Sign Recognition

Evanice Ruegg et Alexis de Aragao

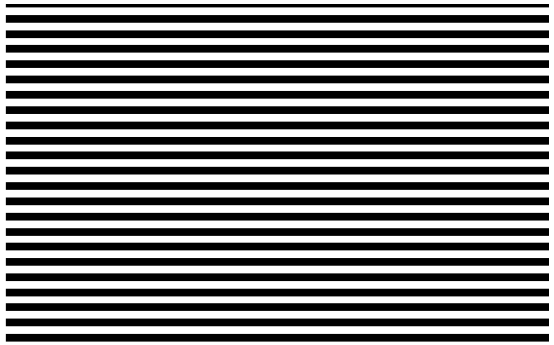
ENG-366 / SIE-BA6

INTRODUCTION

- Simulation of an e-puck robot, equipped with different kind of sensors, in webots
- No temporal synchronisation or memory physical contraints
- Environment of the robot :
  - Maze
  - Directional signs (possibly with noise)
  - External light sources

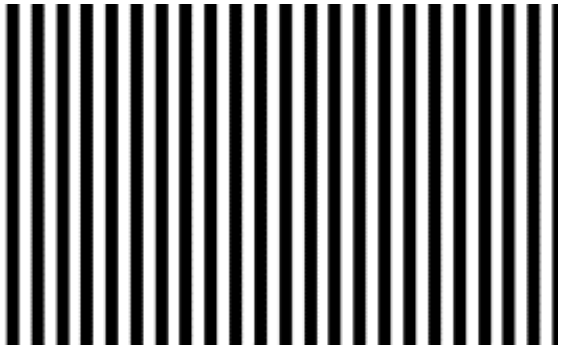
MAIN GOAL OF THE PROJECT

- Robot succeed in exiting the maze, following directional signs
- Robot rapidly exits the maze
- Code as independant as possible from the given environment



Horizontally stripped => LEFT

Black => TURN BACK



Vertically stripped => RIGHT

SIGN RECOGNITION AND FRONTAL WALL AVOIDANCE

- Signal analysis (two 1D discrete FFTs of series of sums per row and per column of the pixel values of the 52x39 picture taken by the robot)

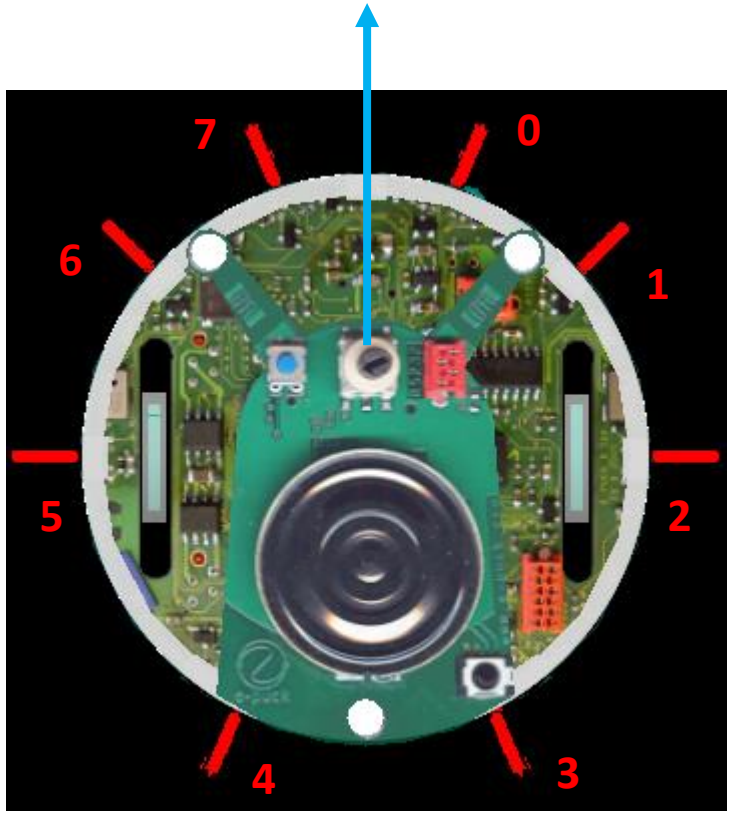
- Spectral magnitude ratio computed for decision :

