Capstone Project of Machine Learning Engineer Nanodegree: Plot and Navigate a Virtual Maze

Yi-Yuan Chiang

February 4, 2017

1 Definition

There are three subsections in this section. They are: *Project Overview*, *Problem Statement*, and *Metrics*.

1.1 Project Overview

The purpose of this project is to design a virtual robot to navigate a given maze. The task of the robot is to plot a path from a corner of the maze to the center. Two runs are allowed. In the first run, the robot has to find the goal and the best path from corner to center. In the second run, the robot attempts to reach the center in the fastest time possible, using what it has learned in the first run. This rule is inspired from the APEC Micromouse Contest [1].

1.2 Problem Statement

As mentioned in the previous section, two runs are allowed for the robot. In the first run, the robot is allowed to freely roam the maze to build a map of the maze. It must enter the goal room at some point during its exploration, but is free to continue exploring the maze after finding the goal. After entering the goal room, the robot 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.

1.3 Metrics

The robots 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 required to execute the first run. A maximum of one thousand time steps is allotted to complete both runs for a single maze.

2 Analysis

There are three subsections in this section. They are: *Data Exploration and Visualization*, *Algorithms and Techniques*, and *Benchmark*.

2.1 Data Exploration and Visualization

(Use the robot specifications section to discuss how the robot will interpret and explore its environment. Additionally, one of the three mazes provided should be discussed in some detail, such as some interesting structural observations and one possible solution you have found to the goal (in number of steps). Try to aim for an optimal path, if possible!)

The given mazes are specified and coded in the files with text format. On the first line of the text file is a number, n, describing the number of squares on each dimension of the maze.

On the following n lines are n comma-delimited four-bit coding numbers describing which edges of the square are open to movement. The coding method is described as below. 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.

Figure 1 is a portion of an example four-bit coding maze, in the bottom left corner, the number 1 means the upper edge of this square is open, 6 means both the right edge and the bottom edge are open.

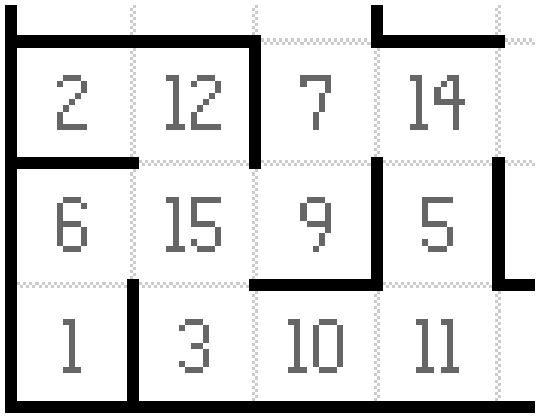
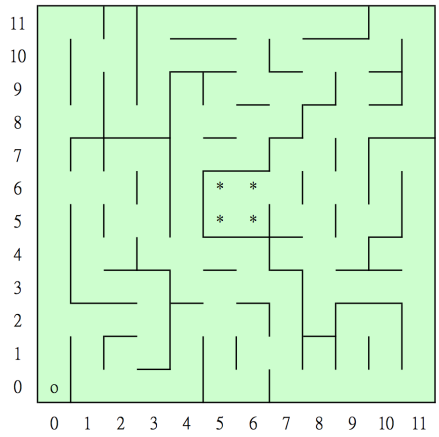


Figure 1: A portion of an example four-bit coding maze.

From this coding method, we are easily to translate the given four-bit coding maze into human understandable mazes specified by walls. Figure 2a is the four-bit coding edition of Maze01; and Figure 2b is the normal one.

```
12
1,5,7,5,5,5,7,5,7,5,5,6
3,5,14,3,7,5,15,4,9,5,7,12
11,6,10,10,9,7,13,6,3,5,13,4
10,9,13,12,3,13,5,12,9,5,7,6
9,5,6,3,15,5,5,7,7,4,10,10
3,5,15,14,10,3,6,10,11,6,10,10
9,7,12,11,12,9,14,9,14,11,13,14
3,13,5,12,2,3,13,6,9,14,3,14
11,4,1,7,15,13,7,13,6,9,14,10
11,5,6,10,9,7,13,5,15,7,14,8
11,5,12,10,2,9,5,6,10,8,9,6
9,5,5,13,13,5,5,12,9,5,5,12
```

