universidade de aveiro

Intelligent Mobile Robotics

Assignment 1

Robotic challenge solver using the CiberRato simulation environment

Diogo Pereira de Jesus, 97596
Pedro Miguel Tavares Rodrigues, 92338

2023/2024 **DETI**

C1 – Control challenge

universidade de aveiro

Control loop:

1) Read Sensors.
2) **Filter values** read from line sensor.
3) Check if robot is **above a checkpoint**.
    -) If true and the **checkpoint detected is not the expected**:
        1) **Turn around** the robot.
        2) End.
4) Traverse the line.
    -) If **a curve is detected**:
        1) Continue to traverse the line **without changing its direction**.
        2) **Store the values to apply to the motors** once the line is not detected.
    -) Else if **no curve is detected**:
        1) **Compute the values to be applied to the motors** through the controllers (one for each motor).
        2) Apply the values to the motors.
    -) Else if the **line is only detected on one side of the robot**:
        1) **Rotate the robot**.
    -) Else if **no line is detected**:
        1) Apply the **previously stored values** to the motors.

C1 – Control challenge

universidade de aveiro

Controller:

- Standard implementation of a PID controller. Similar to the implementation showed in the theoretical classes.
- One controller for each motor (for more precise corrections without loosing has much speed).
- The controller was tuned manually using the Ziegler-Nichols method.

Ziegler-Nichols method

Control type	K_p	K_i	K_d
P	$0.50K_u$	—	—
PI	$0.45K_u$	$0.54K_u/T_u$	—
PID	$0.60K_u$	$1.2K_u/T_u$	$3K_uT_u/40$

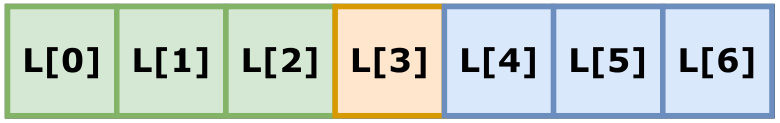
Figure 1: Retired from Wikipedia.

C1 – Control challenge

universidade de aveiro

Line Filtering:

- The line sensor array is divided in two (left and right side).



- The middle value is used to filter both sides.
- The filter will check the number of values set to "1" in the sub array.
- If the number of "1"s is one, the following changes will be applied:

Input	Output
"0100"	"1100" if the complete line is "0110100" else "0000"
"0010"	"0000"
"0001"	"0000"

Table 1: Filtering applied to the right side of the line sensor return values. The left side filtering is the same but reflected.

C1 – Control challenge

universidade de aveiro

Line Filtering:

- If the number of "1"s is two, the following changes will be applied:

Input	Output
"1010"	"1110" if the complete line is "XX11010"
"1001"	"1000"

Table 2: Filtering applied to the right side of the line sensor return values. The left side filtering is the same but reflected.

C1 – Control challenge

universidade de aveiro

Robot position over the line:

- Based on the implementation to the one presented in the theoretical classes.
- Calculated separately to the left and right sides of the robot.
- The line sensor represents a negative distance (-0.08) to achieve a value of 0 when the robot is centered in the line (00111100) and to help the controller increase the velocity of a motor when needed.

$$e = r - y$$

The reference value is 0 for both controllers.

C1 – Control challenge

universidade de aveiro

Results:

- The score on each run will depend on the amount of times a wrong turn caused by noise **is not filtered** and will trigger a rotation of the robot. One example is when **two bits** are affected by noise.
- This will result in the robot traversing the line in the **wrong direction** until it reaches a checkpoint.
- On 10 runs the average score was **10051** *(CycleTime=10)* for the first map provided.
- On 10 runs the average score was **9675** *(CycleTime=10)* for the second map provided.

C2 – Mapping challenge

universidade de aveiro

Cell map structure:

- Map that has the **coordinates of the center of a cell** as the **key** and a **tuple** containing a **list of its neighbor cells** and a **boolean variable** that indicates if the **cell was already traversed** as the value.

```
{
   (x,y) : ([(x2,y2), (x3,y3)], False)
}
```

C2 – Mapping challenge

universidade de aveiro

Control loop:

