Machine Learning Engineer Nanodegree

Capstone Proposal - Robot Motion Planning

Tien Pham
July 4, 2018

Plot and Navigate a Virtual Maze

Domain Background

This project was inspired from Micromouse competitions originated in the 1970s, wherein a robot mouse is given the task of running multiple time in a given maze and plotting the best path from a corner of the maze to its center. The robot mouse discovers the maze during the first run and using the planned best path it has previously learned to reach the center of the maze in subsequent runs.

In this project, a simplified model of the world is provided along with specifications for the maze and robot. The goal of this project is to obtain the fastest times possible in a series of test mazes by controlling the navigation of a virtual robot in a virtual maze.

Problem Statement

On any given maze, the robot must complete two runs. The starting point and the goal room are "known" to the mouse before it starts. The starting point is always in a left corner and is always surrounded by three walls. The goal room is four center blocks of the maze. In the first run, the robot is allowed to freely roam, explore, and analyze the maze to determine the best path plans to reach the center of the maze. To finish the first run, it must enter the goal room during its exploration. After entering the goal room, the robot is free to continue exploring the maze and may choose to end its exploration at any time. The robot is then moved back to the starting position and orientation for its second run. Its objective now is to go from the start position to the goal room in the fastest time possible. The robot's score for the maze is equal to the number of time steps required to execute the second run, plus one-thirtieth the number of time steps needed to execute the first run.

A maximum of one thousand time steps is allowed to complete both runs for a single maze. On each time step of the simulation, the robot may choose to rotate clockwise or counterclockwise ninety degrees, then move forwards or backwards a distance of up to three units.

Datasets and Inputs

The information of mazes is provided in a text file. On the first line of the text file is a number describing the number of squares on each dimension of the maze n, n is even and in a range [12, 16]. On the following n lines, there will be n comma-delimited numbers describing which edges of the square are open to movement. Each number represents a four-bit binary that corresponds four-side, in this order: up, right, down, left. A bit value of 1 if there is no wall, and a bit value of 0 if an edge is closed (walled).

The robot has three obstacle sensors mounted at the front, right and left which detect the number of open squares in the direction of the sensor. These sensors detect the number of open squares in the direction of the corresponding sensor, then store in the form of a list of three numbers.

A 2-digit pair will define the location of the robot mouse as the coordinate [x, y]. The robot decides next location with two values indicating the robot's rotation and movement on that timestep. Rotation is an integer taking one of three values: -90, 90, or 0, indicating a counterclockwise, clockwise, or no rotation. Movement is also an integer in the range [-3, 3].

Solution Statement

Each trial run tries to solve a different problem. During the first trial, the main problem to solve is a detailed knowledge of the maze layout and possible paths reaching to the goal. The robot will explore the maze in the first run where it will learn and map the structure of the maze, and find all possible paths to the goal. The target solution in the second run would be reaching the goal following an optimal path in the fastest time or precisely in a minimum number of steps.

The number of total movements taken along with the whole time of the maze navigation is continuously assessed. After the exploration trial run, the knowledge of the number of steps needed to reach the goal for all possible paths should be stored, and the chosen route after optimization round should be the shortest path with the fewest steps. We can perform multiple runs to test if our robot has learned the optimal policy and the path for each run should remain the same if it is indeed the best path to the goal.

Benchmark Model

The most basic form of benchmark performance is a maximum of 1000 steps to complete both runs. The robot is allowed to roam freely to navigate the maze to find the goal, also continue to explore after hitting the target less than a thousand time steps in the exploratory run. Since the robot has gained the knowledge of the maze in the exploratory run, it should optimize route - which is the minimum amount of steps required to get from the starting location to the goal area.

Then it would reach the goal in the second run in shortest time following an optimal path. The performance score for the benchmark model will be measured by the following metric:

Benchmark Score = [no. of steps in trial 2] + [no. of steps in trial 1 / 30]

Evaluation Metrics

As discussed in previous sections, the evaluation metric to quantify the performance of both the benchmark model and the solution model is impacted by both exploration trial and optimization trial. However, the optimization trial having more weight will affect the score significantly more than the exploration trial. Our goal is to minimize this score, which would be the result of an optimal trial. During the first run, we can consider exploring the maze as much as possible and avoiding multiple visits to the cells it has visited already to minimize the cost for the second run.

Project Design

The robot is given the problem of solving a randomized maze in the least amount of actions. With given the maze dimension initially, the robot is allowed to explore the maze in the first run. The second run, the robot should utilize the optimal path to take the minimum actions necessary to reach the goal. Then, we can receive the best (lowest) score possible after the second run. The following theoretical workflow will be taken:

1. The initialization of the robot
    a. Take in the maze dimension in the text file
    b. Determine the goal location from the maze dimension
2. The first run of the robot:

    The path planning algorithm will be applied for taking in the recorded maze and decide the optimal path for the robot.

    a. Explore the environment and record the maze during exploration

        The next movement could be selecting actions based on the algorithm. The location of walls or dead ends could be stored.

    b. Reach the goal of the maze

        The robot can stop exploration or continue exploration until the environment is wholly discovered.
3. The second run of the robot
    a. Follow the path planned by the algorithm
    b. b. Stop once the goal is reached

In order to get best score, I will test the performance of some path planning and maze solving algorithms by comparing benchmark scores, also make any needed improvements and select the one which performs the best. The following list is several algorithms can be applied:

- *A* algorithm*
- *Dijkstra’s algorithm*
- *Flood Fill algorithm*
- *Breadth First Search algorithm*
- *Depth First Search algorithm*