Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática

Robótica Móvel e Inteligente / Intelligent Mobile Robotics
(Academic year of 2022-2023)

Assignment 2

Robotic challenge solver
using the CiberRato simulation environment

November, 2022

1 Objectives

In this assignment each group should develop a robotic agent to command a simulated mobile robot in order to implement a set of robotic tasks, involving different navigation skills.

The list of tasks is the following:

1. Localization: The agent needs to navigate and localize itself in an unknown maze, defined by line strips drawn on the floor, using the movement model, the line sensor and the compass. GPS is not available. Motors, line sensor and compass are noisy. You can consult the `C4-config.xml` file to get the noise parameters. Every cycle that the robot is completely out of the strip a penalty is applied.
2. Mapping: The agent needs to explore an unknown maze in order to **extract its map** . At the same time, the agent needs to localize target spots placed in the maze. **The number of target spots is available at the beginning. This number may change between mazes.** After completing the mapping task, the agent **should return to the starting spot**.
3. Planning: The agent needs to compute a closed path with minimal cost that allows to visit all target spots, starting and ending at the starting spot.

Figure 1 depicts an example of a maze with 3 target spots. Target 0 is always the starting spot.

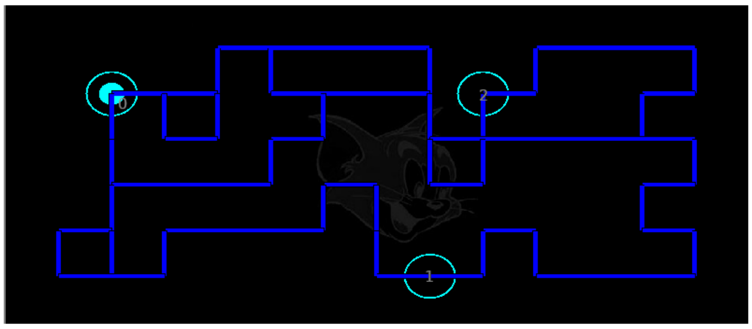

Figure 1: Example of a maze.

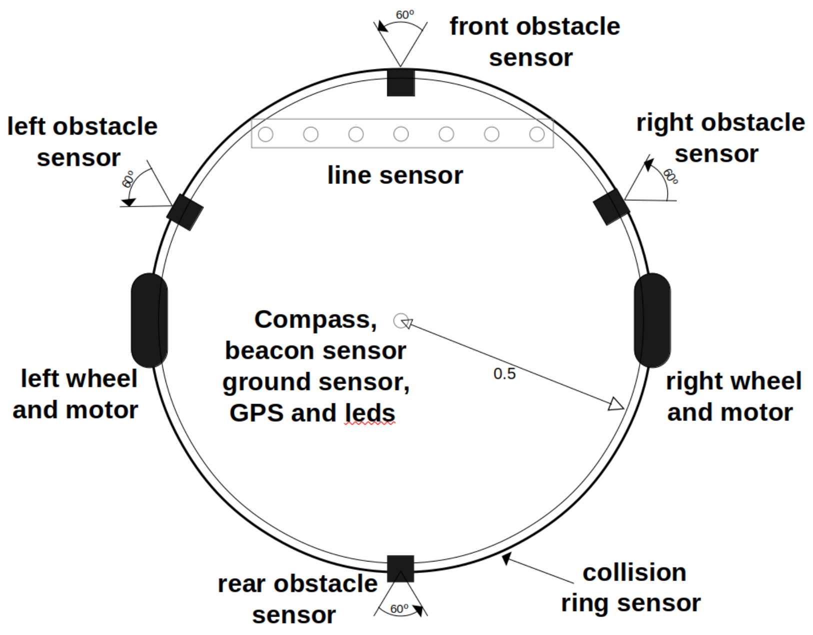

Figure 2: The simulated robot.

2 The CiberRato environment

The CiberRato simulation environment will be used to assess the agent developed to overcome the different tasks. The simulated robot (see figure 2) is equipped with 2 motors (left and right) and 3 leds (visiting, returning and finish). In terms of sensors, it includes a compass, four obstacle sensors, a ground sensor, a collision sensor and a line sensor. The available sensors depend on the challenge to be solved.

The simulated robot navigates in a delimited rectangular arena that can be seen as a bi-dimensional array of fixed size cells. Each cell is a square with side length equal to twice the diameter of the robot. The maximum size of the arena is 6-cells tall and 13-cells wide.

A maze is defined by putting thin strips along the middle of cells, which can be detected using the line sensor. The target cells are detectable by the ground sensor.

3 Movement model

Consider that the robot's pose is given by (x, y, θ), where x and y define the robot position and θ specifies the robot orientation. When the command sent to the simulator at step t is `DriveMotors(`in^l_t, in^r_t`)`, then the following equations determine the new robot pose.