$$s_{cols} = 2\sum_{i=0}^{27}\left|F_{cols,i}\right| \qquad s_{rows} = 2\sum_{j=0}^{19}\left|F_{rows,j}\right|$$

$$r = \frac{s_{cols}}{s_{rows}}$$

If $0.05 < r < 0.55$	: horizontally stripped	=> LEFT
If $1.81 < r < 20$	: vertically stripped	=> RIGHT
If $r \leq 0.05 \cup r \geq 20$	: black	=> TURN AROUND
If $r \leq 1.81 \cup r \geq 0.55$	: «1234» wall	=> TURN AROUND

REQUIRED CONDITION FOR PICTURE RECOGNITION – DISTANCE TO WALL



- Value from distance sensor ≤ 0.1 => not significantly close to a wall

```
if(
(((distance[0] + distance[7]) > 0.2) &&
(distance[1] < 0.1) &&
(distance[2] < 0.1) &&
(distance[3] < 0.1) &&
(distance[4] < 0.1) &&
(distance[5] < 0.1) &&
(distance[6] < 0.1)) = = 1
)
```

REQUIRED CONDITION FOR PICTURE RECOGNITION – FRONTAL POSITION

The previous condition lead to an accepted maximal incident angle of about 25°

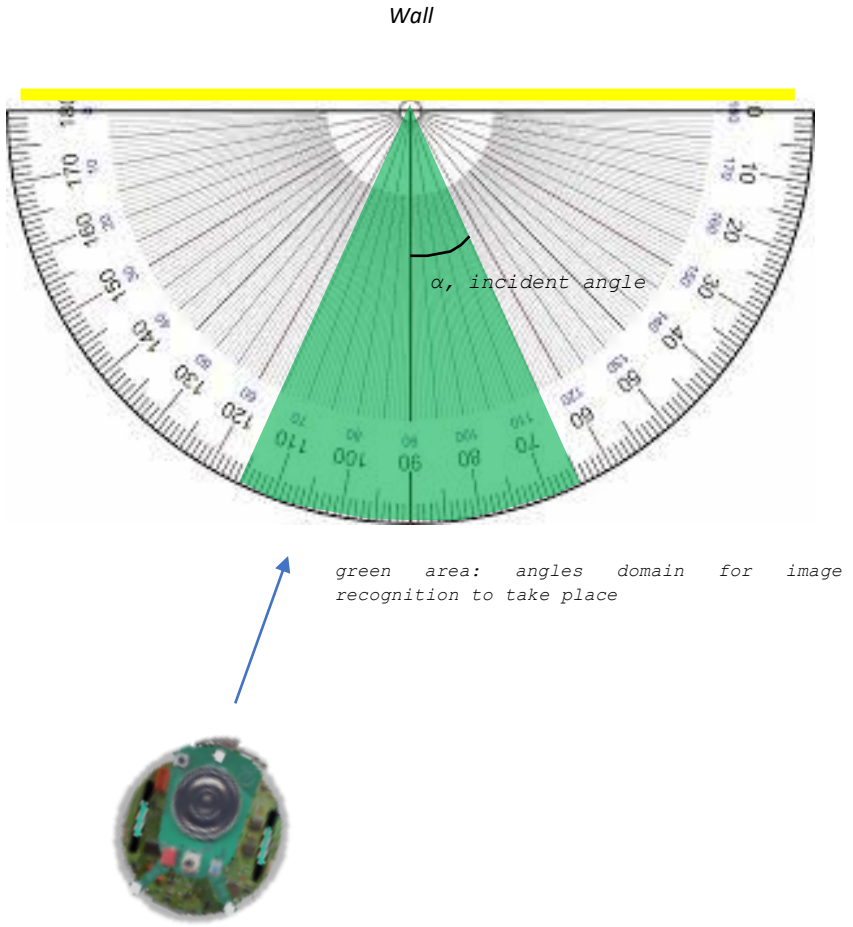


IMAGE RECOGNITION MOTIONS

1) Step back to take picture (during 960 [ms], left and right speed set to -300 [-])

