Introduction to autonomous mobile robot

Project 1 - Line following

Group 7

Saugy Alexandre, Angeles Raphaël, D'Amico Timothée
Why would we do this project?

- Monitoring of physical parameters mostly done automatically
  > need of mobile autonomous robots (driving, moving in water, flying,...)

- For instance, may be applied to following a gradient of pollution
What was our “eyes” on the field?

- Camera 2 kpx, located in the front of the robot
- Angular position sensor on the motor of each wheel
- 8 Infra-red distance sensors
What were the problems we had to overcome?

- Noise
- Change in colors and shapes of the line
- Different colors of the objects edges
Controller design
Controller design

run_again pointer

- **0** is the starting value and means the e-puck has to study its environment.

- **1** indicates the e-puck is lost and has to find (or find back) the line.

- **2** is the case where the robot is on the line and follows it.

- **3** is used when an obstacle is detected and a bypass has to be initiated.
Controller design

- **first run of the main loop**
  - At start, run_again=0

- **lost_fun**
  - run_again=1

- **go_to_the_line_and_align**
  - run_again=1

- **alignment**
  - run_again=2

- **avoid/get_around_obstacle_when_lost**
  - run_again=0

- **is_e-puck_aligned**
  - run_again=1

- **follow_the_line**
  - run_again=3

Legend:
- run_again=0 - initialisation
- run_again=1 - case where the e-puck is lost
- run_again=2 - case where the e-puck stands on the line
- run_again=3 - case where the robot encounters an obstacle
Controller design

go_to_the_line_and_align

follow_the_line

alignment
Experiments
Experiments

Reference World

Black Oval

Black Square

Black Half-square

Black “S”

Blue “S”
Experiments

Up to 200 tests (at least 35 for each world) are launched from different situations and starting points.

- Departure on the line (clockwise and counterclockwise).
- Departure outside of the line.
- Departure from a critical point (only for the oval world).
Experiments

The results can be classified in 4 categories

- Almost perfect.
- Few light problems.
- Some consequent problems.
- Nothing really interesting (in one minute).
Results of the tests
Results

Overall comparison between worlds

![Bar chart showing comparison between different worlds. The chart indicates the number of tests and the categories of results for each world: Almost perfect, Few light problems, Some consequent problems, Nothing really interesting (in one minute).]
## Results

**Evaluation of the different setups**

*black oval world*

<table>
<thead>
<tr>
<th>Number of obstacles</th>
<th>Departure</th>
<th>Almost perfect</th>
<th>Few light problems</th>
<th>Some consequent problems</th>
<th>Nothing really interesting (in one minute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>On the line clockwise</td>
<td>0</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>On the line counterclockwise</td>
<td>0</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>On the top left corner</td>
<td>0</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>On the top left corner (critical point)</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>On the line clockwise</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>On the line counterclockwise</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>On the top left corner</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>On the top left corner (critical point)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>On the line clockwise</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>On the line counterclockwise</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>On the top left corner</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>On the top left corner (critical point)</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Positive points about the code

- Almost never stuck, lots of exit doors -> great autonomy
- Most of line curvature detection are correct
- Simple main structure of the code
- Black wall or black obstacle detection is usually correct, no confusion with line
Results

Black oval world - critical situation
Results

More details about the main problems after many tests in the different worlds

- The e-puck occasionally touches a bit the obstacle in angles
- Sometimes going along the black wall taking it for an obstacle
- The bypass function is not always optimal, depending on the line orientation below the obstacle to get around
- Detection of front pathline not always optimal if “sharp turn” or other line beside the robot
Results

“S blue” world recurring problem
Results

How to improve the code?

- avoid global variables -> more local pointers
- use only one main loop -> change code structure
- odometric data may be useful when stuck
- test a lot more different situations and calibrate the e-puck accordingly
- use a larger zone on the lower side of the camera to detect the pathline -> improve the precision
Odometry

- Absolute position given by supervisor: 73.520000
- Coordinate difference between the given one of the supervisor and the computed on the X axis:
  - The maximal error between the true coordinate and the computed one is 14.6% of the total distance traveled on the X axis.
  - At final state, the difference on the X axis is 0.55 [m].
- Coordinate difference between the given one of the supervisor and the computed on the Z axis:
  - The maximal error between the true coordinate and the computed one is 16.5% of the total distance traveled on the Z axis.
  - At final state, the difference on the Z axis is 0.29 [m].
- Relative computed based on values of e-pack.
Conclusion

- The bigger the code gets, the more difficult it becomes to correct.
- Would have been useful to have underneath camera.
- Challenging to have a code working with all different worlds.
- Necessary to implement the code on a real robot to deal with real light variation.