Capstone Project Proposal
Udacity Machine Learning Nanodegree

Paima Marbun
10/12/2021

Robot Motion Planning

Domain Background:

This project is based on Micromouse competition where a small robot should find autonomously the fastest possible path from predetermined starting node to one of the goals located in the center of an unknown maze. The maze is typically made up of a 16x16 grid of nodes with or without walls on its four edges. The robot can explore and discover the walls in the first run to map out the maze and detect the goals. From the learning of the first run the robot is expected to know the fastest path to one of the goals in the maze center and it attempts to reach it the second run.

I have selected this domain as robotics and autonomous-, intelligent-system are to me currently the most exciting, challenging, and fast-growing field.

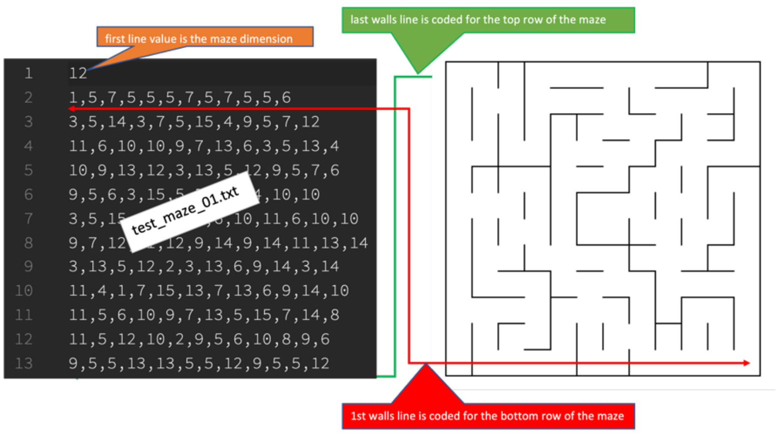
Problem Statement:

The use case should be simulated by applying virtual robot and virtual maze. The virtual robot should find the fastest path from the starting point in the bottom-left corner of the maze and facing upwards to one of the goals in center of the maze. The maze has n x n dimension of squares with minimum n value of 12 and maximum n of 16. Each square or node can have walls on their edges which restrict movement in respective direction. The robot is equipped with three obstacle sensors mounted in the front, left and right side. At any time, robot can move forwards or backwards up to three squares as well rotate clockwise or counterclockwise 90 degree.

The problem of this project is to find an algorithm for the robot as a guide to discover an unknown maze and reaching the goals in fastest time possible. Several algorithms should be investigated to determine the most optimal one to cope with three sample mazes as per the project requirements. To provide more general solution and to reduce biases for future mazes performances against different other samples should be considered too.

Datasets and Inputs:

The only inputs fed into the robot are the maze dimension, start position plus heading, and the four possible goals in the center of a maze. For a test system the entire maze is provided as text file contains comma separated values which outline walls for respective square or node. The four-bit binary value represents four wall edges on a square. Bit value 0 means edge is open and 1 means edge is blocked. The three minimum samples for this project are provided by Udacity. Following scheme should illustrate how to create a new maze based on example test_maze_01.txt.


first line value is the maze dimension
last walls line is coded for the top row of the maze
test_maze_01.txt
1st walls line is coded for the bottom row of the maze

```
12
1,5,7,5,5,5,7,5,7,5,5,6
3,5,14,3,7,5,15,4,9,5,7,12
11,6,10,10,9,7,13,6,3,5,13,4
10,9,13,12,3,13,5,12,9,5,7,6
9,5,6,3,15,5,[illegible],10,10
3,5,1[illegible],10,11,6,10,10
9,7,12,[illegible],12,9,14,9,14,11,13,14
3,13,5,12,2,3,13,6,9,14,3,14
11,4,1,7,15,13,7,13,6,9,14,10
11,5,6,10,9,7,13,5,15,7,14,8
11,5,12,10,2,9,5,6,10,8,9,6
9,5,5,13,13,5,5,12,9,5,5,12
```

