Jeffrey Zhou
Udacity
Machine Learning Nanodegree
10/19/2017
Capstone Proposal

Robot Motion Planning

Domain Background

Since I don't know yet what my passion is, I have picked one of the few suggested problem areas to explore, Robot Motion Planning. After researching and looking through all the suggested problems, I am most interested in this one because this project is actually inspired from the Micromouse competitions that took place in the late 1970s. I thought it would be fun challenge to analyzing how the Micromouse behaves and finding the solution to the Micromouse problem. The goal of this project is to have a Micromouse traverse an *n x n* grid maze to find the center. The Micromouse must not only find the center but also find the shortest path within the allowed 2 attempts. The Micromouse will start at a different corner or different starting point after every attempt. The first attempt will be for the Micromouse to explore the environment and the second attempt will be the actual test for the Micromouse to find the most optimal path to the center. This will be demonstrated using a virtual robot mouse and virtual test mazes.

Problem Statement

As said above, the goal is for the Micromouse to reach the center of the grid in the shortest amount of time. The maze is a square grid of *n* rows x *n* columns. Since this is a square grid, the number of rows must equal the number of rows. For example, 12x12, 14x14, 16x16, etc. Just like a maze, there will be walls to keep the Micromouse from going outside and collections of paths, as obstacles, for the Micromouse to overcome in order to find the center. The goal will be a 1x1 sub-grid in the center of the main grid. With the 2 attempts that the Micromouse will have to reach its goal, the Micromouse will first use the first attempt to explore every path possible. The second attempt, the Micromouse will use the information that it gained from the first attempt to find the fastest path. So the first attempt can be seen as a testing trial and the second attempt can be seen as a training trial. Each trial, the mouse will have a maximum of 1000 steps. The Micromouse can turn left or right in a 90 degrees fashion and can move forward and backward. The performance of the Micromouse will be based on this metric: *1/30 * time_steps_of_first_run + time_steps_of_second_run.*

Datasets and Inputs

Given the text files, the architecture of the maze will be built based on them. Once a text file, for example *text_maze_01.txt*, is opened, you will see numerous lines of number values. The first line, a single number, will represent the number for *n.* So for example the first line is 12, then the grid will be 12x12. The following lines will represent the maze. The comma-separated number values represent the walls and openings of a square. We can decode the numbers to find number of walls/openings using this rule: "Each number represents a four-bit number that has a

bit value of 0 if an edge is closed (walled) and 1 if an edge is open (no wall); 1s register corresponds with the upwards-facing side, the 2s register the right side, the 4s register the bottom side, and the 8s register the left side." For example, if we see a square that indicates a 15. We will see that 1 + 2 + 4 + 8 = 15. In binary that would be 1111, which means that all the edges are open and there will be no walls. For the first attempt, the Micromouse will gain knowledge of the environment. This consists of the exact starting location, the number of squares in the grid, the number of walls, and all the possible routes that can lead to the center of the grid. The location for the Micromouse or any other squares can be defined as x,y coordinates, {2,10} for example. The movements will a number in the range from -3 to 3(negative for backwards and positive for forward) and the rotations will be -90, 0, and 90.

Solution Statement

Some of the algorithms that I think can help the Micromouse to finding the optimal path to the center. These algorithms are Breadth-First Search, Depth-First Search, and A*. Breadth-First Search is an algorithm that starts at an arbitrary node and searches level by level and uses a queue data structure to stores nodes, or in our case squares, that will be explored. Depth-First Search is an algorithm that starts at an arbitrary node and that searches as far as possible along each branch and uses a stack data structure to store nodes that will be explored. A* uses a greedy search and finds the least-cost path from an initial node to the goal node and uses a priority queue data structure. All of these algorithms work because they will find a path to the goal, in different ways. The idea is to find the best path to the goal, so I will have to try all of these algorithms to see which one works best.

I have already discussed about the 2 attempts that the Micromouse would have to go through for each test. For the first attempt, or the exploration phase, I plan to use Depth-First Search to search all branches of the maze. The solution for the first attempt is to gain as much knowledge as possible of the grid. The Micromouse would have knowledge of all the possible routes to the center now. The solution for the second attempt is to use the information gained in the first attempt and reach the center in the shortest amount of time. My plan for this is to use Breadth-First Search or A* to find the optimal path.

Benchmark Model

The benchmark model will be the Flood-fill algorithm. This algorithm visits every point, given the starting point and the destination, in a maze. Because this algorithm visits every point in the maze, this will not be the most optimal algorithm for this problem. However, this algorithm does eventually find the destination. So this will make a good benchmark for me to compare results against once I have a successful solution. Once I have a perfect solution, I will test my algorithm on a certain test data and compare the results with the results of the Flood-fill algorithm, executed with the same test data.

Evaluation Metrics

Both the first and the second attempts will play a role in the evaluation metric. The final score will depend on the steps taken in both the attempts. As said before the performance metric will be based on this formula: *1/30 * time_steps_of_first_run + time_steps_of_second_run.* The score consist of 1/30 times the number of steps that the Micromouse took in the first attempt plus the number of steps that it took in the second attempt. For example, if the Micromouse took 500 steps in the first attempt and then

350 on the second attempt, the overall steps will be 850. Divide that by 30 and we have our final score: ~28.33. This would be most simple and easy way to evaluate the performance of the Micromouse. The goal is to get the lowest score possible. If we run more tests(2 attempts) and the score cannot get lower than 28.33, then 28.33would be the optimal score. Evaluation would be done on the final score and if the Micromouse has improved on the steps for reaching the goal on the second attempt.

Project Design

To find a solution to this challenge, I plan to look through all the provided files from the *AI_Startercode* folder: *maze.py, robot.py, showmaze.py, tester.py, test_maze_01.txt, test_maze_02.txt, and test_maze_03.txt*. I will examine what each of these files entail, how they work with each other, and how to run these files together. Once I have done that, I will look into how to set up the maze to get a visualization of how each maze looks like. Since it was stated that only robot.py can be modified, I will implement the logic for the Micromouse to reach the goal in there. To do this, some of the algorithms that I will look into using are Breadth-First Search, Depth-First Search, and A* Search. I will try Depth-First Search on the first attempt, the exploration phase, and then Breadth-First Search or A* on the second attempt, the testing phase. Once I have one of these algorithms to implement, I will then run multiple tests to find the optimal score for the Micromouse. If there are any adjustments that that can be made to optimize the process of reaching the goal, I will do so and retest. I will document all the trials, process of getting the best implementation, and the performance scores. I will compare performance scores with my model with the performance scores of the benchmark model, the Flood-fill algorithm.

Sources

Robot Motion Planning Capstone Project
https://docs.google.com/document/d/1ZFCH6jS3A5At7_v5IUM5OpAXJYiutFuSIjTzV_E-vdE/pub

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

Labyrinth Algorithms
http://bryukh.com/labyrinth-algorithms/