Assignment 1

Alexandre Abreu[1 [89290]] and Rodrigo Martins[1 [93264]]

Universidade de Aveiro

Keywords: Autonomous agent · A-star algorithm · ciberrato · Robótica Móvel e Inteligente.

1 Introduction

In this assignment the C1 challenge was completed using Java and the C2 and C3 challenges were done using Python. In C2 and C3 the default movement was the same with some adjustments to fulfill the tasks presented by them. In both C2 and C3 challenges the A star algorithm was used in order to find the shorthest path to a given position.

In the C1 challenge the agent does aproximately 8 laps around the maze in 5000 cycles, in the C2 challenge the agent can map everything in aproximately 4100 cycles and in the C3 challenge the agent can map everything and find the shortest required path in aproximately 4200 cycles.

2 C1 challenge

In the C1 challenge the agent was programmed to always drive straight forward and if the agent got too close to a side wall it would then turn slowly to the oposite side of that wall following an equation that gives the turning speed. The turning speed will only be maximum if the agent doesn't have enough space in front of sensor 0 and in that case it will turn to the left or to the right depending on the free space on those cells.
The power of the left motor was calculated using this equation $LeftMotor = lin + rot/2$ and the power of the right motor using this one $RightMotor = lin - rot/2$, lin represents the linear value of speed and will always be 0.15 and the rot represents the rotation speed and depends on the distance to the side walls of the agent. The agent movement was programmed in the following way:

```
double lin = 0, rot = 0;

if (irSensor0 <= 1.1){
    lin = 0.15;

    if (irSensor1 > 3){                // wall too close to left side -> turn right
        rot = 0.1*(1/irSensor1);
    }else if (irSensor2 > 3){          // wall too close to right side -> turn left
        rot = -0.1*(1/irSensor2);
    }

    cif.DriveMotors(lin + rot/2, lin - rot/2);

}else  if (irSensor1 > irSensor2){    // wall too close to left side -> turn right
    rot = 0.15;
    cif.DriveMotors(lin + rot, lin - rot);
}else if (irSensor1 < irSensor2){    // wall too close to right side -> turn left
    rot = -0.15;
    cif.DriveMotors(lin + rot, lin - rot);
}
```

3 C2 challenge

In the C2 Challenge the agent was programmed to run straight forward with maximum speed (0.15, 0.15) on both motors until it notices a wall in front of it. While running forward the agent must follow a certain angle depending on the direction it's going (0^o, 90^o, -90^o, 180^o) if the agent strays from the angle for more than 3^o it will then rotate in place to slightly adjust the angle to its intended value. If the agent gets too close to a wall while running forward, the power of power of the motors will be adjusted slightly depending on which side of the agent is too close to a wall, if there is a wall close to the agent on its left, the

power of the motors will be (0.15, 0.14), and if there is a wall close to the agent on its right, the power of the motors will be (0.14, 0.15).
While the agent is running it will read each sensor to register if there are any walls or free space nearby but only if it stands on an even cell(this is calculated by subtracting the initial gps position of the agent to the current gps position and checking if it's even).
When the agent notices a wall right in front of it, it will then select the closest free cell (the distance between the agent and cells is calculated in a straight line, ignoring walls, using the Pythagorean theorem), when the nearest free cell is selected the program will start the A star algorithm, the initial position will be the current agent position and the goal will be the previously selected free cell, the program will also give the algorithm the playable area (visited cells), the position of the walls to avoid the algorithm selecting a path that goes inside walls and the position of the other free cells known to the agent, this is done because the algorithm might find a path to another free cell before it finds a path to the current goal, this saves time and mitigates some of the existing flaws of the way the program selects the closest free cell.
These actions (moving forward until a wall is too close, selecting the closest free cell and starting the A star algorithm) are repeated until the agent has explored all of the free cells of the map, when all of the cells are explored the program will end and the map will be printed to the output file.
Following this strategy the agent is able of maping the entirety of the C2 map in about 4100 cycles.

4 C3 challenge

In the C3 challenge the agent was programmed in a similar way to the C2 agent, this means that the behaviour of the agent will the same except when the agent detects it's passing by a beacon it will store its id and position. When the agent has explored the entirety of the map, it will then start the A star algorithm, the initial position will be the starting beacon and the goal will be to create the shortest path that goes over all of the beacons and returns to the initial position.
Following this strategy the agent is able to map the entirety of the C3 map in about 4600 cycles and of produce the optimal path for this map.

5 Conclusion

In this assignment we were able to achieve all of the goals of each challenge but there is still room for improvement. In the C3 challenge the agent doesn't always need to explore the entirety of the map to find the optimal path and the way we compute the shortest path could also be improved by using permutations to find all of the paths that start and end in the initial position and go through all the beacons and then check which one is the shortest.