D* Lite Algorithm Implementation for 2D Grid-Based Path Planning

James Scott

Kenadi Waymire

California Institute of Technology

ME / EE / CS 234a

Professor Joel Burdick

3/19/25

I. Summary

Project Overview

This project focuses on the implementation and simulation of the D* Lite algorithm, a dynamic path planning algorithm, in a 2D grid-based environment. The D* Lite algorithm is an incremental version of the A* algorithm, designed to efficiently replan paths in environments where obstacles may change or be discovered during execution. This capability is particularly useful for robotics applications where the robot must navigate through unknown or dynamic environments.

Motivation

The motivation behind this project is to explore how the D* Lite algorithm can handle dynamic changes in the environment, such as newly discovered obstacles, and how it can replan paths in real-time. This is crucial for autonomous robots operating in real-world scenarios where the environment is not fully known in advance.

Scope

The scope of the project includes implementing the D* Lite algorithm in Python, simulating the algorithm in a 2D grid environment with varying obstacle configurations, and visualizing the robot's path. The algorithm was tested in three different environments: a maze, a simple environment with no obstacles, and an environment with a few obstacles.

Approach

The approach taken to solve this project involves implementing the D* Lite algorithm from scratch, creating a 2D grid environment with walls and obstacles, and using Matplotlib to visualize the grid, obstacles, and the robot's path in real-time. The algorithm was tested in three environments to evaluate its performance and adaptability to dynamic changes.

II. Details of the Project

Algorithm Implementation

The D* Lite algorithm was implemented in Python, with key components including a Node class to represent each cell in the grid, a priority queue to manage nodes based on their priority, and a *D Lite Class** containing the core logic for path planning. The Robot class simulates the robot's movement through the grid, updating its position as it follows the planned path.

Environment Setup

The environment is a 25x25 grid with walls and obstacles defined as polygons. Three different obstacle configurations were used: a maze environment with multiple obstacles forming a maze-like structure, a simple environment with no obstacles, and an obstacles environment with a few obstacles to test the robot's ability to navigate around them.

Visualization

The visualization was handled using Matplotlib, with grid lines, walls, and obstacles marked on the grid. The robot's path was visualized in real-time, with the original planned path shown in red, the current path in blue, and the actual path traveled in green. As the robot moved and

discovered new obstacles, the grid was updated to reflect the changes, and the path was replanned accordingly. Obstacles and walls were handled and visualized using Shapely polygons.

III. Output of the Project

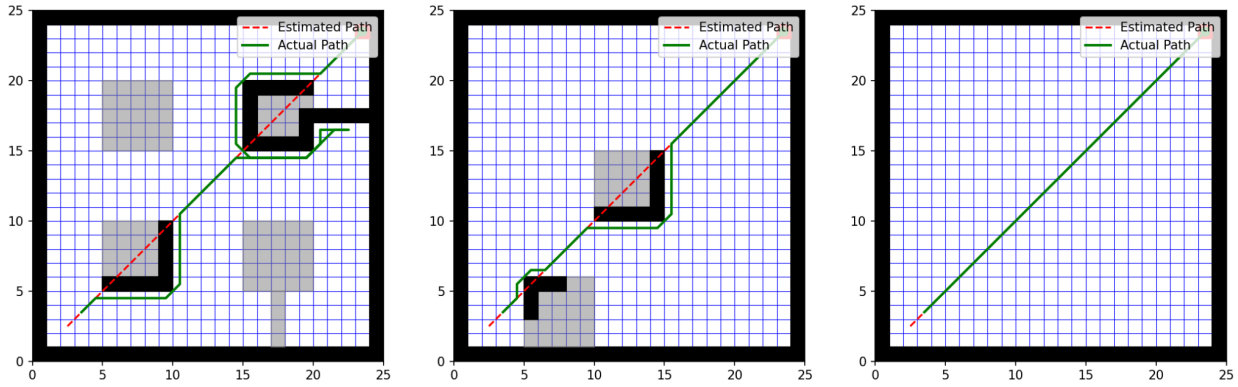
Simulation Results

The algorithm was tested in three different environments, and the results were visualized using Matplotlib. In the maze environment, the robot successfully navigated through the maze, avoiding obstacles and reaching the goal. In the simple environment, the robot followed a straight path from the start to the goal, with no need for replanning. In the obstacles environment, the robot navigated around the obstacles, demonstrating the algorithm's ability to handle dynamic changes in the environment.

Figures

The following figures were generated during the simulation:

1. Maze Environment: An image of the robot navigating through the maze, with the original, current, and actual paths shown.
2. Simple Environment: An image of the robot moving in a simple environment with no obstacles.
3. Obstacles Environment: An image of the robot navigating around a few obstacles.



IV. Conclusion

Summary of Results

The D* Lite algorithm was successfully implemented and tested in three different environments. The algorithm demonstrated its ability to handle dynamic changes in the environment, efficiently replanning paths as new obstacles were discovered. The visualization provided a clear understanding of the robot's movement and the algorithm's decision-making process.

Shortcomings

One limitation of the current implementation is its computational complexity, which can become expensive in large or highly complex environments, especially when many obstacles are present. Currently, this algorithm calculates the entire path planner each time a new obstacle is encountered, which can happen quite frequently. Additionally, the algorithm is currently limited to grid-based environments and may not be directly applicable to continuous or more complex real-world scenarios.

Future Improvements

Future work could focus on optimizing the algorithm to reduce computational complexity, especially in large environments. Additionally, the algorithm could be extended to continuous environments using techniques such as probabilistic roadmaps or rapidly exploring random trees (RRT). Testing the algorithm on a physical robot in a real-world environment would also provide valuable insights into its performance in more complex and dynamic scenarios. One direction of improvement for applicability in real-world scenarios is terrain modifiers; namely, having some attribute for positions on the grid (or positions in a continuous environment) that describe the nature of the cell (such as mud or water) that might affect traversal through that location.

V. Code

The complete code for this project is available on GitHub:
https://github.com/jascott4427/Grid-Based-D-Lite

VI. References

Koenig, S., & Likhachev, M. (2002). D* Lite. *Proceedings of the National Conference on Artificial Intelligence*.

Stentz, A. (1994). Optimal and Efficient Path Planning for Partially-Known Environments. *Proceedings of the International Conference on Robotics and Automation*.