2) Spectral analysis

3) Decision and associated motion :

- If must turn left or right, during 824 [ms], left and right speed set to ±550 [-] (depends on the case)
- If the robot must turn back, during 1648[ms], left speed set to 550 [-], right speed to -550 [-]

LATERAL/FRONTAL WALL AVOIDANCE - BRAITENBERG

If(
((distance[1] > 0.2) ||
(distance[6] > 0.2)) = = 1
)

Distance-Sensors	Braitenberg coefficients
Front	{-2.0,1.8} et {1.8,-2.0}
Side-front	{-1.6,1.2} et {1.2,-1.6}
Lateral-central	{0,0} et {0,0}
Rear	{0,0} et {0,0}

- Braitenberg right bypass :

Speed[left or right] = $30*\sum_{j=0}^{7}$ distance_sensors_values[j] $*$ weights[j][left or right]

(distance_sensors_values[j] set to 0 if distance_sensors_values[j] ≤ 0.2)

|Speed[left or right]| ≤ 1000 or set to 999.99 or -999.99

LATERAL-CENTRAL OR REAR WALL DETECTION AND NO DETECTION CASES

```
If(
((distance[0] < 0.1) &&
(distance[1] < 0.1) &&
((distance[2] > 0.1) ||
(distance[3] > 0.1) ||
(distance[4] > 0.1) ||
(distance[5] > 0.1) )&&
(distance[6] < 0.1) &&
(distance[7] < 0.1)) = = 1
)

speed[left] = 999.99
speed[right] = 980
```

```
If(
((distance[0] < 0.1) &&
(distance[1] < 0.1) &&
(distance[2]  < 0.1) &&
(distance[3] < 0.1) &&
(distance[4] <  0.1) &&
(distance[5] < 0.1) )&&
(distance[6] < 0.1) &&
(distance[7] < 0.1)) = = 1
)

speed[left] = 999.99
speed[right] = 999.99
```

STATISTICS (1)

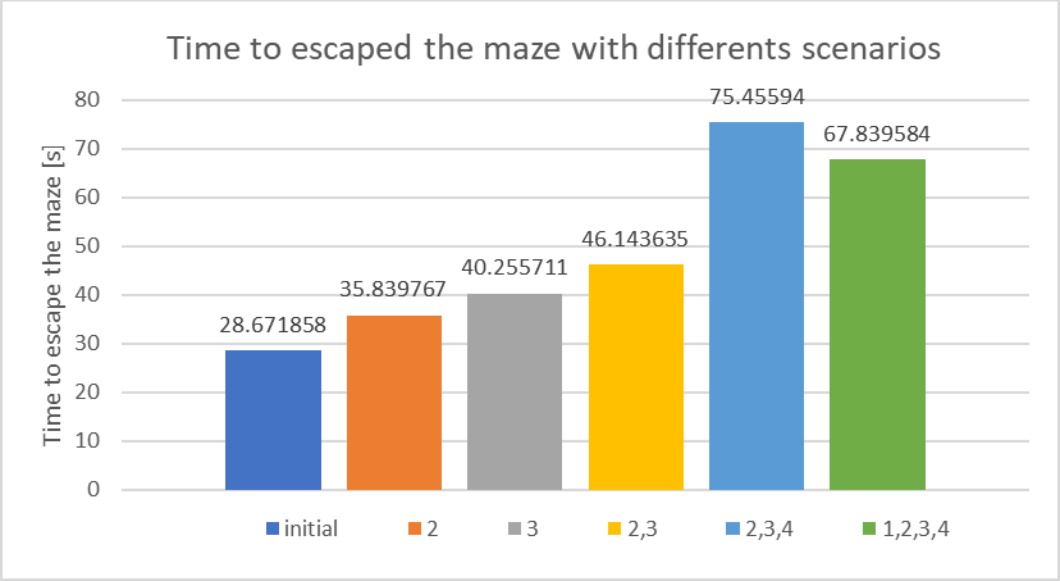
- Used to evaluate the performance of the strategy on matlab
  - With the original picture
  - With noisy pictures
- Success rate of 100%

