MLND Capstone Project Proposal

Robot Motion Planning in a Maze

Qingyuan Ji

November 8th, 2017

Domain Background

The micromouse competition is an annual event in the robotics industry where a small robot mice is asked to navigate through a small sized maze (Misha et al, 2008). The maze is usually a space consisting of 16 by 16 cells, where each cell is about 180 mm square. Walls exist between cells so that a cell is only allowed to be entered and leave in limited directions. Figure 1 show an example of a maze.

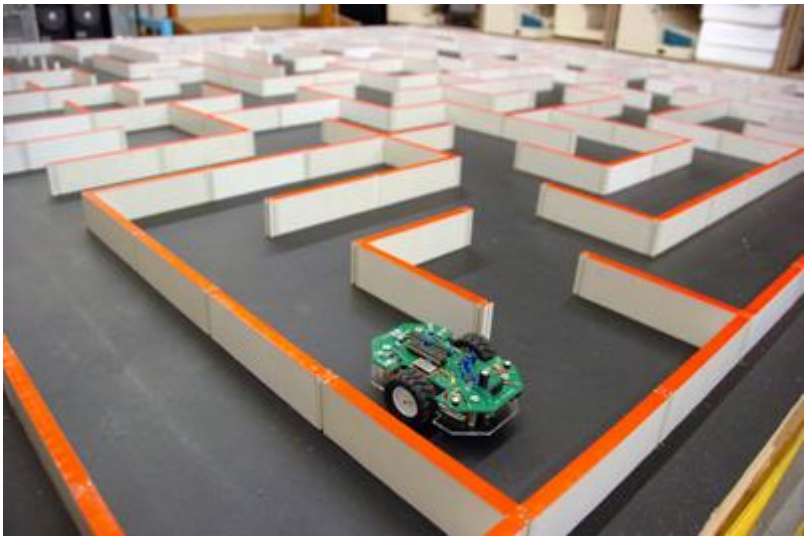

Figure 1. A maze example for the micromouse competition (source: Google Images).

A robot mouse should be completely autonomous and heuristic to be able to find the correct path from the predetermined starting point to the goal (normally the central of the maze) without external help or command. The key task for the robot mouse is to roam around the maze with no prior knowledge, detect the location of walls blocking its way, and keep track of where it has explored and where the goal is. Finally, the robot should be able to somehow map the maze and find the optimal path leading to the goal, from the starting point.

This competition starts from 1970s, and is still being hold every year nowadays. Despite being around 40 years old, its importance in the field of robotics is still unparalleled, as it requires complete analysis of unknown space and proper path planning and decision making. Different algorithm such as Dijkstra (Gupta and Segel, 2014) and A* search (Ambeskar et al, 2014) haven been extensively applied and modified to tackle this issue. This project is inspired by the micromouse competition, where I plan to implement the simplified mazes and robot mouse digitally to find a model allows the robot mouse to reach the goal in the fastest way.

Problem Statement

The problem I want to solve is that, given mazes of different layouts and different models, let the robot mouse beat the mazes as fast as possible. For example, figure 2 shows a specific layout of maze that will be used in the project.

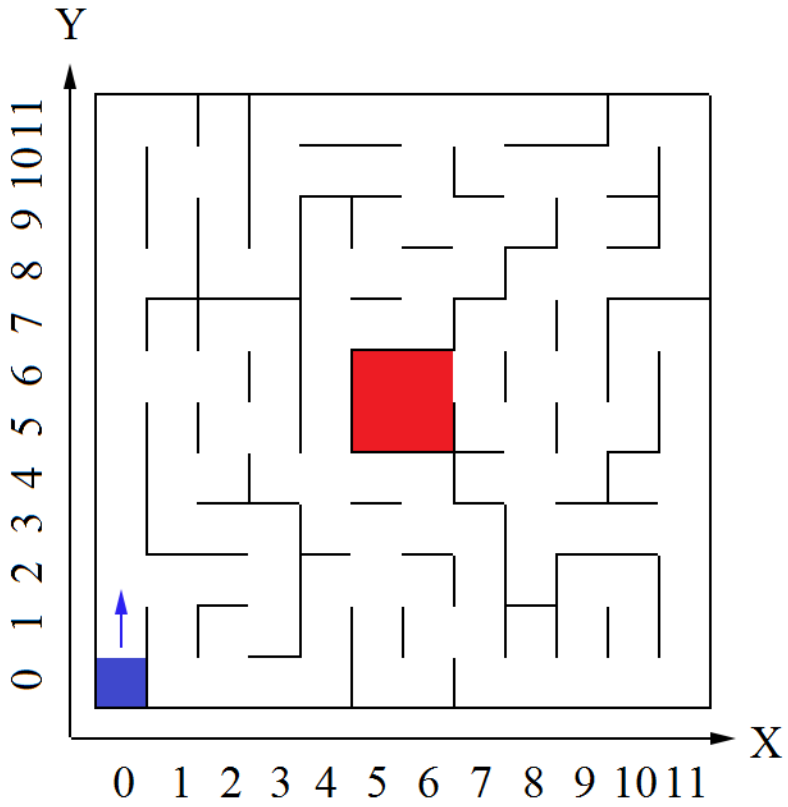

Figure 2. A sample maze (12×12) that will be used in the project.

In figure 2 is a 12×12 maze, which means this maze consists of 144 grid cells. Since the maze is a 2D space, it can be described by a 2D coordinate system, and each cell has its own coordinate, for example, coordinate of cell on the top right is (11, 11). Walls exist in the maze to block the entry into a cell from a specific position. For example, it is impossible to travel from the cell (1, 0) to (0, 0), because there is a wall between them.

