Assignment 1
Robotic challenge solver using the CiberRato simulation environment

Diogo Andrade[1] and Luís Larajeira[2]

[1] Universidade de Aveiro diogo.andrade@ua.pt 89265
[2] Universidade de Aveiro lclm@ua.pt 81526

Date: November 21, 2021

1 Introduction

Antes de indicar soluções, deve-se indicar qual o problema. Neste caso, o que é o Challenge ou, pelo menos, qual o contexto em que estes challenges se realizam.

In first challenge, we have a basic approach where when the front sensor is about to collide with a wall, the agent will rotate, for the side of the sensor (right or left) with the lowest value. In this challenge, the agent complete the 10 laps on approximately 6350 time cycles.

In second challenge, the agent moves on map using the GPS and compass, when the agent is on a cell where it already pass, it will look for positions it left behind and find a path using A* [1] [2] algorithm. This challenge is successfully complete in approximately 2420 time cycles.

In the last challenge, we use the same approach as the second challenge for agent navigation, after finding all targets, we use the A* [1] [2] algorithm to find the shortest path between targets, if there is a better path through the unexplored area, it will explore this path, otherwise we will find the best path that the agent can take going from the first target and ending on it, passing through the other targets. This challenge is successfully complete in approximately 2630 time cycles.

2 C1 – Control challenge

2.1 Objective

The objective of the first challenge is to control the movement of a robot through an unknown closed circuit as fast as possible and without colliding with the surrounding walls.

2.2 Approach

The approach we use was, the agent moves always in front, when the front sensor is about to collide with a wall, the agent will rotate, for the side of the

sensor (right or left) with the lowest value, that is farther from the wall. If the agent is about to collide with one of the side walls, the agent will be do an adjustment reducing the velocity of the opposite wheel.

We also implemented a security mechanism, so when the agent turns around (360º), this mechanism is activated when it detects that it has returned to the last target it passed.

2.3 Results

In this challenge, the agent make a lap in approximately 634 time cycles, that is 7,9 complete laps in 5000 time cycles (3150 points with 0 collisions). The agent can do the 10 laps on approximately 6350 time cycles, without collide (4000 points).

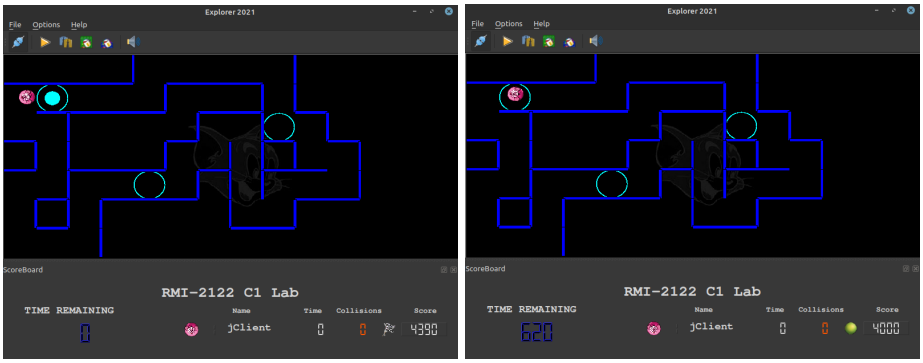

Fig. 1. Challenge 1 execution example.

Listing 1.1. Challenge 1 execution output example.

```
$ ./run.sh -c 1
Status: Ok
Lap: 1 - Lap Time: 615.0
Lap: 2 - Lap Time: 631.0
Lap: 3 - Lap Time: 626.0
Lap: 4 - Lap Time: 647.0
Lap: 5 - Lap Time: 627.0
Lap: 6 - Lap Time: 633.0
Lap: 7 - Lap Time: 633.0
Lap: 8 - Lap Time: 640.0
Lap: 9 - Lap Time: 647.0
Lap: 10 - Lap Time: 643.0
Complete 10 laps - Total Time: 6342.0 - Average Time: 634.2
```

3 C2 – Mapping challenge

3.1 Objective

The goal of this challenge is to explore an unknown maze in order to extract its map.

3.2 Approach

The agent moves on the map using the GPS and the compass, this movement is done every 2 positions. When the agent is near the center of the cell it will map the positions around it.

In this approach contrarily to the previous one, the movement is always made by checking if the position of the robot is centered in the target cell, relative to the position of the map.

The agent always prefers to continue to do the same movement, if this is not possible, it will go to one side.

If the agent's only possible move is to a cell he has already mapped, it will look for positions it left behind. If it doesn't find any, it means it's already mapped the entire maze.

For efficiency the agent uses an A* [1] [2] algorithm, to evaluate the best path to cell that was unexplored, our approach is focused en using a graph so that this algorithm can be easily implemented. For the cost between cells we use the absolute distance, and the rotations, as this impacts seriously how much time is needed to travel from point A to point B.

It is possible that all positions around a cell are mapped even if the agent does not pass through the middle of that cell, in which case the agent does not need to go to that cell.

3.3 Results

This challenge is successfully complete in approximately 2410 time cycles, mapping the entire maze correctly.