(a) Four-bit coding edition of Maze01. (The given format)



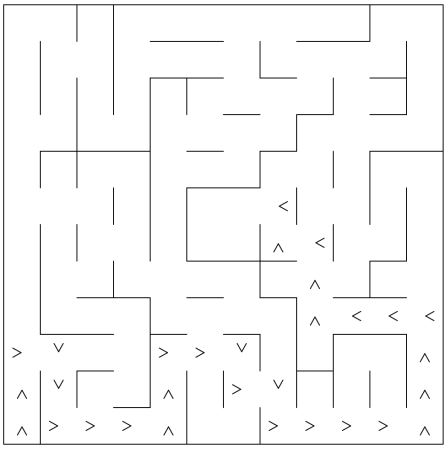
(b) Normal edition of Maze01. The start is labeled as o and the goal is labels as *.

Figure 2: The four-bit coding representation and the normal wall representation of Maze01.

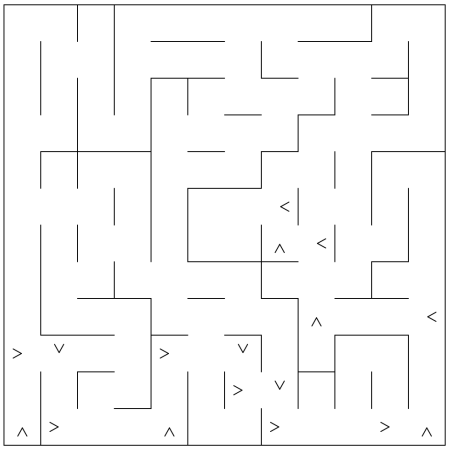
As for robot, it has three obstacle sensors, mounted on the front of the robot, its right side, and its left side. Obstacle sensors detect the number of open squares in the direction of the sensor; for example, in its starting position, the robots left and right sensors will state that there are no open squares in those directions and at least one square towards its front. 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. It is assumed that the robots turning and movement is perfect. If the robot tries to move into a wall, the robot stays where it is. After movement, one time step has passed, and the sensors return readings for the open squares in the robots new location and/or orientation to start the next time unit.

After understand the sensing method of robot, lets take a look at maze01 more detail. The start location is $(0, 0)$, and the goal is a square with four positions, they are $(5, 5)$, $(5, 6)$, $(6, 5)$, $(6, 6)$. When navigating such a maze, a robot may encounters at least two kinds of hazards. I named them as: *dead-end hazard* and *loop hazard*. When a robot encounters a dead-end hazard, it has no way for any direction except moves backwards. For example, when the robot get into the position $(1, 7)$, $(4, 9)$, $(8, 1)$, or $(10, 9)$, it encounters a dead-end hazard. The other hazard is loop hazard, which is caused by isolated walls. For example, the right-hand-side wall of position $(0, 9)$ and $(0, 10)$ may causes loop hazard. Our robots should avoid these hazards in the second run.

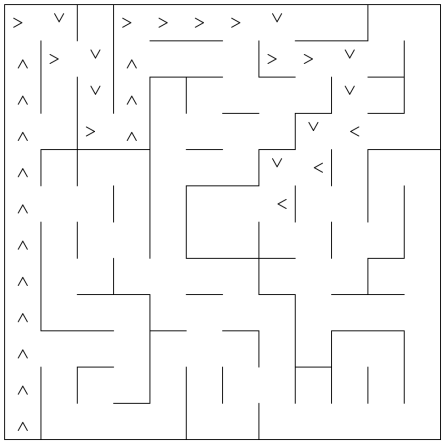
There are many paths for a robot to reach the goal. I just list three of them as shown in Figure 3a, 3c, 3e. Because the robot can move up to three units, the moving steps can be reduced. When the steps of a path are reduced, I call this path as an optimized path (Figure 3b, 3d, 3f).