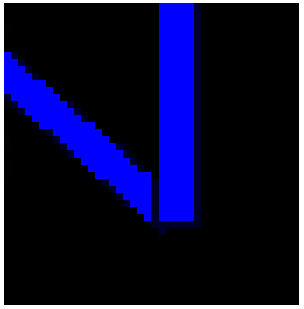
1) Read Sensors.
2) Update the perception of the robot (it's **current position** and **direction**, the **closest center of a cell to the line sensor** and if the **line sensor is before or after that center**).
   -) If the **closest cell coordinates** equal to the **next cell coordinates**:
      1) **Detect curves** with the **line sensor**.
      -) When detection ends:
         1) Convert the **relative direction** of the curves (left, soft right, etc) to **absolute directions** (south, northeast, etc).
         2) Compute the **coordinates of the neighbor cells** based on the **detected curves** and add them to the cell map, updating the current cell neighbors list.
         3) **Decide** which **path to follow** (update next cell). End if **all cells were traversed**.
3) **Traverse** the line.
   -) If cell center **different** than next cell:
      1) **Rotate** until current direction **equal** to the computed direction.
         -) If no line is available at the computed direction:
            1) Remove the perceived neighbor from the cell map.
            2) Update the next cell.
   2) **Traverse the line** using the controller to correct the path (similar to the first challenge).

C2 – Mapping challenge

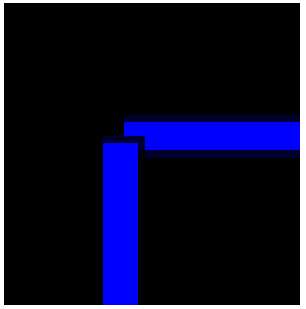
universidade de aveiro

Curve detection:

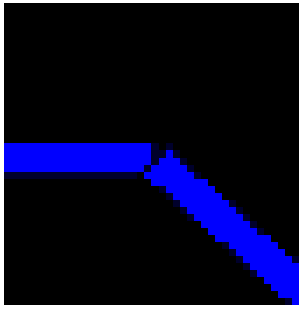
- Analyzing the **line sensor**.
- If a deviation of the line to one side is detected **before traversing the center of a cell**, a **V shaped curve is detected** to that side.
- **After** the line sensor **traverses the cell center**, the distance to the line is **compared** to check if the detected curve is a ***P* (perpendicular)** or a ***S*(soft curve)**. A *P* cannot be detected **after** an *S* to the same side.



V Curve



P Curve



S Curve

C2 – Mapping challenge

universidade de aveiro

Decision:

- To decide the next cell we simply **analyze the neighbors** of the **present cell**. If one of those was **not traversed**, it is selected as the **next cell**.
- If all its neighbors are already expanded we **search for a cell** that is **yet to be expanded** on the **cell map**. Then we use the **A* algorithm** to compute a **path** from the present cell and store it on next.
- In the middle of the computed path traversal a **cell neighbor might be available to be expanded**. In that case we **cancel** the computed path and **traverse to that neighbor**.
- If all the cells available on the cell map **are expanded**, we end the search, compute the map and store it to a file.

C2 – Mapping challenge

universidade de aveiro

Writing map to a file:

- Use of a **matrix** to represent the file inside *Python*.
- The coordinates for each strip are computed using the **cell coordinates** and the **neighbor coordinates**.

```
if neighbor_x > cell_x && neighbor_y == cell_y: # to the right
    line = cell_line
    column = cell_column + 1
    matrix[line][column] = "-"
```

Example of a strip being stored on the matrix representing a horizontal line, on the right side of the cell

C2 – Mapping challenge

universidade de aveiro

Results:

- Curves that are **wrongly detected** or pass **undetected** might influence the final score.
- The amount of wrong curves detected by the agent is volatile, but is usually 0. Usually, the agent can also correct these cases.
- **Undetected** curves can lead to **parts of the map not being explored**.
- On 10 runs, the average map score was **528** *(CycleTime=10)* for the map provided.

C3 – Planning challenge

universidade de aveiro

- This challenge uses the **same approach** of challenge 2.
- It explores the map using the **same algorithms**. The only addiction is the **detection of checkpoints** and the storage of its **coordinates and value** (1,2,3).
- When the map is explored or the simulation time has ran out, the **A* search algorithm** is used to compute the **shortest path between each checkpoint**.
- The path is then stored in a file. The relative coordinates are calculated **subtracting the start cell coordinates to each cell in the path**.

C3 – Planning challenge

universidade de aveiro

- **Results:**
  - The limitations for this challenge will be **similar** to the ones **presented in challenge 2**.
  - The main source of error comes with the **amount curves that go undetected**. This might influence the final score.
  - On 10 runs, the average planning score was **0.97** *(CycleTime=10)* for the map provided.