Universidade de Aveiro

DETI - Departamento de Eletrónica Telecomunicações e Informática

Robótica Móvel e Inteligente

2020/21

Assignment 1

Pathfinder solver using the CiberRato simulation environment

Objectives

In this assignment each group should develop a robotic agent to command a simulated mobile robot, that:

- Navigates through a maze-like arena to find the target position
- Returns back to the start position through the shortest path.
- At the end, saves the estimated map and the path used to return.

Project description

The project is organized in a sequence of challenges, of increasing difficulty, involving different navigation skills.

The list of challenges is the following:

1. Planning: The robot is provided initially with the map of the labyrinth, its initial position and the target position. It may use a GPS sensor without any noise, providing its ground-truth pose (x, y, θ). The robot should visit the target position and return in the least possible time.
2. Mapping and Planning: The robot may use a GPS sensor without any noise, providing its ground-truth pose (x, y, θ), but the map is unknown. The robot should use mapping techniques to build an internal representation of the map of the labyrinth. It should be noted that the environment is quite structured, as, for example, walls can only appear (or not) at known intervals. The robot should search target zone, while building the map of the labyrinth and return in the least possible time (which should involve planning). The map of the environment discovered by the robot should be saved at the end.
3. Localization, Mapping and Planning: The robot may not use a GPS sensor. It should estimate its position in the environment using the obstacle sensors, the compass, its actuation orders and the internal model of the labyrinth that it incrementally builds. It should be noted that the environment is quite structured, as, for example, walls can only appear (or not) at known intervals. The robot should search target zone, while building the map of the labyrinth and return in the least possible time (which should involve planning). The map of the environment discovered by the robot should be saved at the end.

The script `startChallenge_1_2` starts the simulator (and the viewer) in the configuration of Challenges 1 and 2 and the script `startChallenge_3` starts the simulator (and viewer) in the configuration of challenge 3. The difference between starting challenge 1 and 2 is that the agent will be started with the `-m lab.xml` command line option in challenge 1 but not in challenge 2.

Environment

The CiberRato simulation environment will be used to test and assess the developed agent.

The simulated robot is equipped with 2 motors (left and right) and 3 leds (visinting, returning and finish). Its sensors include a GPS, 4 obstacle sensors and a ground sensor.

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 7-cells tall and 14-cells wide. Cells may be referenced by their position in the labyrinth where cell (0,0) is located at the bottom left corner and cell (6,13) is located at the top right corner.

A maze is defined by putting thin walls between adjacent cells. The target cell is detected by the ground sensor. Figure 1 shows a possible scenario.

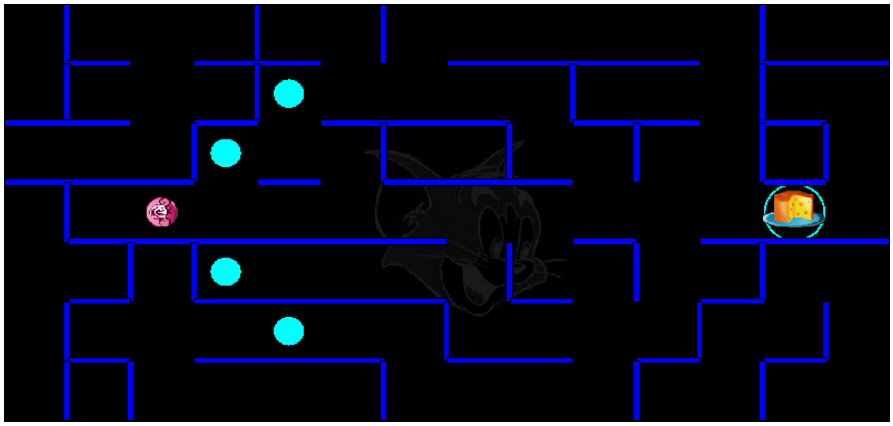

Figure 1: Scenario of the assignment: the pink robot should find the cheese (target) and return to start position,

Some scenarios are defined by XML files that are contained in the `Labs/PathFinder` folder. The groups should test their agents in new scenarios that stress the different capabilities of their agents.

The XML file that defines the scenario of the previous scenario contents is the following:

```
<Lab Name="Default LAB" Height="14" Width="28">
      <Beacon X="25" Y="7.0" Height="4.0"/>
      <Target X="25" Y="7.0" Radius="1.0"/>
      <Row Pos="12" Pattern="  |        |     |                 |     "  />
      <Row Pos="11" Pattern="  ·--·  ·--·--·  ·  ·--·--.--·--·  ·--·--"  />
      <Row Pos="10" Pattern="  |        |              |        |     "  />
      <Row Pos="9"  Pattern="--·--·  ·--·  ·--·--·--·  ·--·--·  ·--·  "  />
      <Row Pos="8"  Pattern="        |        |     |     |     |  |  "  />
      <Row Pos="7"  Pattern="--·--·--·  ·--·  .--·--·--·  ·  ·  ·--·  "  />
      <Row Pos="6"  Pattern="  |                                       "  />
      <Row Pos="5"  Pattern="  ·--·--·--·--·--·--·  ·  ·--·  ·--·--·--"  />
      <Row Pos="4"  Pattern="     |  |              |     |     |     "  />
      <Row Pos="3"  Pattern="  ·--·  ·--·--·--·--·  ·--·  ·  ·--·  ·  "  />
      <Row Pos="2"  Pattern="  |                 |           |     |  "  />
      <Row Pos="1"  Pattern="  ·--·  ·--·--·--·  ·--·--·  ·--·  ·--·  "  />
      <Row Pos="0"  Pattern="  |  |           |           |     |     "  />
</Lab>
```

The agent may be developed in C/C++, Java or Python. The CiberRato Environment

includes libraries that provide adequate APIs for communication between the agent and the simulator in these programming languages, and a parser for labyrinth files that can be used for challenge 1.