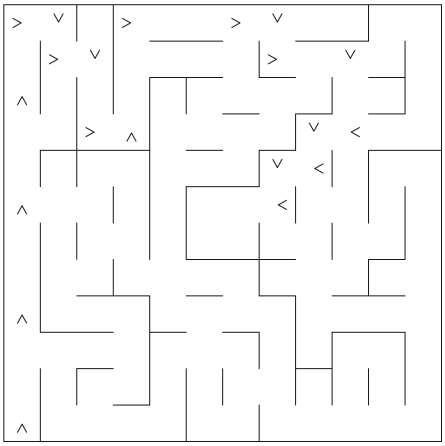
(a) Maze01: path-1 (30 steps)



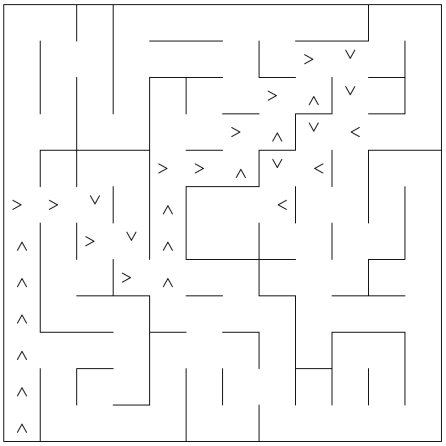
(b) Maze01: path-1 optimized (17 steps)



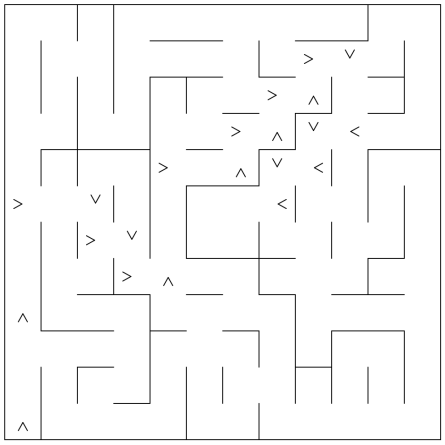
(c) Maze01: path-2 (34 steps)



(d) Maze01: path-2 optimized (20 steps)



(e) Maze01: path-3 (30 steps)



(f) Maze01: path-3 optimized (21 steps)

Figure 3: Example Paths of Maze01

The shortest path of maze01 is path1 (Figure 3a); it takes 30 steps. After optimization, only 17 steps are required (Figure 3b).

2.2 Algorithms and Techniques

2.3 Benchmark

(You will need to decide what you feel is a reasonable benchmark score that you can compare your robots results on. Consider the visualization and data exploration above: If you are allowed one thousand time steps for exploring (run 1) and testing (run 2), and given the metric defined in the project, what is a reasonable score you might expect? There is no right or wrong answer here, but this will help with your discussion of solutions later on in the project.)

3 Methodology

3.1 Data Preprocessing

(Because there is no data preprocessing needed in this project (the sensor specification and environment designs are provided to you), be sure to mention that no data preprocessing was necessary and why this is true.)

3.2 Implementation

3.3 Refinement

4 Results

4.1 Model Evaluation and Validation

4.2 Justification

5 Conclusions

5.1 Free-Form Visualization

(Use this section to come up with your own maze. Your maze should have the same dimensions (12x12, 14x14, or 16x16) and have the goal and starting positions in the same locations as the three example mazes (you can use test_maze_01.txt as a template). Try to make a design that you feel may either reflect the robustness of your robots algorithm, or amplify a potential issue with the approach you used in your robot implementation. Provide a small discussion of the maze as well.)

5.2 Reflection

5.3 Improvement

(Consider if the scenario took place in a continuous domain. For example, each square has a unit length, walls are 0.1 units thick, and the robot is a circle of diameter 0.4 units. What modifications might be necessary to your robots code to handle the added complexity? Are there types of mazes in the continuous domain that could not be solved in the discrete domain? If you have ideas for other extensions to the current project, describe and discuss them here.)

References

[1] *APEC Micromouse Contest Rules 2015*, http://www.apec-conf.org/wp-content/uploads/2013/10/APEC_2015_Micromouse_Contest_Rules.pdf

[2] Naoki Shibuya, "Plot and Navigate a Virtual Maze", *Udacity Machine Learning Nanodegree Capstone Project Sample Report*, https://github.com/udacity/machine-learning/blob/master/projects/capstone/report-example-3.pdf, March 20, 2016.