Robot Motion Planning

I. Definition

Project Overview

In this project, a robot mouse is tasked with plotting a path from a corner of the maze to its center. The robot mouse may make two runs in a given maze. In the first run, the robot mouse tries to map out the maze to not only find the center, but also figure out the best paths to the center. In subsequent runs, the robot mouse attempts to reach the center in the fastest time possible, using what it has previously learned.

In this project, based on the provided simplified model of the world, I've created functions to control a virtual robot to navigate a virtual maze. My goal was to obtain the best score in a series of test mazes.

Problem Statement

On each maze, the robot must complete two runs. 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 reset for its second run. Its objective now is to go from the start position to the goal room in the fastest time possible.

Metrics

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 required to execute the first run. A maximum of one-thousand time steps is allotted to complete both runs for a single maze. That should be reasonable due to the score and in addition, taking into account both the second and first run as well, it balances the time and the accuracy. Apparently, the robot can find the optimal path if it explores the whole maze. So if the score is equal to the number of time steps required to execute the second run, the robot will choose to explore the whole maze in the first run. Although the robot mouse gets the optimal path, it wastes a lot of time.

II. Analysis

Data Exploration

Maze Specifications

Most mazes have an entrance and an exit. You enter the maze, and try to escape. In contrast to most mazes, this instead is a closed square maze with no entrance or exit since we do not want the mouse to escape. The maze exists on an n x n grid of squares. The value of n can be twelve, fourteen or sixteen. Along the outside perimeter of the grid, and on the edges connecting some of the internal squares, are blocked by thin pieces of wood so the mouse cannot climb the obstacles.

We put the mouse in the square in the bottom left corner of the grid facing upwards. A goal room consisting of a 2 x 2 square in the center of the grid is set up with some food inside. The mouse then explores the maze in an attempt to obtain the food.

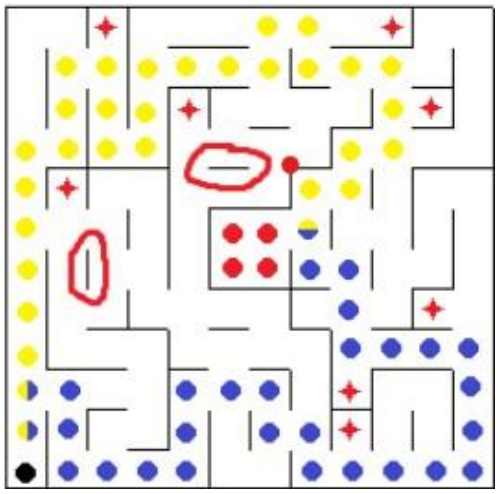
Robot Specifications

The robot mouse lives in a discrete world moving from one square to another with only the ability to strictly face either north, east, south or west. The mouse can sense the distances to the left, front and right obstacles. In this discrete world, the time is also discrete. On each time step, the robot may choose to rotate clockwise or counterclockwise ninety degrees, then move forwards or backwards with a distance of up to three units. The robot's turning and movement is perfect. If the robot tries to move into a wall, the robot stays where it is and its location is not progressed.

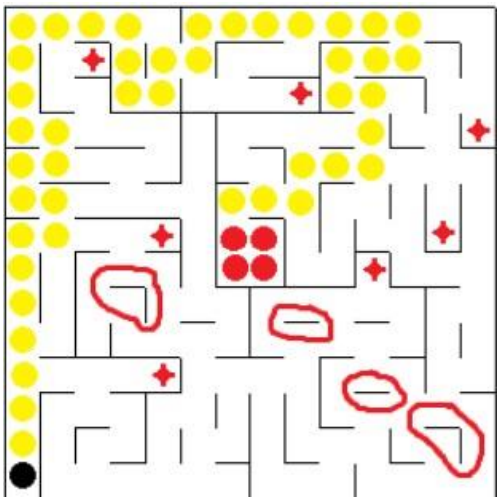
Exploratory Visualization

Starter code for the project includes three sample mazes. The black ball locates at the start point which is at the bottom left of the maze. The "red balls" goal locates at the goal area which is located at the center of the maze. At the start location, the robot is facing north. The optimal path from the start location to the goal area is indicated the "blue balls" which takes 30 single steps from the start location to the goal area. The "yellow balls" indicate the suboptimal path from the start location to the goal area. We can see that in each maze, there are many dead ends and loops. Some dead ends are indicated with read star marks and some loops are indicated with read lines.