STATISTICS (2)

- Implementation of a supervisor
- Two metrics :
  - Does the robot escape the maze ?
  - How long does the robot take to exit the maze ?
- The robot exits the maze in 28.671858 seconds with a success of 100%

STATISTICS (3)

- Different configurations of the maze :

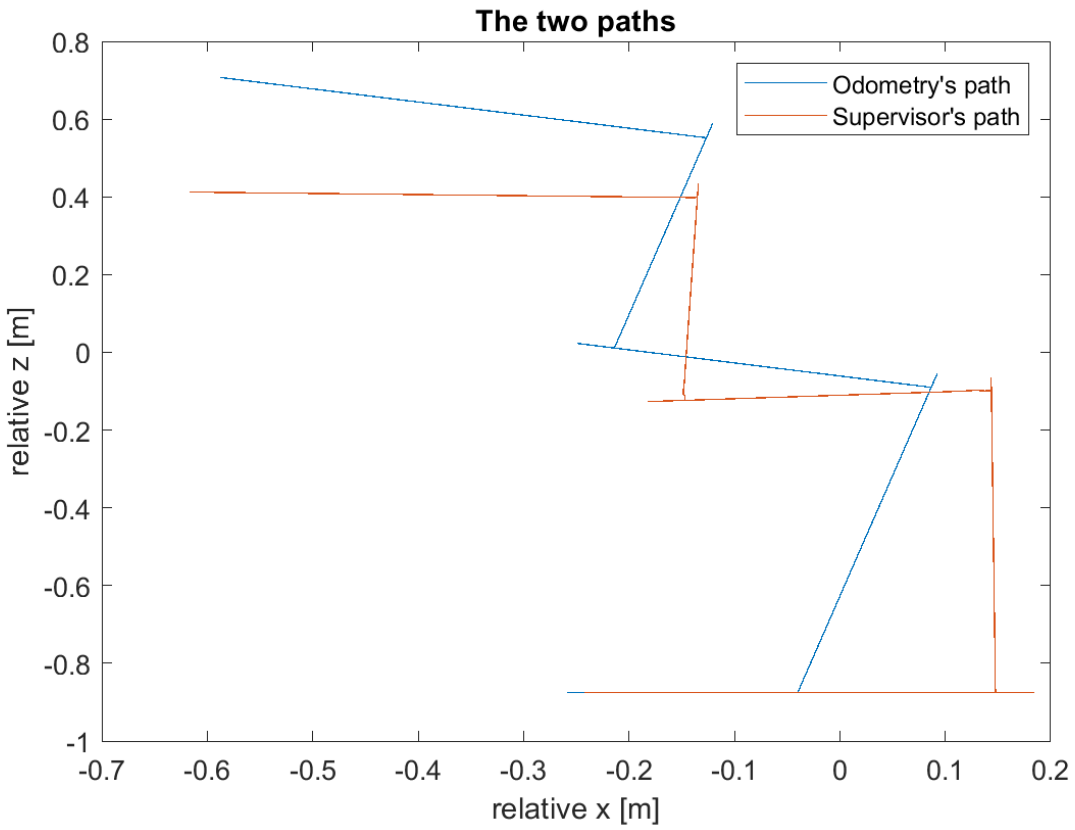


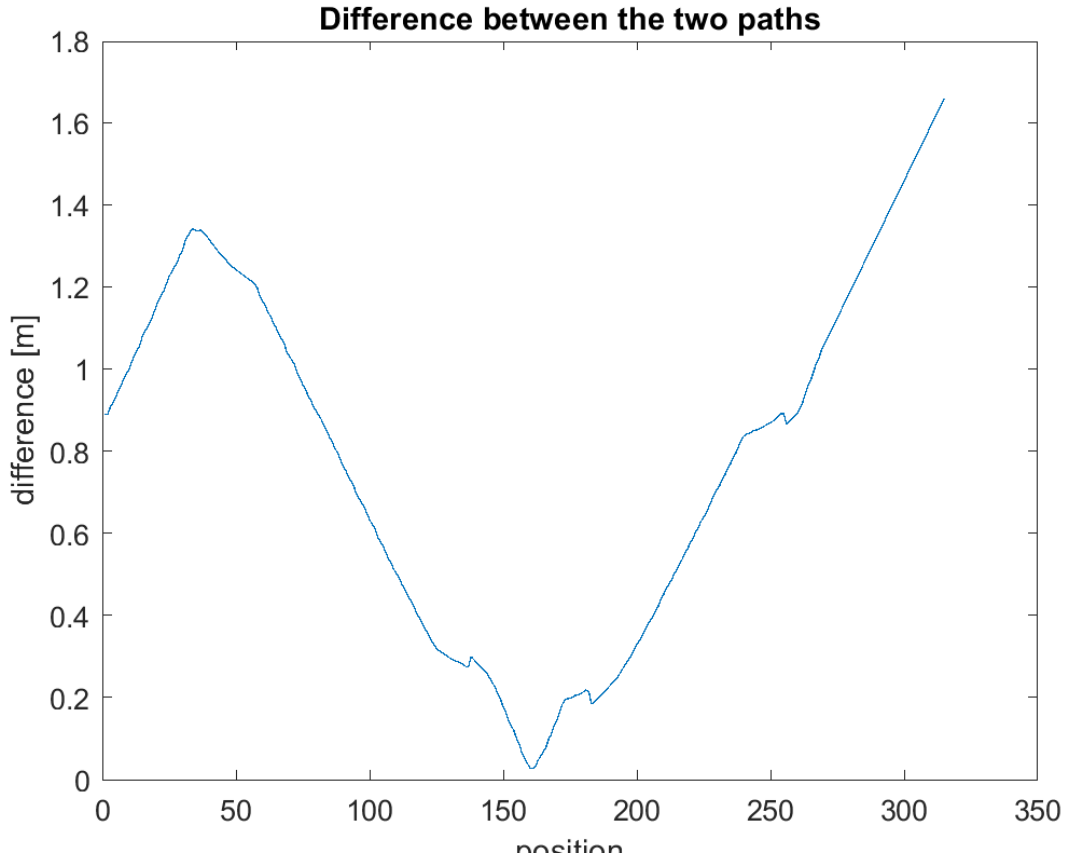
- Legend : Initial configuration, 2nd image rotated, 3rd image rotated, 2nd and 3rd rotated, 2nd 3rd and 4th rotated, all images rotated

ODOMETRY (1)

- Odometry : use of data obtained by the motion sensors to estimate the position of the E-puck
- Objectif : transform the wheel positions in the local frame of reference of the robot to the absolute frame of reference of the world
- Functions used :
  - wb_position_sensor_get_value
  - wb_motor_get_position_sensor(right_motor/left_motor)
- Two paths are obtained: one from the odometry and one from the supervisor

ODOMETRY (2)


The two paths
Odometry's path
Supervisor's path
relative z [m]
relative x [m]
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.7
-0.6
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2


Difference between the two paths
difference [m]
position
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
250
300
350

CONCLUSION

- Webots gave us the opportunity to apply the theory seen in class

- The simulation is made on webots; there are some difference with the reality

- The strategy chosen is suitable for the environment provided and for some modified conditions, it might not worked for very dissimilar environments.