Listing 1.2. Challenge 2 execution output example.

```
$ ./run.sh -c 2
Status: Ok
Time: 2407.0
File created: mapping.out

$ awk -f mapping_score.awk mapping.out ../agent/mapping.out
mapping score:3110
```

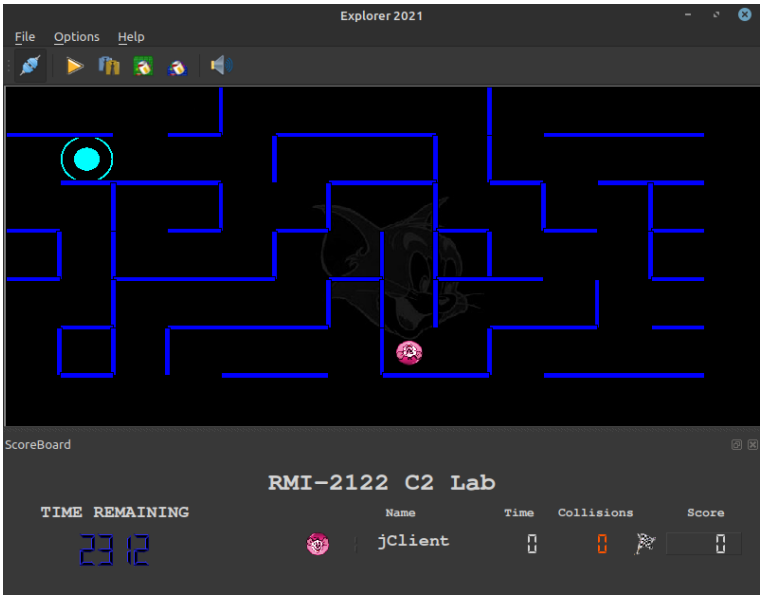

Fig. 2. Challenge 2 execution example.

4 C3 – Planning challenge

4.1 Objective

The objective of this challenge is to explore an unknown maze in order to locate all target spots and compute a best closed path that allows to visit all target spots starting and ending in the starting spot.

4.2 Approach

For this agent we have recurred to the mapping agent used in the previous challenge, adding some small changes and tweaks.

The agent moves thought the unknown world, mapping like in the previous challenge, but also trying to discover each target. For each target found the agent saves it's information, like the relative position.

After finding all targets, the agent will continue to look for paths that could lead to a more efficient way of travelling between each target, this process is done while trying to avoid to explore the whole map, as it's not intended to do so.

The process of finding the best closed loop between all targets is done once again recurring to the A* [1] [2] algorithm, using the same approach of costs between cells as the previous challenge.

Contrary to what happens in the previous challenge the agent doesn't validate if the positions around a cell are mapped.

4.3 Results

This challenge was also successful, the agent can find the correct best closed path that allows to visit all target spots starting and ending in the starting spot, without mapping the whole maze.

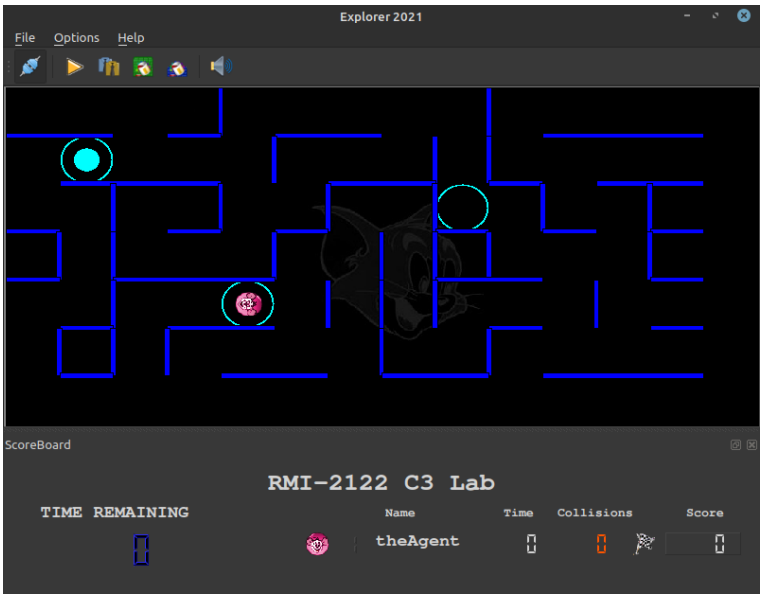

Fig. 3. Challenge 3 execution example.

Listing 1.3. Challenge 3 execution output example.

```
$ ./run.sh -c 3 -f path.out
Status: Ok
There may be a better path between targets 0 and 2
There may be a better path between targets 0 and 2
Time: 2634.0
File created: path.out
```

Faltou avaliar o agente com os programas de teste fornecidos...

5 Conclusion

Developing the agents or each challenge as allowed us to understand some of the characteristics of this type of robot, and what challenges we might expect while developing this type of software. As the agent needs to be aware of its surroundings and for each step or decision, it needs to make to avoid colliding with encountered objects or travel from point A to point B.

One of the most challenging problems we encountered was the speed to apply in each motor at a certain time, and calibrate each one correctly and the angle of the sensors and how much it impacts the agent decisions. It's also important to refer that the sensors provided readings with some noise, which we need to take in account to correctly identifying the real position of the robot relative to each object. We found out that the variation of the threshold for the sensor readings had great impact in the performance of the agent.

To conclude, we believe that agents can solve the propose challenges quickly and efficiently. What could be improved would be a better move decision in the maze mapping, which would lead to better efficiency.

References

1. Graphs in Java - A* Algorithm, https://stackabuse.com/graphs-in-java-a-star-algorithm/. Last accessed 16 Nov 2020
2. GitHub Repository, https://github.com/marcelo-s/A-Star-Java-Implementation. Last accessed 16 Nov 2020