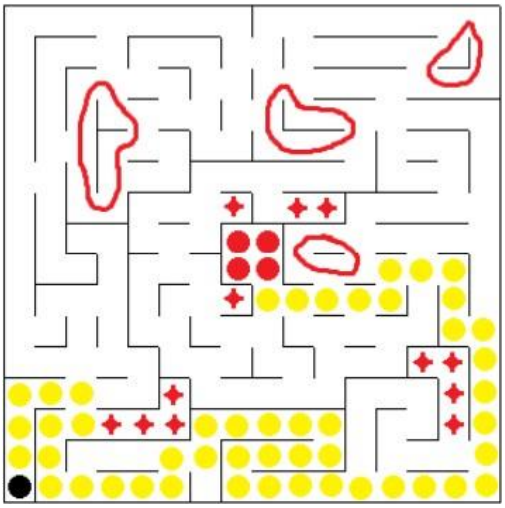
Maze 1



Maze 2



Maze 3



Algorithms and Techniques

In the first run, I will use four algorithms to exploring the maze.

- Random
- Dead-End
- Counter
- Heuristic

With the Random Controller, the robot gets valid potential rotations and movements, and based on the sensors it just randomly chooses one. The robot is moving randomly with its eyes open, so it will never hit the wall. It also knows how to move backwards to escape a dead end, but it has poor memory so it may return to the same dead end or get trapped in a loop as well.

The Dead-End Controller extends the Random Controller providing the robot better memory. Thus, preventing repetitively poor decision-making, but unfortunately still subjects a possibly of getting trapped in a loop.

The Counter Controller extends the Dead-End Controller which equips the robot with perfect memory. This means that the robot remembers every dead end it has visited, the robot remembers how many times each position has been visited and prefers to move to a less visited position. In short, it will not return to the same dead end and will never get trapped in a loop.

The Heuristic Controller extends the Counter Controller meaning, the robot not only has perfect memory but also has a “magic instinct”. In a sense, it is born to get a special feeling to the goal, and the closer to the goal, the stronger that feeling increases. When there are two valid potential positions as the same visited times, it will choose the one closer to the goal.

In the first run, no matter which controller controls the robot, each time it moves, heading, location and sensors data will be collected.

Before we start the second run, I will map the maze based on the collected data, then use dynamic programming to get the value for how long the path of each location to the goal. Using these values, I can get the optimal path from any place of the maze to the goal. In the Methodology section, I will explain the details of dynamic programming.

In the second run, on each step the robot will check all the available places it can move to, find the place that has the smallest value and move there. Finally, the robot mouse will hit the goal with the optimal path/moves.

Benchmark

Different controllers (such as Random, Dead-End, Counter, Heuristic) cause different performances. I will compare the path length, optimal moves, and score each controller obtains. The random controller will set the worst score benchmark. I make the robot stop exploring the maze until the robot visited every place of the maze, then start the second run, using the dynamic programming method, we can find the optimal path/moves which will be set as the optimal path/moves benchmark.

