Road sign recognition with an e-puck

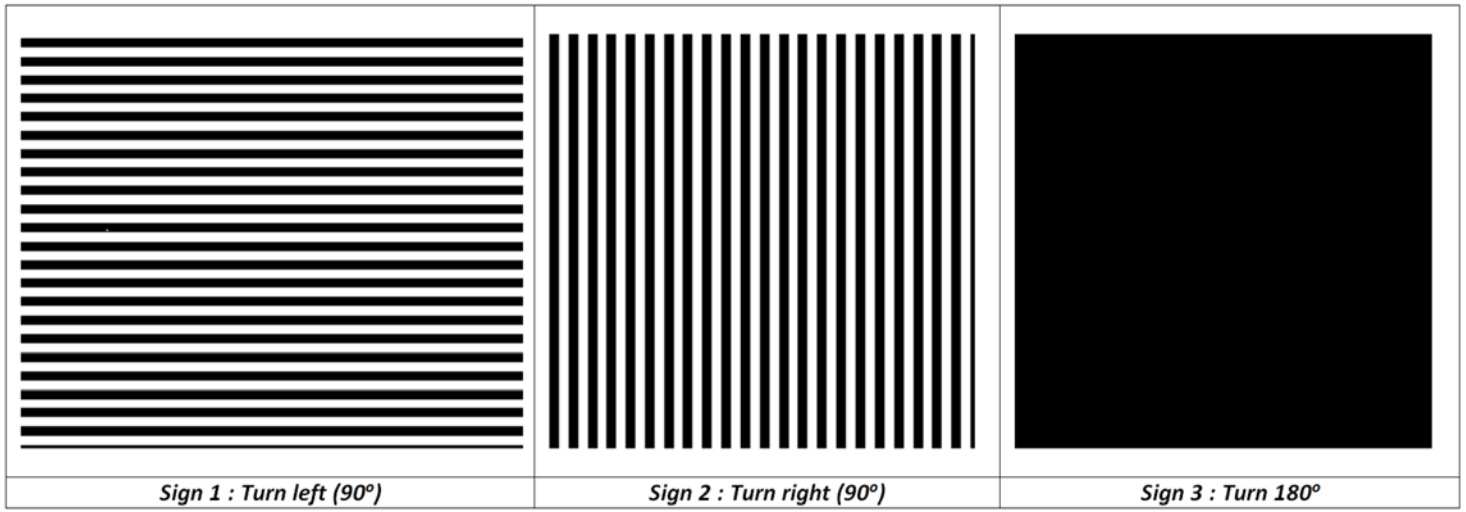
Signal, Instruments and Systems

Julien Clark, Gabriel Grosclaude, Théophile Maeder

Introduction

- Project goal

Program an e-puck to make it exit a maze by recognizing 3 signs roads.

