relevant in a hybrid setting where classical algorithms will be combined with RL-based components. MIT's work with MPC on the Cheetah robot, amongst other quadruped control MPC-based projects, have demonstrated how complex yet classical control techniques can achieve exceptional performance in quadruped locomotion. By simplifying MPC and dynamics, this project aims to build on that success, to understand the underlying principles behind these type controllers.

5 Next Steps

Our immediate next step is to replace the current teleportation-based execution with a low-level controller. This will allow the A1 quadruped to follow the planned path with physically realistic motion. We will do this by adapting our existing MPC static controller and a gait planner to make the A1 robot walk through the waypoints generated by the planner.

In parallel, we will begin developing a reinforcement learning policy. The RL agent will learn to reach the goal autonomously, based on simulated sensory inputs such as proprioceptive data. The goal is to train a policy that does not rely on predefined planning but learns an end-to-end behavior from experience.

Once both pipelines are functional, the two approaches will be tested under the same environment conditions and assessed based on success rate, path quality, execution time, and robustness. This comparison will provide insights into the trade-offs between classical and learning-based navigation for quadruped robots in structured environments.