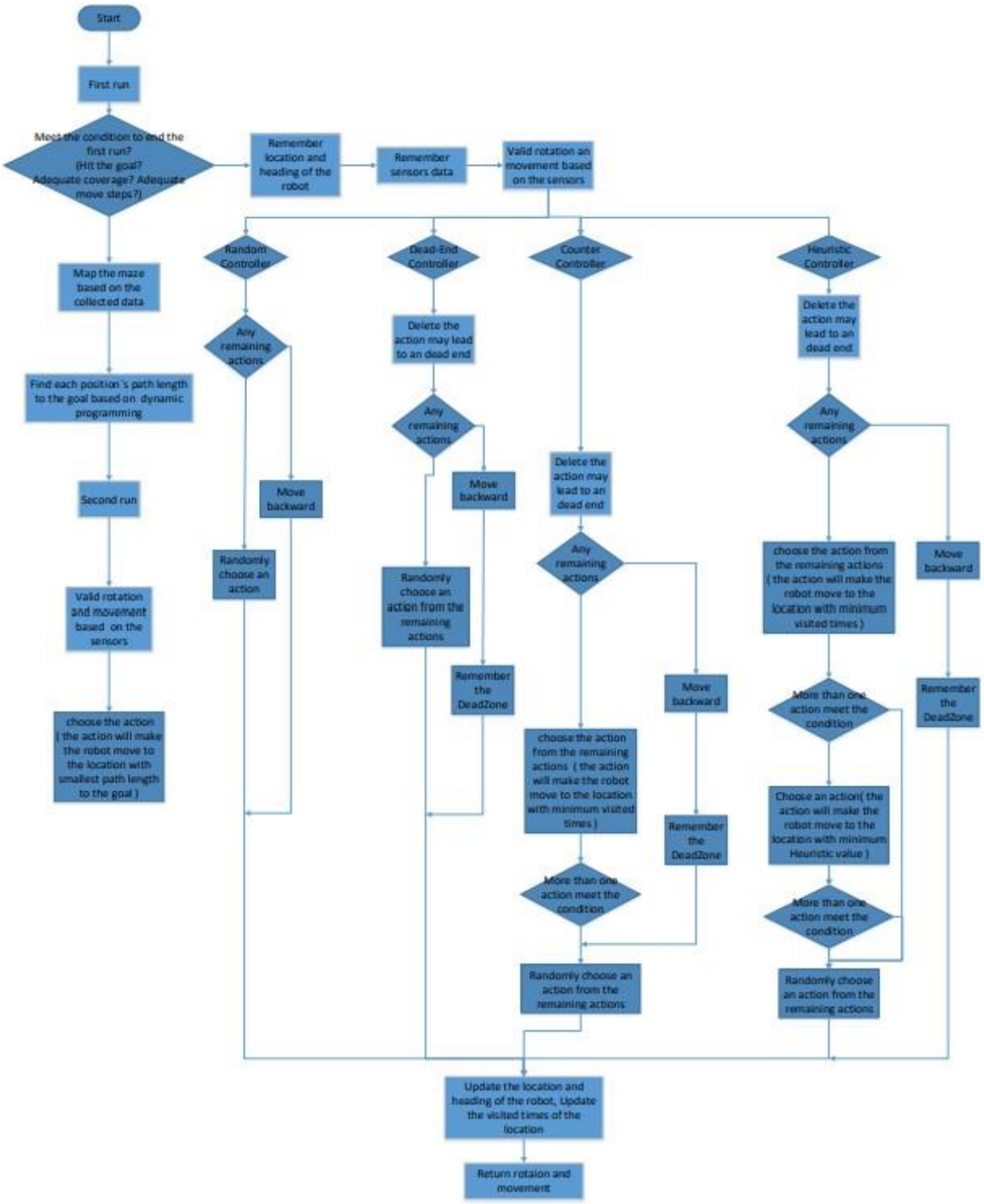
III. Methodology

Data Preprocessing

No data pre-processing is required because the obstacle sensors and robot's positioning and movements are perfect.

Implementation

The flow chart below shows the operation of the code.



Dynamic Programming

At the end of the first run, I implemented a dynamic function to find each location's path length to the goal based on the collected sensors data and location and heading data of the robot.

Pseudocode shows how it works.

1. Create a matrix with the same size as the maze (n x n), the default value of each element is 99.

2. Loop each element.

3. If the element is located in the goal bound, set its value as 0.

4. Loop adjacent element of the element.

- Check if there is any wall between the element and the adjacent element based on the collect data from the first run if there is no wall, and 1 plus the value of its adjacent element is smaller than the value of the element, set it as the new value of the element.

