SIS project:

Road sign recognition

By Karni Siraganyan, Florence Cuttat & Félix Fouret
Outline

1. Introduction
2. Experiments and algorithm
   a) Obstacle detection & placement
   b) Take picture and filter image
   c) Analyze Picture and action
3. Results
4. Conclusion
5. VIDEO
Why is road sign recognition interesting?

- Thanks to this technology, robots can react according to their environment. They have a better autonomy. Cars, for example, can drive without driver.
Introduction

What is the project about?

• Get in touch with robot (E-puck) programming using C language.

• Learn how process different types of signals using sensors.
Introduction

Sensors used for our project

The E-puck
Introduction

Sensors used for our project

Proximity sensors

- Find when there is a road sign
- Place itself according to the road sign
- Avoid hitting the walls when driving around

The E-puck
Introduction

Sensors used for our project

Proximity sensors

• Find when there is a road sign
• Place itself according to the road sign
• Avoid hitting the walls when driving around

Camera

• Take the picture
• Process the image using pixels management and fft
• React according to image
Introduction

Sensors used for our project

Proximity sensors
- Find when there is a road sign
- Place itself according to the road sign
- Avoid hitting the walls when driving around

Camera
- Take the picture
- Process the image using pixels management and fft
- React according to image

Take action
- Turn left
- Turn right
- Turn around

The E-puck
Experiments and algorithm

a) Obstacle detection & placement

Proximity sensors

**ps0 & ps7**: Used to detect the wall coming in front.

**ps1 & ps6**: Used to detect wall coming a little from the side → Repositioning

**ps0 to ps7**: Avoid wall using Braitenberg coefficient technic
Experiments and algorithm

b) Take picture and filter image

E-puck well placed  ➔  Take picture

(Image only 40 x 40 pixels because of memory limitations)
Experiments and algorithm

b) Take picture and filter image

E-puck well placed \[\rightarrow\] Take picture

(Image only 40 x 40 pixels because of memory limitations)

Filter picture (The original is flipped 90° to the left)
Experiments and algorithm

Filter

Why?
- To enhance contrast.

How?
1. Take the 256 first pixels of columns or rows.
2. Calculate the mean value of them.
3. Run through the same vector again
   - if (pixel value < mean) → pixel value = 0 (black)
   - else → pixel value = 255 (white)

With this technic → problem with black

Introduction of a threshold

if (pixel value < mean + 17) → pixel value = 0 (black)
else → pixel value = 255 (white)
Experiments and algorithm

c) Analyze Picture

Do the fft of the two filtered vectors (rows and columns) of size (256 x 1).
Experiments and algorithm

• Make a *for loop* to find the higher amplitude of the FFT in the bandwidth 40-180 Hz.

• Subtract rows maximum amplitude value to columns maximum value.

\[
\text{Difference} = \text{AmplitudeMaxColumns} - \text{AmplitudeMaxRows}
\]
Experiments and algorithm

Distinguish 3 different cases

Case 1

If difference > 3000
Vertical stripes → Turn right

Case 2

Else if difference < -3000
Horizontal stripes → Turn left

Case 3

de else
Black image → Turn around
Tests situation:

- Local: Kitchen in front of the laboratory.

- Light conditions: Day Light, Artificial Light, Lights off by closing the curtains and Flashing lights by moving the curtains.

- 2 Different paths: Daylight Ahead/Behind at the starting point.

- If the e-puck failed a road sign the first time there is a BLACK picture at the opposite of this image to turn around and try again.

- Two tries per case and for each case we stop the experiment if it does more than two mistakes.
### Results

<table>
<thead>
<tr>
<th>Stripes</th>
<th>Natural Light Front</th>
<th>Artificial Light</th>
<th>Lights off</th>
<th>Flashing Lights</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Test 1</td>
<td>Test 2</td>
<td>Test 1</td>
<td>Test 2</td>
</tr>
<tr>
<td>H</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>H</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>B</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
</tbody>
</table>

Mistakes: 0, 0, 0, 0, 0, 2, 1, 0, 0

**Total maze 1**

Success: 91.5%
Fail: 8.5%
Results

<table>
<thead>
<tr>
<th>Stripes</th>
<th>Maze 2</th>
<th>Natural Light Back</th>
<th>Artificial Light</th>
<th>Lights off</th>
<th>Flasing lights</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Test 1</td>
<td>Test 2</td>
<td>Test 1</td>
<td>Test 2</td>
</tr>
<tr>
<td>V</td>
<td></td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>V</td>
<td></td>
<td>X</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
</tbody>
</table>

| Mistakes |        |        |        |        |        |        |        |        |        |
|          | 1      | 0      | 0      | 0      | 3      | 1      | 0      | 0      | 0      |

**Total maze 2**
- Success: 86.5%
- Fail: 13.5%

**Total both mazes**
- Success: 89.3%
- Fail: 10.7%
Problems

1. White wall looks like black wall
   (→ E-puck can get stuck)

2. Waste of time at the end

3. No placement if proximity sensor 0 or 7 detects
   the object before proximity sensors 1 or 6

4. Buggs (re-initialization of code after picture)
Problems video
Conclusion

What did we learned??

1. Being more comfortable with C language
2. Face computation problem
3. Manage E-puck memory
4. First contact with robotics
5. Handle data coming from sensors
6. Deal with noise
7. How FFT can be useful to analyze data
Finally it works well