universidade de aveiro

Project 2

Robotic challenge solver using the CiberRato simulation environment

Intelligent Mobile Robotics
DETI

Rui Miguel Oliveira (89216)
Carlos Costa (88755)

Introduction

- **Problem Context**
    - Localization
    - Mapping
    - Planning
- **Proposed Solution**

Localization (GPS)

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

$$\begin{aligned} x_t &= x_{t-1} + lin * cos(\theta_{t-1}) \\ y_t &= y_{t-1} + lin * sin(\theta_{t-1}) \\ lin &= \frac{out_t^l + out_t^r}{2} \end{aligned}$$

$$\begin{aligned} \theta_t &= \theta_{t-1} + rot \\ rot &= \frac{out_t^r - out_t^l}{D} \end{aligned}$$

- **Translation Equations**
- **Orientation Equations**
- **“Sensor’s correction”**

Example of a **front** correction when the robot is facing **North**:

closest_x_wall_distance = 1/measures.irSensor[0]
closest_x_wall_coordinate = round(movement_model_x + 1)
sensors_x = closest_x_wall_coordinate - wall_diameter - closest_x_wall_distance - robot_radius

Localization (Compass)

- Conversion to degrees
- Normalization (0 to 180 and 0 to -180)
- Average between theta (movement model) and the noisy compass

Mapping

Three different main states:

1. **In Rotation**
2. **At the center of a cell**
3. **Between cells**

Mapping (In Rotation)

In this case, the robot will remain in this state until it reaches the value of the previously proposed compass.

Mapping (At the center of a cell)

In this state the program makes a **significant sequence of actions**:

- If the current coordinate is a **beacon**, its coordinate and identifier are added to a list of beacons.
- The current state of visited positions is drawn in an **output file**.
- The **positions** around the robot are evaluated in **visited or to be visited**.
- The **A* map is updated** (All map positions are initialized to 1 and whenever a navigable position is discovered the map is updated with 0 in these respective positions)
- From the A* algorithm and the set of visitable positions, the **nearest point** and the respective path (set of navigable positions) from the robot to it is calculated. If all positions have been visited, it returns to the initial position and the program ends.

Mapping (Between cells)

If the robot is crossing between cells, the following instructions are carried out:

- If the agent has not yet reached the next position (current coordinate plus 2 diameters) it advances.
- If for some reason it exceeds the predicted coordinate, it slowly re-treats.

As there is a considerable amount of noise, sometimes the robot does not move perfectly. There is control over the distances between walls. In other words, whenever the robot deviates from the center of the cell or is too close to a wall in front, there is an adjustment of its movement.

Thus, the greater the distance between the agent and a wall, the greater the deviation to return to the center of the cell.

Planning

In order to determine the closed path with minimal cost that allows to visit all target spots we apply the **traveling salesman** problem.

Thus, **after exploring the entire map and returning to the initial position**, the program simulates all **permutations** of **positions** between **beacons** and chooses the **smallest** one that starts at the initial position.

To determine the distances between beacons, we use the **A* algorithm** again.

Handling exceptions and collisions

The agent is somewhat prepared to handle collisions with walls and exceptions thrown at runtime.

In both cases, in order to **preserve the current state of the map**, since it is essential for generating the final path, the program leaves its natural run, calculates the final path and ends.

In this way, we will be left with a map that, although not fully explored, remains a **valid map**.

End of simulation time

If at some point the simulation time ends, the final path is calculated with the map explored so far and the program ends.

Results (Map Output File)

```
 - - - - - - - - - - - - - -
|XXXXXXX|XXXXXXXXX|XXXXXXXXX|
 - - X - X - - - X X - - - X
|XX0XXXXXX|XXXXX|X|XXXXXXXXX|
 X - - - X X - - X - X - - X
|XXX|XXX|XXX|XXX|XXX|XXX|3XX|
 - X X - X - X X - X - X - X
  |X|XXXXX|XXX|X|1|XXXXX|XXX|
 - X - - - X - X X - X X - X
|XXX|XXXXXXX|X|X|XXXXX|XXXXX|
 X - X - - - X X X - - X - X
|X|2|X|XXXXXXX|XXX|XXXXXXXXX|
 X X X X - - X - - X - - - X
|XXXXXXXXXXX|XXXXXXXXXXXXXXX|
 - - - - - - - - - - - - - -
```

(Média tempo restante: 650 ticks)

Results (Path Output File)

```
 1 0 0 #0     11 14 -2       21 18 -6       31 24 -4      41 10 -10     51 -2 -10
 2 2 0        12 16 -2       22 18 -4       32 24 -6      42 10 -8      52 -2 -8
 3 4 0        13 16 -4       23 20 -4       33 22 -6      43 8 -8       53 -2 -6
 4 6 0        14 18 -4       24 20 -6       34 20 -6      44 6 -8       54 0 -6
 5 6 2        15 18 -6       25 22 -6       35 20 -8      45 4 -8       55 0 -4
 6 8 2        16 16 -6       26 24 -6       36 18 -8      46 4 -10      56 0 -2
 7 10 2       17 14 -6       27 24 -4       37 16 -8      47 2 -10      57 -2 -2
 8 12 2       18 14 -4 #1    28 24 -2       38 16 -10     48 0 -10      58 -2 0
 9 14 2       19 14 -6       29 22 -2 #3    39 14 -10     49 0 -8 #2    59 0 0 #0
10 14 0       20 16 -6       30 24 -2       40 12 -10     50 0 -10
```

(Média tempo restante: 650 ticks)

Bibliography

https://github.com/BaijayantaRoy/MediumArticle/blob/master/A_Star.ipynb?fbclid=IwAR0caLz9Z7JVTpnJdKc200_DAxtHy94aIrZd0ZHKRwwzP05kypoI2B3ynX4