5. End the loop if no element's value changes any more.

Refinement

In the first run, the initial controller is the Random Controller which makes the robot move randomly, then I made a Dead-End Controller which extends the Random Controller to make the robot have the ability to avoid entering the same dead-end more than one times, then the Counter Controller extends the Dead-End Controller to make the robot tend to move to the location with less visit times, the final controller is Heuristic Controller which extends the Counter Controller to make the robot has a magic instinct to know the direction of the goal.

Before the second run, I used dynamic programming finding each location's path length to the goal based on the collected sensors data and location and heading data of the robot. In the second run, based the result of the dynamic programing, the robot can find the optimal rotation and movement to reach the goal.

IV. Results

Apparently, the Random Controller and the Dead-End controller make random moves. The Counter Controller and the Heuristic Controller also have random moves, sometimes there are two or more valid potential positions that have the same visiting times and even heuristic value, and the controller will randomly select one causing the results to differ each time. Below are the recorded results for the different controllers collected from each simulation.

Model Evaluation and Validation

Random Controller

Using the Random Controller, sometimes the robot will hit achieve its goal. The robot very easily gets stuck in the loop and visits a dead end more than once.

Maze 1

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	no	n/a	n/a	no	n/a	n/a	n/a
2	no	n/a	n/a	no	n/a	n/a	n/a
3	yes	958	0.846	yes	30	21	49.2
4	no	n/a	n/a	no	n/a	n/a	n/a
5	yes	246	0.701	yes	31	21	29.2
6	yes	767	0.965	yes	31	21	46.6
7	yes	370	0.882	yes	30	21	33.4
8	no	n/a	n/a	no	n/a	n/a	n/a
9	no	n/a	n/a	no	n/a	n/a	n/a
10	yes	412	903	yes	32	23	36.8
Average	50%	550.6	181.279	50%	30.8	21.4	39.0

Maze 2

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	193	0.505	yes	45	30	36.5
2	yes	200	0.536	yes	61	35	41.7
3	no	n/a	n/a	no	n/a	n/a	n/a
4	no	n/a	n/a	no	n/a	n/a	n/a
5	yes	541	0.643	yes	43	27	45.1
6	yes	416	0.505	yes	43	23	36.9
7	yes	654	0.806	yes	43	23	44.8
8	yes	668	0.755	yes	43	23	45.3
9	yes	581	0.801	yes	43	23	42.4
10	no	n/a	n/a	no	n/a	n/a	n/a
Average	70%	464.7	0.650	70%	45.9	26.3	41.8

Maze 3

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	212	0.504	yes	49	27	34.1
2	yes	923	0.895	yes	50	25	55.8
3	yes	624	0.762	yes	49	27	47.8
4	yes	387	0.684	yes	51	25	37.9
5	yes	271	0.504	yes	49	27	36.1
6	no	n/a	n/a	no	n/a	n/a	n/a
7	yes	261	0.504	yes	51	26	34.7
8	yes	699	0.836	yes	49	25	48.3
9	yes	689	0.746	yes	49	27	50.0
10	no	n/a	n/a	no	n/a	n/a	n/a
Average	80%	508.3	0.679	80%	49.6	26.1	43.1

Dead-End Controller

Using the Dead-End Controller, the robot also sometimes hits the goal with some failure and not much performance increase.

Maze 1

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	776	0.944	yes	34	20	45.9
2	no	n/a	n/a	no	n/a	n/a	n/a
3	no	n/a	n/a	no	n/a	n/a	n/a
4	yes	327	0.84	yes	31	21	31.9
5	yes	231	0.729	yes	30	17	24.7
6	yes	281	0.78	yes	31	21	30.4
7	yes	745	0.94	yes	31	21	45.9
8	yes	98	0.51	yes	30	20	23.3
9	yes	964	0.97	yes	31	21	53.17
10	no	n/a	n/a	no	n/a	n/a	n/a
Average	70%	488.9	0.816	70%	31.1	20.1	36.5

