Assignment 2
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: January 23, 2022

1 Introduction

The CiberRato simulation environment is composed by an agent on an unknown maze, and its aim is to evaluate if the agent can pass the proposed tasks. The maze has maximum size of 7 by 14 cells and contains multiple targets/beacons. The agent is equipped with 2 motors, 4 obstacles sensors, a ground sensor and a compass (it also has 3 leds, a collision sensor and a GPS, none of them used in this work).

The challenge proposed is divided in three tasks localization, mapping and planning. For the first task, we use the movement equations and the obstacles sensors to make a estimate of agent position, and with the compass the agent angle. On second task, mapping, the agent runs through the entire maze and register, using the obstacles sensors, if a cell is rounded by walls or not. For the planning task, the agent will register, during the mapping step and with assistance of ground sensor, if a cell is a target; after completed the mapping the agent use the A* [1] [2] algorithm to find the closed path with minimal cost between targets.

In this challenge, we use the sensor on angles of 90^{o} (0^{o}, 90^{o}, -90^{o}, 180^{o}/-180^{o}), and every time that the agent is in the center of a cell it will rotate, to correct its angle, for a maximum deviation of 1.5^{o}.

This challenge is successfully complete in approximately 4200 time cycles (3730 time cycles for mapping, plus 470 time cycles to go to initial position), and without collisions. It can map the maze and plan the closed path in perfectly.

2 Localization

2.1 Equations

Power applied to motors in time t:

$$out_t = \frac{in_i + out_{t-1}}{2} * \mathcal{N}(1, \sigma^2)$$

Translation movement in time t:

$$lin = \frac{out_t^l + out_t^r}{2}$$

$$x_t = x_{t-1} + lin * cos(\theta_{t-1})$$

$$y_t = y_{t-1} + lin * sin(\theta_{t-1})$$

Rotation movement in time t:

$$rot = out_t^l + out_t^r$$

$$\theta_t = \theta_{t-1} + rot$$

2.2 Objective

The objective of the first task is to control the movement of the agent through an unknown maze using the movement model and without colliding with the surrounding walls.

2.3 Approach

For locate the agent on map, we use the movement equations showed above plus the compass and obstacle sensors. In the orientation part, we create a rule that say the estimate angle is 2 time more right over the compass read.

$$\theta_t = \frac{\theta_t * 2 + compass}{3}$$

In terms of position, the calculations are more complex, firstly we need to verify if the agent orientation is not to far form the optimal (0^o, 90^o, -90^o, 180^o/-180^o), after that we compute the distance of agent from the walls, using the next formulas:

$$inverseSensorRead^s = \frac{\mid cos(angleDeviation) \mid}{sensorRead^s}$$

$$x^s = x_{wall} \pm (inverseSensorRead^s + wallSize + agentRadius)$$

$$y^s = y_{wall} \pm (inverseSensorRead^s + wallSize + agentRadius)$$

Now, we estimate using the side distance values, and only when exist walls in both sides, the x position in case of agent moves on y axle or the y position in case of agent moves on x axle, on following way:

$$x_t = \frac{x_t * 2 + \frac{x^l + x^r}{2}}{3}$$

$$y_t = \frac{y_t * 2 + \frac{y^l + y^r}{2}}{3}$$

If exists a wall in front or backward, we use a different rule, using the front or back distance values, and is just used one of than (the closer):

$$x_t = \frac{x_t * 3 + x^s}{4}$$

$$y_t = \frac{y_t * 3 + y^s}{4}$$

3 Mapping

3.1 Objective

The goal of this task is to explore an unknown maze to completely map it and localize the target spots placed in the maze. After completing this task, the agent will also be at its starting point.

3.2 Approach

The agent moves on the maze using the compass and its sensors, as explained in the environment. When the agent is near the center of the cell, it will map its positions.

Our approach consists of mapping the entire maze while also discovering each target. The agent saves information for each target found, like the relative position.

While mapping, 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.

After mapping all the maze and finding all targets, the agent will also find the best path to the starting point based on the A* [1] [2] algorithm.

4 Planning

4.1 Objective

The objective is to complete what was done in the task of mapping. Computing a closed path with minimal cost allows visiting all target spots starting and ending in the starting spot.

4.2 Approach

We have recurred to the planning task used in the previous challenge, adding some minor changes and tweaks for this agent.

The process of finding the best-closed path with minimal cost that allows visiting all targets is done once again recurring to the A* [1] [2] algorithm, using the same approach of costs between cells as the last challenge.

5 Results

This challenge is successfully complete in approximately 4200 time cycles, 3730 time cycles of that time is for mapping and 470 time cycles for the agent go to initial position. The mapping task is made correctly, and the agent can also find all targets, and the correct best closed path, with minimal cost, that allows to visit all target spots starting and ending in the starting spot.

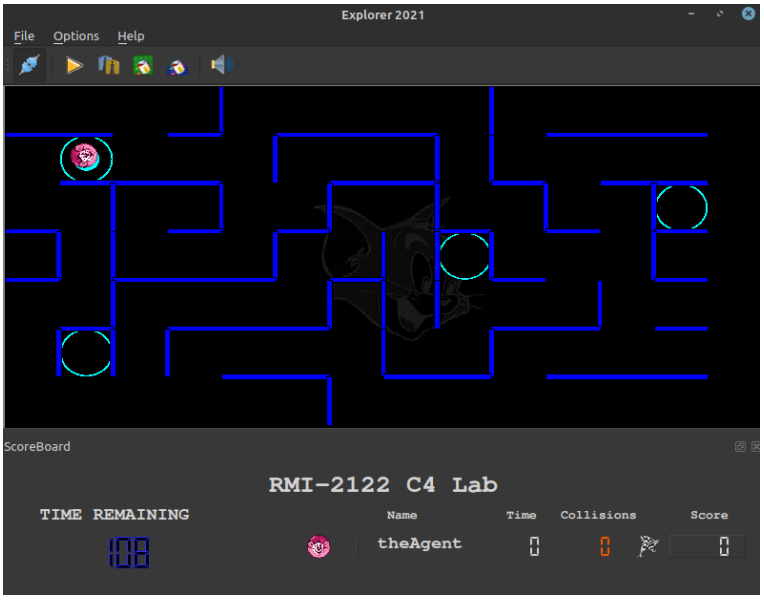

Fig. 1. Challenge 4 execution example.

Listing 1.1. Challenge 4 execution output example.

```
$ ./build.sh

$ ./run.sh
Status: Ok
File created: file.out.map
Time: 4208.0
File created: file.out.path

$ awk -f mapping_score.awk planning.out ../agent/solution.map
mapping score:3130

$ awk -f planning_score.awk planning.out ../agent/solution.path
```

```
x 0 y 0
x -2 y 0
x -2 y -2
x 0 y -2
x 0 y -4
...
x 0 y 0
Planning score: 1
```

6 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 2021
2. GitHub Repository, https://github.com/marcelo-s/A-Star-Java-Implementation. Last accessed 16 Nov 2021