The rule for the robot mouse is as follows -

The robot mouse is given two runs totally to solve a specific maze. At the start of each runs, it will always be placed at the bottom left cell (the blue cell) whose coordinate is (0, 0). Moreover, the mouse will initially face north (the blue arrow). The mouse is equipped with sensors on the forward, left, and right side to know if currently there are wall on these sides. For example, at the starting point, the mouse detects walls on the left and right sides, but no wall on the forward side.

In the 1st run, the mouse knows nothing about the maze and needs to start exploring it. A mouse is allowed to take a step from its previous position each time. A step consists of two sequential actions: **rotation and movement**. A rotation can be clockwise 90 degrees, counter-clockwise 90 degrees or no rotation. A movement can be moving forward or backward for 0, 1, 2 or 3 units, where one unit is the distance to move into the adjacent cell. **Note this simplified maze world is discrete, and rotation cannot be things like 45 degrees clockwise or counter-clock**

wise, and the movement cannot be 1.5 or 2.5 units. For example, in figure 2, from its starting location, a valid step for this mouse is to not rotate at all and then move forward for 2 units, where it will end in the cell (0, 2). The goal is depicted in the red areas (consisting of 4 cells). In the 1st run, the mouse is given a limited steps to move from the start location, to explore the maze and try to reach the goal. If the mouse reaches the goal and still has remaining steps, it can continue explore the maze freely.

In the 2nd run, the mouse still starts from the same cell, but this time, it will try to reach the goal as fast as possible, based on knowledge it receives in the first run.

The performance of both runs (the steps used in both runs) will be used to judge how fast a mouse solves a maze.

Datasets and Inputs

The datasets for this project is quite straightforward, which are mazes of different layout. In particular, first there will be three mazes of different sizes (12×12, 14×14 and 16×16), to test and judge different models. Finally, I will construct a more complex and difficult maze for exploring the transferability of the different models. Mazes will be designed based on different characteristics (for example some may have lots of dead ends, while others have many loops).

The inputs will be particular models (or more preciously speaking, the algorithm) applied on the robot mouse. For a specific maze, the robot mouse will be given two runs to solve it. The 1st run is actually an exploration process and the 2nd run is actually a shortest path problem. Therefore, it seems to be a good idea to deal with this two steps separately, using different models (please see the solution statement section).

Solution Statement

As mentioned in the last section, 1st run and 2nd run needs to be dealt with separately, since they are different problems. So different algorithms will be applied in these two runs, and a final model should be a combination of these two algorithms.

For the 1st run, we want the robot mouse to be able to explore the maze as much as possible (within step limit), and also reach the goal (very important, otherwise, 2nd run will fail). The key thing here is to let the robot mouse not waste much time into dead ends or loops and try to mark whether this is the position it has visited. Therefore, an algorithm can be developed based on this criteria, or more preciously speaking, it is actually a policy to tell the mouse what it should do given the current location and its progress in exploring the maze. Key issues to be considered here will be how to let the mouse avoid dead ends or loops, etc.

For the 2nd run, it will be a little easier. If the mouse finish the 1st run, it will be able to map the maze, and actually mathematically speaking, the maze can be transformed into a graph consisting of edges and nodes. Node can be regarded as a cell here, and edge can be regarded as allowed traffic between two nodes (cells). Therefore in this run, I will potentially using graph related algorithm such as Dijkstra shortest path algorithm and the A* algorithm.

The combination of the algorithms used in these two runs will be one model in this case.

Benchmark Model

In my opinion, the benchmark model should focus on the 1st run. Since in 2nd run, we just transform the maze into a graph and apply shortest path algorithm, which is straightforward. The knowledge of the maze the robot mouse received in the 1st run, will be critical to whether or not a shortest path found in the 2nd run is actually "shortest".

The benchmark model of the 1st run can be that, the robot mouse doesn't care about potential dead ends or loops, and it is allowed to travel freely in the maze, with no preference of the next step (as long as its movement is still restricted by walls). The is likely to be a bad model, since the mouse might waste lots of steps but still only explores a small part of the maze (might even not be able to reach the goal), and it is quite bad for its 2nd run. However, this model can be useful to judge how good our actual solutions are.

Evaluation Metrics

The metrics for this project is quite simple, which is the one thirtieth (1/30) of the 1st run steps plus the 2nd run steps. A good model should have this result as small as possible. The reason for doing this is that we want to use steps in both runs, and we actually add a weight here. That is, if the robot mouse spends quite a long time to explore the maze in the 1st run, but is really fast in the 2nd run, it is still considered to be a good model. As long as long time exploration pays off, it is worthwhile.

Project Design

First, the robot mouse is given three different mazes (12×12, 14×14, and 16×16), and we applied different models on the robot mouse. For example, a model can be that, "in the 1st run, let the mouse to explore the maze freely but try to mark potential loops and avoid get into it, and in the 2nd run, use Dijkstra algorithm to resolve shortest path". Different combination of 1st and 2nd run algorithm will result in different models for each maze. Use the metrics, we can judge the performance of different models in a given maze, and analyse its strength and weakness.

Then, I will customise a new maze of different layouts to test the transferability of good models, and make a summary and potential improvement finally.

Reference

Ambeskar, A., Turkar, V., Bondre, A., & Gosavi, H. (2016, August). Path finding robot using image processing. In *Inventive Computation Technologies (ICICT), International Conference on*(Vol. 3, pp. 1-6). IEEE.

Gupta, B., & Sehgal, S. (2014, September). Survey on Techniques used in Autonomous Maze Solving Robot. In *Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference*- (pp. 323-328). IEEE.

Mishra, S., & Bande, P. (2008, November). Maze solving algorithms for micro mouse. In *Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on* (pp. 86-93). IEEE.