Maze 2

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	239	0.51	yes	45	27	35.0
2	yes	371	0.71	yes	44	23	35.4
3	yes	632	0.68	yes	43	23	44.1
4	yes	158	0.51	yes	57	35	40.3
5	no	n/a	n/a	no	n/a	n/a	n/a
6	yes	753	0.88	yes	43	23	48.1
7	yes	245	0.51	yes	43	23	31.2
8	yes	166	0.51	yes	63	39	44.5
9	no	n/a	n/a	no	n/a	n/a	n/a
10	yes	390	0.75	yes	44	23	36.0
Average	80%	369.3	0.633	80%	47.8	27.0	39.3

Maze 3

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	246	0.61	yes	69	36	44.2
2	yes	580	0.82	yes	49	25	44.4
3	no	n/a	n/a	no	n/a	n/a	n/a
4	yes	423	0.61	yes	51	26	40.1
5	no	n/a	n/a	no	n/a	n/a	n/a
6	yes	395	0.63	yes	59	33	46.2
7	yes	265	0.50	yes	49	27	35.8
8	no	n/a	n/a	no	n/a	n/a	n/a
9	yes	234	0.50	yes	49	27	34.8
10	yes	249	0.50	yes	55	27	35.3
Average	70%	341.7	0.596	70%	54.4	28.7	40.1

Counter Controller

Using the Counter Controller, the robot never misses its goal but sometimes the coverage is very high meaning the robot loses track of the goal.

Maze 1

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	224	0.85	yes	34	21	28.5
2	yes	315	0.97	yes	31	21	31.5
3	yes	271	0.92	yes	31	21	30.1
4	yes	149	0.79	yes	30	21	26.0
5	yes	228	0.92	yes	30	21	28.6
6	yes	209	0.85	yes	33	22	29.0
7	yes	271	0.96	yes	31	21	30.1
8	yes	97	0.51	yes	34	20	23.3
9	yes	385	0.97	yes	31	21	33.9
10	yes	86	0.51	yes	30	20	22.9
Average	100%	223.5	0.825	100%	31.5	20.9	28.4

Maze 2

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	128	0.51	yes	45	25	29.3
2	yes	151	0.51	yes	44	25	30.1
3	yes	130	0.51	yes	45	27	31.4
4	yes	455	0.96	yes	43	23	38.2
5	yes	134	0.51	yes	43	22	26.5
6	yes	126	0.51	yes	47	28	32.2
7	yes	171	0.51	yes	45	25	30.7
8	yes	125	0.51	yes	43	23	27.2
9	yes	146	0.57	yes	44	23	27.9
10	yes	198	0.63	yes	43	23	29.6
Average	100%	176.4	0.573	100%	44.2	24.4	30.3

Maze 3

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	258	0.71	yes	57	29	37.6
2	yes	139	0.50	yes	51	25	29.7
3	yes	175	0.56	yes	49	25	30.9
4	yes	194	0.50	yes	51	25	31.5
5	yes	277	0.70	yes	51	27	36.3
6	yes	306	0.67	yes	49	27	37.2
7	yes	181	0.52	yes	52	25	31.1
8	yes	174	0.50	yes	51	27	32.8
9	yes	628	0.98	yes	49	25	46.0
10	yes	145	0.52	yes	67	34	38.9
Average	100%	247.7	0.616	100%	52.7	26.9	35.2

Heuristic Controller

Using the Heuristic Controller, the robot performs even better while also noticing the coverage being lower than the results from using the Counter Controller.

I set a rule that the coverage must be higher than 0.5 before the robot ends the first run. But we can see that in "Maze 3", the coverage is always 0.5 in ten trails which means the robot always hits the goal at a low coverage. So, I change the rule in regard so that the robot can end the first run having only one condition that it hits the goal. Now trail 11 in "Maze 3" shows the result of it hitting the goal with 0.25 coverage and get the best score without finding the optimal way.

