|  |  |
| --- | --- |
| **Capstone Project** | Muhammad Zeeshan Karamat |
| **Machine Learning Engineer Nanodegree** | July 13, 16 |

Plot and Navigate a Virtual Maze

# Definition

**Project Overview:**

A robotic mouse in an unknown virtual maze, is programmed to firstly plot the unknown environment and then find the optimal path from start location to the center of the virtual maze. [This video](https://www.google.com/url?q=https://www.youtube.com/watch?v%3D0JCsRpcrk3s&sa=D&ust=1468346325966000&usg=AFQjCNFDNy5fisGIOs7mRSo8o8Lu-e9FXw) (Youtube) is an example of a Micromouse competition.

**Problem Statement:**

The robotic mouse has two runs for this project. In the first run, the robot will explore and map the unknown maze surface and store its maps in its memory with certain time limit. It will continue exploring the space no matter it has reached its goal i.e. center of the maze. Then, in the second run the robotic mouse will exploit what it has learnt in the first run and come up with optimal strategy from start location to the end location and will plan its route in a optimal way.

Some initial code is already provided along with specifications for the robotic mouse environment and testing. I have programmed the robotic mouse to first plot and then optimally navigate through that virtual environment. For the first run the robot will explore as much surface as it can in minimal steps and for then for the second run, it will use artificial intelligence search methods to come up with optimal route to reach center of maze.

**Metrics:**

For scoring the performance of the robotic mouse, combination number of steps taken by the robotic mouse in the first run plus number of steps for the second run. For adding more weight to the second run, the first run has been divided by thirty and then added to the second run steps to make the final score. Total number of steps taken by the robotic mouse are limited to 1000.

# Analysis

**Data Exploration:**

The shape of virtual maze is n\*n square with goal in the center of it, which is 2\*2 grid. For each cell of the grid, it might has walls in the left, right, up or down side of the cell, which will prohibit the robot motion though them. Robot will always start from the bottom left corner of the virtual maze where it can only all walls closed except for the up one.

The robot has three distance sensor mounted on it, that will measure the robot distance from its current location to the walls on their respective sides. It is assumed that the sensor readings are accurate and free from any noise. Below are these three sensors,

* Left distance measurement sensor
* Front distance measurement sensor
* Right distance measurement sensor.

For the robot motion, it can rotate either clockwise or anticlockwise direction or move straight and then move forward or backward. It is assumed the robotic motion is perfect with no probability of moving into the wrong direction. If the robot hits the wall while moving, it will stay where it is after that time step and there is not penalty for this collision. Sensors values are available after each movement of the robot.

Rotation of the robot is expressed in angles with possible values -90(counter clockwise), 0 or 90(clockwise) degrees. And the movement is expresses as an integer with possible values form -3 to 3 inclusive, which indicates the number of steps taken in the respective direction.

Mazes are provided as text files with first lines describing the dimension of the square maze and the subsequent lines indicates the allowed location for each column delimited by commas. 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); the 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, the number 10 means that a square is open on the left and right, with walls on top and bottom (0\*1 + 1\*2 + 0\*4 + 1\*8 = 10), as shown below.

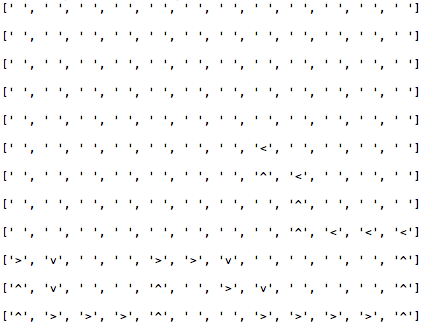
****

These maze files are only used for testing the search approach independently and for testing the robot performance, but the robot will plot the unknown virtual environment by itself, rather than from these files.

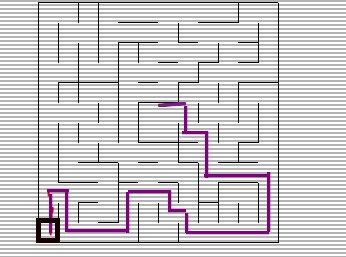
For example for the first maze provided, its visualization will be covered in the next section. Some of the interesting points that can be exploited for obtaining better score are avoiding dead-ends, avoiding loops, using initial heuristics to guide the robot towards the center of maze, and preferring the front motion as compared to turning. They are further discussed in the next section.

**Exploratory Visualization:**

For the first maze, it is shown below with optimal route from bottom left corner to the center of the maze output by the program is shown below, where the arrows indicate the robot moving direction.

****

and the equivalent path plotted on the actual maze is shown below, with thick black rectangle on the bottom letf corner indicating the start position of the robot and violet color lines indicating the robot trajectory.

****

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**

**Problem Statement:**

**The robotic**