Work plan

To facilitate the development of the agent, a set of appropriate basic behaviors (coping with moving towards a point, rotation, obstacle avoidance, wall following, etc.) should be implemented.

1. To accomplish the first challenge, a search algorithm (such as A*) should be applied to the map in order to find the shortest path from the start position to the target position and to return back. The robot skills of moving to a point and rotating are of particular interest here. It should be noted that if the robot always goes straight from the center of a cell to the center of an adjacent (reachable) cell, the obstacles sensors may be unnecessary. More optimized paths are also welcome.

2. To accomplish the second challenge, the robot should build an internal model of the map of the labyrinth. Here, the obstacles sensors and the compass will be very important. It should be noted that the environment is quite structured, as, for example, walls can only appear (or not) at known intervals. In a first stage, the robot should search the environment for the target position, trying to explore all unvisited cells. Having found the target position, and if no additional search is required to find the optimal return path, a search algorithm (such as A*) should be applied in order to find the shortest path from the target position to the start position.

3. To accomplish the third challenge, the robot should keep a good estimate of its location within the environment at the same time as it builds its internal model of the map of the labyrinth. The robot has to find the target position without losing accuracy in the estimate of its location and then plan and execute the path to the start position.

4. The final report should describe the approach used in the resolution of the previous points. The report should describe the overall architecture of the agent, the implemented behaviors and how they can be combined to solve the problem, and the results obtained by the agent. It should also contain references to documents (articles, books, reports, etc.) that supported the development of the work.

   This report should be written using the LNCS template available at the following link: http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0).

Implementation

- When reaching the target area, the robot must turn on its 'visiting led'.
- Before starting the returning phase, the robot must turn on its 'returning led' while it is still inside the target cell.
- The ground sensor can be used to sense the target cell.
- Only the following sensors may be used in the final demonstration of challenge 3: ObstacleSensors, GroundSensor, Compass, Bumper and Time. Challenge 1 and

2 may additionally use the GPS Sensor without noise. The noise configurations should be the same as those provided in the source code for each challenge.

- The initial position of the robot will always be centered in one cell and horizontally aligned pointing towards the right (virtual North).
- The agent should be able to use one of the following command line formats:

```
./robsample -h host_or_ip -m lab.xml -p 1 -s 3,2 -t 3,12 -c 1
./GUISample -h host_or_ip -m lab.xml -p 1 -s 3,2 -t 3,12 -c 1
java jClient -h host_or_ip -m lab.xml -p 1 -s 3,2 -t 3,12 -c 1
python pClient -h host_or_ip -m lab.xml -p 1 -s 3,2 -t 3,12 -c 1
```

  to connect to a simulator that is executing in a different computer (-h), read the map description from *lab.xml* (-m), start at position 1 in the grid (-p), which is located in cell (3,2) (-s), having the target at cell (3,12) (-t) and going to execute challenge 1 (-c). In the final demonstration all agents will be started using these options, hence it is very important to be prepared for them.
- The estimated map should be saved in a file called “map” (with an optional extension) that represents the estimated maze map (only in challenges 2 and 3). This representation should clearly show all cells that have been visited by the robot and also all walls separating cells that have been detected. The planned path for the return should also be represented. The representation may be performed in textual-manner of graphically.

Movement Model

Consider that the robot 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_{left_t}, in_{right_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_{left_t} and in_{right_t}) that models the inertial characteristics of the motors and generates the effective powers that will be applied to the motors:

$$out_t = (\text{in}_t * 0.5 + out_{t-1} * 0.5) * noise$$

Then, the movement is separated in a translation of the robot position, considering its current orientation, and the change of the orientation of the robot.

For translation, we have:

$$lin = \frac{out_{\text{right}} + out_{\text{left}}}{2}$$

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

For rotation, we have:

$$rot = \frac{out_{\text{right}} - out_{\text{left}}}{robotDiam}$$

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

This provides the new robot pose (x_t ,y_t, θ_t) at the next step.

Presentation

Each group will make a presentation of its work which consists of three parts: an oral presentation, based on powerpoint or similar (10 minutes, maximum); a demonstration of the performance of the robot agent in some unknown scenarios and finally a short discussion of the work.

Deliverables

- Source code of the developed agent
- Report (in PDF format; according to Springer LNCS paper template)
- Presentation (in PDF format).

All deliverables must be available at the group repository at the code.ua platform after the relevant “Important date” (see below).

Evaluation

- Robotic agent (structure and performance): 45%
- Report: 25%
- Presentation: 30%

Important dates

- Submission of the deliverables: November 18, 2020 (1pm)
- Presentation: November 19, 2020 (during class)

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