Optimal Path Planning for a Quadruped Robot in Cost-Constrained Grid Terrains using A* Search

Gagan [UG Student, AI + Robotics Enthusiast]

July 10, 2025

Abstract

Quadruped robots designed for autonomous navigation in dynamic and unstructured environments require robust path planning strategies. This report presents a cost-aware path planning framework over a 20×20 grid, where each cell represents varying terrain difficulty or impassable obstacles. We implement and analyze the A* search algorithm, adapt it to cost-constrained environments, and animate the resulting path using Pygame for real-time visualization. The system is ideal for navigation in real-world robot dog projects and can be extended to handle dynamic environments, elevation data, and energy optimization.

Contents

1 Introduction

Quadruped robots such as Boston Dynamics' Spot or MIT's Mini Cheetah are revolutionizing field robotics, especially in terrains where wheeled robots fail. However, their real-world usability relies on intelligent decision-making systems capable of navigating unknown and obstacle-rich environments. This project addresses the path planning component — the brain guiding the feet — enabling the robot to traverse complex terrain efficiently.

2 Problem Statement

We consider a 20×20 2D grid, where:

- Each cell contains a positive integer representing the cost of traversal (e.g., grass, mud, sand).
- Some cells are marked as obstacles with a value of -1, representing impassable regions.
- A start position (x_s, y_s) and a goal position (x_g, y_g) are given.

Our objective is to compute the **least-cost path** from start to goal while avoiding obstacles.

3 Algorithm: A* Search

3.1 Overview

A* (A-Star) is a popular pathfinding algorithm that efficiently finds the shortest path by combining:

- **Uniform Cost Search**: Explores cheapest paths.
- **Heuristic Search**: Guides exploration towards the goal.

3.2 Cost Function

Each node maintains:

- $g(n)$: Actual cost from start to current node n.
- $h(n)$: Estimated cost from n to goal (heuristic).
- $f(n) = g(n) + h(n)$: Total estimated cost of path via n.

3.3 Heuristic Used

We use the **Manhattan Distance**:

$$h(n) = |x_n - x_g| + |y_n - y_g|$$

This is admissible (never overestimates) and consistent.

3.4 A* Pseudocode

```
1. Add start node to the open set with f = h(start).
2. While open set is not empty:
    a. Pick node with lowest f(n).
    b. If goal is reached, return path.
    c. Otherwise, expand neighbors:
        - Ignore if obstacle or already visited.
        - Compute g, h, f for each neighbor.
        - Add neighbor to open set.
3. If goal is unreachable, return failure.
```

4 Implementation Details

4.1 Grid Generation

The terrain is generated as a 20×20 NumPy matrix with:

- Random costs from 1 to 9.
- 60 randomly placed obstacles marked as -1.

4.2 Visualization with Pygame

We animate the robot's traversal using **Pygame**, rendering:

- Obstacles (black)
- Terrain (shades of gray)
- Start (blue), Goal (green)
- Robot's real-time position (yellow)
- Path (red trail)

This creates an engaging simulation of how the quadruped would realistically move through a known map.

5 Applications in Quadruped Robotics

5.1 Terrain-Aware Planning

Quadrupeds must adapt to terrain difficulty to minimize energy consumption and mechanical stress. By using terrain cost maps (from sensors or satellite data), this planner allows:

- Avoiding steep, muddy, or risky regions.
- Selecting low-cost, stable paths.

5.2 Integration in Real Robots

This grid-based A* system is a building block for:

- Real-time onboard planning using LiDAR-generated maps.
- Footstep planners that convert global paths into gait commands.
- Energy-aware gait selection — more costly terrain means slower, careful steps.

6 Extensions and Future Work

- **Dynamic Replanning (D* Lite)**: When terrain changes dynamically.
- **Elevation Maps**: Use 2.5D data to factor in slope/height.
- **Learning Heuristics**: Reinforcement learning-based cost estimators.
- **Path Smoothening**: Make path curved/gait-friendly using splines.

7 Conclusion

We implemented an efficient, cost-aware, and visually intuitive path planner for a quadruped robot using A* search. This framework not only demonstrates sound theoretical underpinnings but also lends itself well to practical robotic navigation. With minor extensions, this system can serve as a core planner for real-world terrain-aware robotics.

Code and Demo

GitHub Repo : `https://github.com/gaganchandra765/robotics`
Demo Video: Included in the repository (via Pygame animation).