e-puck road sign recognition

S. Vitali, X. Tendon, R. Gitton
Supervisor : F. Rahbar
Introduction

Sample picture

Complex picture
Webot FFT strategy

Horizontal image

Vertical image
Webots Result

- Basic road signs : total success !
  
  => No failure on more than 300 tries even with an angle (~30°)

- Striped cube : 100% success rate

- Special cube : 100% for little angle.
Webot: Hard maze video

Webotmaze.mp4
E-puck: Main step in coding

• Develop tools to study the code

  => sprintf(), cat/dev/… , matlab graphics

• Develop 2 codes:
  - FFT code
  - Non-FFT code
E-puck: Main step in coding (2)

1° Moving and Wall detection.

2° Take a photo

3° Selection of two rectangles (32*8 and 8*32) and sum along the side to get two vectors (32*1 and 1*32)

4° Make FFT along row and column (not for the code without FFT)

5° Stand conditions:

<table>
<thead>
<tr>
<th></th>
<th>FFT</th>
<th>Without FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left or right</td>
<td>sum(mag_col) &gt; sum(mag_row)</td>
<td>diff(max_col-min_col) &gt; diff(max_row-min_row)</td>
</tr>
<tr>
<td>White or black</td>
<td>abs(diff(max_col-min_col) - diff(max_row-min_row)) &lt; X</td>
<td></td>
</tr>
</tbody>
</table>

6° Move according to the decision

7° Calculate the right angle with function \text{wait}()
E-puck code - Result of the $32 \times 1$ vectors

Easy picture // Taken at 1-2 cm

Hard picture // Taken at 8-10 cm
E-puck code - Result of the FFT

32*1 vectors

Magnitude of the FFT
## E-puck: Results

### Simple picture // Normal conditions

<table>
<thead>
<tr>
<th>Image type</th>
<th>Vertical stripes</th>
<th>Horizontal stripes</th>
<th>Black</th>
<th>White</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Success rate</strong></td>
<td>19/20</td>
<td>19/20</td>
<td>18/20</td>
<td>17/20</td>
</tr>
</tbody>
</table>

### Simple picture // Darker conditions

<table>
<thead>
<tr>
<th>Image type</th>
<th>Vertical stripes</th>
<th>Horizontal stripes</th>
<th>Black</th>
<th>White</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Success rate</strong></td>
<td>19/20</td>
<td>19/20</td>
<td>19/20</td>
<td>19/20</td>
</tr>
</tbody>
</table>

### Complex picture // Normal conditions

<table>
<thead>
<tr>
<th>Image type</th>
<th>Vertical complex stripes</th>
<th>Horizontal complex stripes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Success rate</strong></td>
<td>19/20</td>
<td>20/20</td>
</tr>
</tbody>
</table>
E-puck: Video performance

Hard picture detection with non-FFT Algorithm

Lab maze with FFT algorithm
Problems encountered with e-puck

- The e-puck need some time to do a while loop, to react to the code
  
  => wait(10000) at the end

- Memory limited at around 1500 arrays

- The variables need good declarations, overflow!
  
  => int ≠ double or unsigned long

- Importance of initialising the values
  
  => unsigned long sum_col_p[64]={0};

- The e-puck do more error when the battery is discharged
Conclusion and improvements

- Optimize the distance to have better results with the complex picture
- Change the conditions on the FFT
- Retake a photo when the decision isn’t sure enough