Maze 1

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	220	0.73	yes	31	17	24.4
2	yes	241	0.84	yes	31	17	25.1
3	yes	221	0.72	yes	31	17	24.4
4	yes	134	0.60	yes	31	17	21.5
5	yes	181	0.71	yes	32	18	24.1
6	yes	116	0.51	yes	30	17	20.9
7	yes	178	0.71	yes	32	18	24.0
8	yes	121	0.52	yes	31	17	21.1
9	yes	114	0.51	yes	30	17	20.8
10	yes	136	0.62	yes	31	17	21.6
Average	100%	166.2	0.647	100%	31	17.2	22.8

Maze 2

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	177	0.61	yes	44	25	30.9
2	yes	162	0.55	yes	45	29	34.4
3	yes	140	0.51	yes	45	27	31.7
4	yes	182	0.61	yes	44	25	31.1
5	yes	176	0.61	yes	43	25	30.9
6	yes	135	0.51	yes	45	27	31.5
7	yes	141	0.51	yes	45	27	31.7
8	yes	153	0.52	yes	45	29	34.1
9	yes	137	0.51	yes	45	27	31.6
10	yes	160	0.55	yes	46	29	34.4
Average	100%	156.3	0.549	100%	44.7	27	32.2

Maze 3

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	177	0.50	yes	51	28	33.9
2	yes	178	0.50	yes	51	31	37.0
3	yes	173	0.50	yes	51	31	36.8
4	yes	197	0.50	yes	53	27	33.6
5	yes	197	0.50	yes	53	27	33.6
6	yes	176	0.50	yes	51	28	33.9
7	yes	178	0.50	yes	51	31	37.0
8	yes	153	0.50	yes	51	31	36.1
9	yes	176	0.50	yes	51	28	33.9
10	yes	191	0.50	yes	51	31	37.4
Average	100%	179.6	0.500	100%	51.4	29.3	35.3
11	yes	73	0.25	yes	51	31	33.5

Justification

The below table compares the Counter Controller and Heuristic Controller with best score and average score for each test maze.

Test Maze	Counter Controller Best Score	Counter Controller Average Score	Heuristic Controller Best Score	Heuristic Controller Average Score
01	22.9	28.4	20.8	22.8
02	26.5	30.3	30.9	32.2
03	29.7	35.2	33.6	35.3

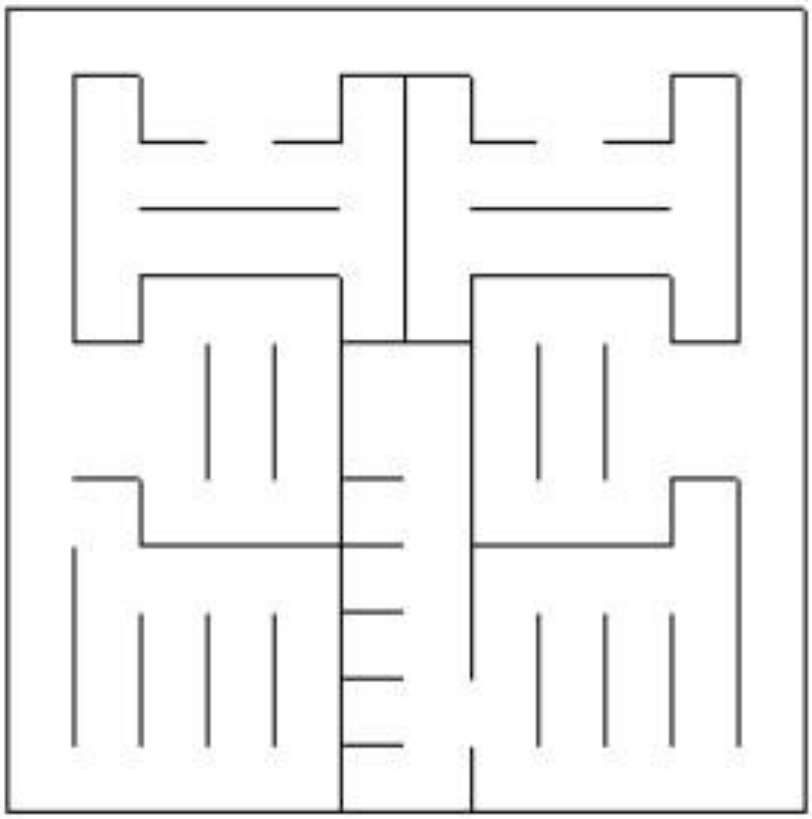
V. Conclusion

Apparently, both of the two controllers (“Counter Controller” and “Heuristic Controller”) perform very well. They successfully make the robot hit the goal in every trail. I think the Heuristic Controller is comfortably able to gauge the distance between the goal and the robot with efficiency, while in some cases this ability is not as useful.

