Capstone Proposal
Plot and Navigate a Virtual Maze

Taylor Mitchell
October 24th 2017

Domain Background

This project is based off the Micromouse competition, where teams design a robot to navigate a maze from the corner to the center without any other knowledge of the structure of the maze aside from what it can see around it while inside.

The domain is autonomous robotics, since the robot will have to make decisions on how to navigate through the maze without any human input. This is an area I find incredibly interesting! I've always been fascinated by the existing contrast between the nimble, expressive, and dynamic motions of animals compared to the constrained, rigid, and fragile motions of present day robots. And an animal's ability to navigate novel terrain is still far beyond that of today's best robots. It just show how much room for improvement there is! I'm excited by how much progress is going on in the field right now and am looking forward to the breakthroughs in the years ahead.

Problem Statement

An agent must be programmed which can autonomously navigate through a variety of unseen mazes. During the first run through the maze, the agent starts at the bottom left corner of the maze and must navigate through until it reaches the goal room. Once the agent chooses to finish the first run, it is relocated back to the start of the maze, in its original orientation. The agent then begins its second run, using the information it gained from the first run to get to the goal room in the least time steps possible.

Datasets and Inputs

The maze which the agent will be navigating is a 12x12, 14x14, or 16x16 grid. The mazes used for testing were provided by Udacity. The starting square is the bottom left corner of the maze, and the goal is a 2x2 room in the center. The agent is considered at the center of the square it's located, and is facing one of four orthogonal directions (up, down, left, right). It is equipped with sensors which allow it to see 3 squares in front, to the left, and to the right of itself. Each time step, the agent first receives information from its sensors. Based on this information, the agent can rotate 90 degrees clockwise or counterclockwise, or not rotate, and then move up to 3 squares forward or backwards. Once the agent moves, this concludes the time step.

Solution Statement

The proposed solution is an agent which is biased towards moving to squares which are closer to the center, while avoiding squares that has already explored. It'll do so by initially assigning a utility value to each of the squares, where the squares in the center have the highest utility, and the utilities decrease as you move out towards the edges. When the agent moves to a square, the utility value of that square is decreased, so the agent is less likely to choose this square in the future. The agent chooses which of the available squares to move to by selecting the one with the largest utility. Which squares can be moved to from which will be saved while moving through the maze on the first run, and the A* search algorithm will be used to find the shortest path in the explored maze for the second run.

Benchmark Model

I'll create a simple agent to navigate the maze, and use its results as a benchmark for the final agent design. The agent will choose randomly between the squares available to it which haven't been explored yet. To obtain the average performance of the benchmark agent, I'll conduct multiple trials of each maze, and the average results will be used for comparison.

Evaluation Metric

Three metrics will be used to compare the benchmark agent to the final agent. The agents will be compared based on these metrics for each of the three provided test mazes.

1. Average number of time steps to complete the first run.
2. Average number of time steps to complete the second run.
3. Average Score. Where,

    Score = (steps in first run)/30 + (steps in second run)

Project Design

The first step will be to create the benchmark agent, and have it successfully navigate the maze. This includes creating functionality to: process the sensor inputs, select squares to move to, determine the rotation and movement required to get from the current location to the selected location, save the explored sections of the maze, and determine the shortest path from start to finish based on the information obtained during the first run. Once the first agent is created, I'll create another agent which is biased towards the center of the maze, as well as adding on any additional functionality to deal with the inevitable problems which will present themselves. I'll also create an agent which continues exploring the maze during the first run after the goal is found. It will probably take some trial and error to determine the correct trade off between spending time exploring the maze during the first run, and minimizing time steps spent. I'll use the metrics mentioned above to iteratively approve upon the agents, and eventually select a final design.