Solution Statement:

There are several proven algorithms for path finding available e.g., the popular A*. But these all require some knowledge of the maze to output an optimal path from start to goals. And the best optimal path can certainly be achieved if entire maze is known or 100% discovered during first exploratory run. Thus, the main challenge in this project is to explore the maze as many possible in minimum timesteps to find as many as possible paths to goal so that the second optimization run can achieve optimal path close to the best optimal path popular algorithms would achieve with full knowledge of the maze. Since the robot is allowed to further map out the maze after visiting the goal in the first run two approaches will be performed. The one explores the given maze until maximum coverage and the other one explores only until one of the goals found.

Benchmark Model:

I choose A* algorithm as a benchmark model to set the baseline for the shortest achievable time steps in the second run given full knowledge of the maze. The potential algorithms for the first run should find out a necessary exploration of the maze so that A* algorithm in the second run can achieve optimal path as close as possible to the baseline.

Evaluation Metrics:

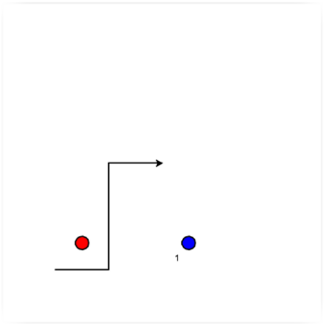
The performance of the robot is calculated as

$$\text{Score} = (\text{run1 timesteps} / 30) + \text{run2 timesteps}$$

The robot can spend only maximum of one thousand timesteps to complete both run1 and run2 for a single maze. Since the weight for run2 portion is 30 times higher than run1 portion the algorithm will prioritize getting better performance on the second run at the expense of timesteps required in run1.

Project Design:

The first step will be to review the provided starter code and get an understanding of the program flow and conclude on what part can be taken for full re-use and where the main function should reside. To get better overview and orientation of the provided mazes graphical visualization need to be plotted. Since A* will be used as my benchmark model this should be first implemented and tested. The function should return best optimal path given maze, start and goal position. For fast reproduction and easy debugging purpose simple maze should be taken as an input. It has a dimension of 6x6 squares as shown below. The start is marked as blue, goal is red and black line should indicate walls.



The goal and start can be placed everywhere in the squares so the function should be able to deal with general use cases and not only tailored for predetermined start as centered goals as this function is also needed for run1. Once this is verified that it works properly then it should be further tested against three provided mazes and verify it manually with eyeball that best optimal path is returned.

For exploratory run1 two approaches will be investigated. The first one is a random movement of robot to visit permissible closest nodes (smallest distance from current node). A priority list containing already seen nodes as result of sensor feedback should guide the robot during the exploration. The node with smallest distance (utilizing max move of the robot) should be put on the top of the list and already visited nodes are marked and should be skipped. This process continues until either the goal is entered, or high maze coverage is reached. The second one is following heuristic value or f-cost in A* term with same termination logic as the first one. A test function should be extended to accommodate various tests and repetition due to random nature of the solution and at the end provide test statistics in a panda dataframe for details analysis. I also plan to have animation support using turtle module to visualize robot movement in addition to logging.

Sources:

Udacity Project Files Link:

https://docs.google.com/document/d/1ZFCH6jS3A5At7_v5IUM5OpAXJYiutFuSIjTzV_E-vdE/pub

Udacity starter code:

https://www.google.com/url?q=https://drive.google.com/open?id%3D0B9Yf01UaIbUgQ2tjRHhKZGlHSzQ&sa=D&source=editors&ust=1633951177995000&usg=AOvVaw20wxDnXySWm08F-g3vMcnc

Micromouse Wikipedia:

https://en.wikipedia.org/wiki/Micromouse

A* Search Algorithm:

https://www.geeksforgeeks.org/a-search-algorithm/

https://en.wikipedia.org/wiki/A*_search_algorithm

A* Pathfinding (E01: algorithm explanation) Video:

https://youtu.be/-L-WgKMFuhE