Free-Form Visualization

In order to discover when the Heuristic Controller performs less efficient, I designed a new maze in file “test_maze_04.txt”. Below are the test results of both the Counter Controller and Heuristic Controller in simulation.

Maze 4



Counter Controller

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	81	0.44	yes	43	18	20.7
2	yes	201	0.78	yes	44	18	27.1
3	yes	133	0.66	yes	44	18	25.5
4	yes	59	0.38	yes	44	18	20.0
5	yes	234	0.90	yes	44	18	25.8
6	yes	65	0.42	yes	44	18	20.2
7	yes	131	0.64	yes	44	18	22.4
8	yes	286	0.95	yes	43	18	27.6
9	yes	71	0.38	yes	44	18	20.4
10	yes	282	0.85	yes	43	18	27.4
Average	100%	154.3	0.640	100%	43.7	18.0	23.7

Heuristic Controller

Trail	First Run			Second Run			Score
	Goal?	Moves	Coverage	Goal?	Path length	Moves	
1	yes	275	0.85	yes	44	18	27.2
2	yes	273	0.85	yes	44	18	27.1
3	yes	225	0.83	yes	44	18	25.5
4	yes	225	0.83	yes	44	18	25.5
5	yes	220	0.83	yes	44	18	25.4
6	yes	273	0.84	yes	44	18	27.1
7	yes	226	0.83	yes	44	18	25.6
8	yes	227	0.83	yes	44	18	25.6
9	yes	286	0.85	yes	44	18	27.6
10	yes	265	0.85	yes	44	18	26.9
Average	100%	249.5	0.839	100%	44.0	18.0	26.4

Counter Controller Best Score	Counter Controller Average Score	Heuristic Controller Best Score	Heuristic Controller Average Score
20.0	23.7	25.4	26.4

Comparing the two controllers' performance, we can see that both of them get the job done. The Heuristic Controller performs more stable than the Counter Controller, with the Counter Controller performing less consistently efficient. This noted, overall the Counter Controller performs better than the Heuristic Controller on average.

Reflection

I find it quite satisfying to be able to distinguish between simulation events as to which controllers perform best or uniquely accordingly to their circumstances. At first, the Random Controller makes the robot moves totally random causing the robot to hit the wall. I think a blind robot mouse's behavior is not a good benchmark, but instead allowing it to see the obstacles and learn how to escape and avoid dead-ends.

The robot mouse's performance under the Counter Controller really impressed me. Noticing how the robot mouse never misses the goal due to good memory really helps me to understand why memory is so important. The robot mouse in the second run can be seen as the offspring of the robot mouse in the first run. The second robot learns from the first which makes it easily hit the goal. Overall, the genetic algorithm may also offer a good solution for problems like finding the optimal path.

I noticed the Heuristic Controller would not always work well meaning that the local optimal choice may not be the global optimal choice. Sometimes the Heuristic Controller helps the robot hit the goal with much fewer steps, and in other cases it's far less efficient.

In the second run, I chose dynamic programming to find every position's optimal path to the goal.

Improvement

This was a pretty cool project, being that it is a simplified maze completion because the real world is continuous with vast amounts of variances while in this project, everything is discrete, precise, and for the most part controlled. So, some techniques are not used such as Gaussian filter, PID, SLAM. It would be interesting to check out the Robotics Software Engineer Nanodegree to learn more of these particular techniques and use them in the real world.

References

- Plot and Navigate a Virtual Maze - Project Description
- Plot and Navigate a Virtual Maze - Naoki Shibuya (report example 3)