An IIR filter is applied to each of the powers provided by the agent (in^l_t and in^r_t) that models the inertial characteristics of the motors and generates the effective powers that will be applied to the motors, corresponding to

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

where out_t is the power applied at time t, out_{t-1} the power applied at time $t-1$, and $\mathcal{N}(1, \sigma^2)$ Gaussian noise with mean 1 and standard deviation σ.

Then, the movement is splitted in a translation of the robot position, considering its current orientation, followed by a the change of the orientation of the robot. For the translation one has

$$x_t = x_{t-1} + lin * cos(\theta_{t-1}) \tag{2}$$

$$y_t = y_{t-1} + lin * sin(\theta_{t-1}) \tag{3}$$

$$lin = \frac{out_t^l + out_t^r}{2} \tag{4}$$

and for the rotation

$$\theta_t = \theta_{t-1} + rot \tag{5}$$

$$rot = \frac{out_t^r - out_t^l}{D} \tag{6}$$

where D is the robot diameter (1 in the CiberRato environment). This provides the new robot pose (x_t, y_t, θ_t) at the next step, in case no collisions occur. If the new pose involves a collision, the simulator only applies the rotational component.

4 Scoring scripts

Each execution of your agent should create a **map file** and a **path file** . These files should have the same base filename and have extensions `.map`, for the map file, and `.path`, for the path file (see `run.sh` for an example on how to create these files).

To check if the map file is correct, you can use the `mapping_score.awk` script, executing, in the `simulator` directory:

awk -f mapping_score.awk mapping.out ≪***path-to-your-map-file***≫

The map file is a text file that contains 21 lines, each with 49 characters, where the center of the file always represents the starting position, marked as 0, which corresponds to target 0. Additional targets should be marked with their ids. Below you can find an example of a map file (line numbers are not part of the file contents):

```








                             - - - -     - - -
                            | |     |   |     |
                        0- -   - - -  2-     -
                        | | |   |   | |     |
                           -   -     - - - - -
                        |     |     | |       |
                         - - -   -   -       -
                        |       | |         |
                       -   - - -       -     -
                      | | |       |   | |     |
                       - -         -1-   - - -


```

To check if the path file is correct, you can use the `planning_score.awk` script, executing, in the `simulator` directory:

awk -f planning_score.awk planning.out ≪***path-to-your-path-file***≫

The path file is a text file that contains, in each line, the x and y coordinates of the center of the cells that are visited by the path. Coordinates are relative to the starting position and are measured in robot diameters. The starting and ending coordinates should always be `0 0`. Below you can find an example of a path file (line numbers are not part of the file contents):

```
0 0
2 0
4 0
4 2
6 2
8 2
10 2
12 2
12 0
12 -2
14 -2
14 0
14 -2
16 -2
18 -2
20 -2
22 -2
22 -4
20 -4
20 -6
22 -6
22 -8
20 -8
18 -8
16 -8
16 -6
14 -6
14 -8
12 -8
10 -8
10 -6
10 -4
8 -4
8 -6
6 -6
4 -6
2 -6
2 -8
0 -8
0 -6
0 -4
0 -2
0 0
```

5 Assessment

The assessment of this assignment will be composed of 2 components: an execution test of the agent and a presentation.

- The agents will be tested and graded by teachers, in batch mode, in their own computers. Thus, the format specified for their execution is mandatory. The agent must generate a file representing the map of the maze, generate a file indicating the best path computed and finish its run in the starting spot.
- Each group will make a presentation of its work, consisting of a 10-minutes oral presentation (including discussion time), based on PDF slides, made available to the evaluation board beforehand.

6 Deliverables and deadline

- Source code of the developed agent. The source code should be contained in a folder called **agent** that includes:
  - all the source code of the agent;
  - an executable version of the script **run.sh** that starts the agent for C4 and supports the options presented in the following example:
    ./run.sh -c 4 -p 0 -r myagent -h 127.0.0.1 -f fname
    where `fname` is the name of both the mapping and path files.
  - an executable version of the script **build.sh** that allows to build (compile, ...) your agent.
- Presentation (in PDF format).

Source code and presentation must be submitted in the Moodle's course page. The following dates apply:

- Source code: December 30th, 2022, by 23:59.
- Presentation: January 4th, 2023, by 23:59.

7 Bibliography

- "Principles of Robot Motion: Theory, Algorithms, and Implementations", Howie Choset et al., MIT Press, Boston, 2005.
- "Introduction to Autonomous Mobile Robots", Second Edition, Roland Siegwart et al., MIT Press, 2011.
- "Artificial Intelligence: A Modern Approach", 3rd edition, Russel and Norvig, Pearson, 2009.