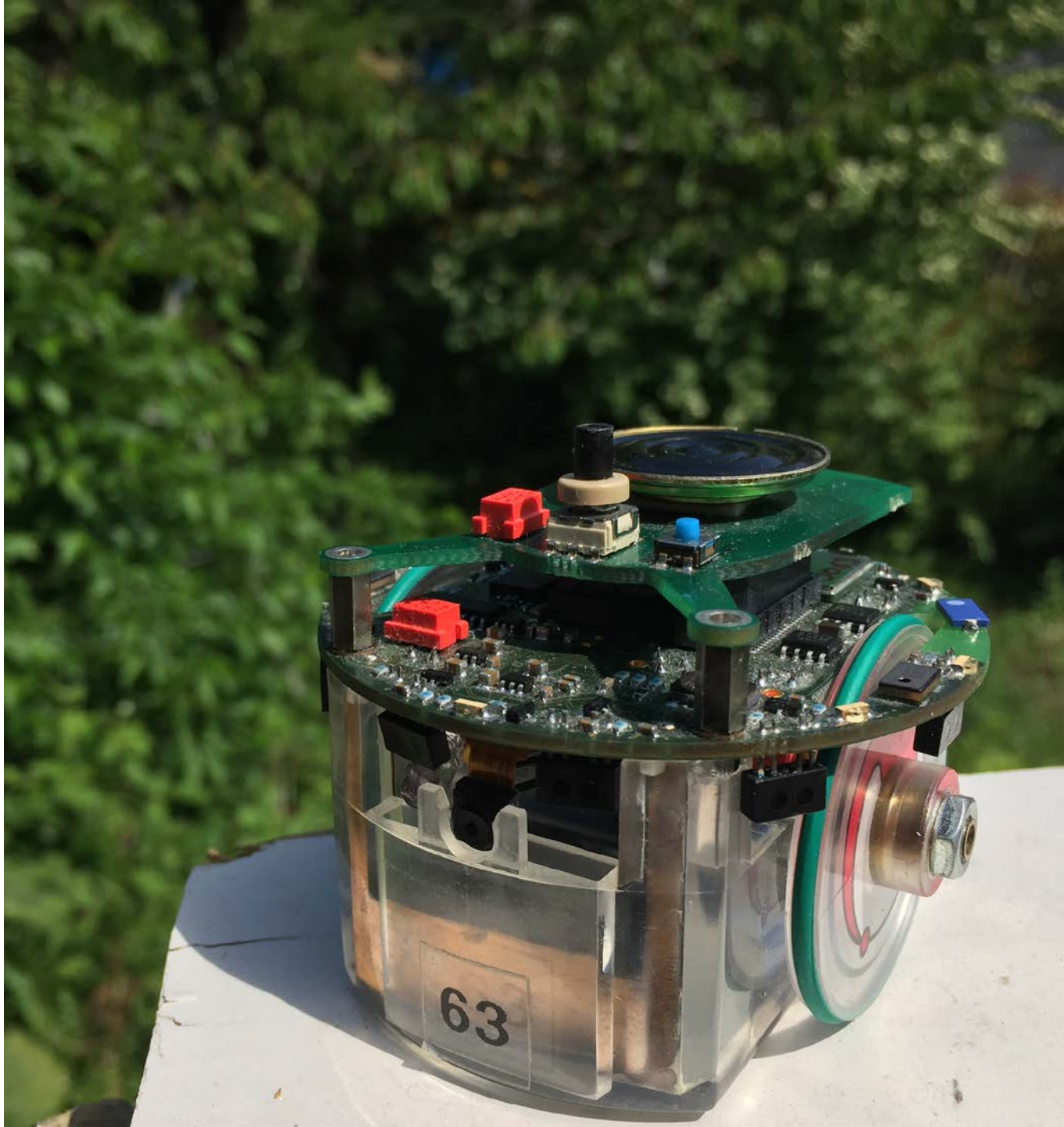


Sign 1 : Turn left (90°)	Sign 2 : Turn right (90°)	Sign 3 : Turn 180°

E-puck

a robot designed for education in engineering

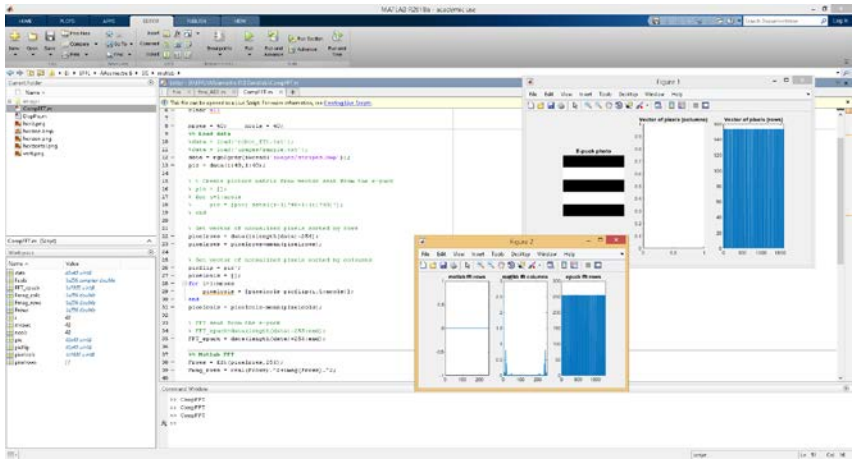
- 8 Infrared sensors
- 1 Accelerometer
- 3 microphones
- 1 front color camera



Methods

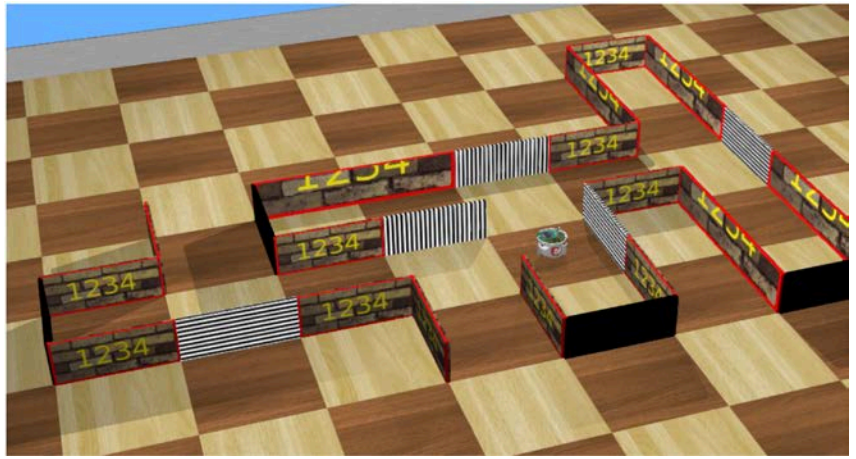
1. Matlab

- Understand FFT theory
- Design recognition strategy



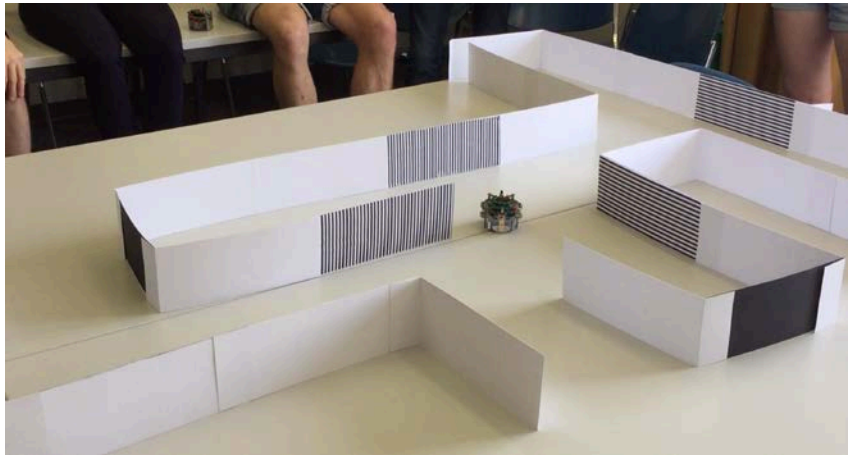
2. Webots

- Implement an algorithm
- Simulate into noise free world



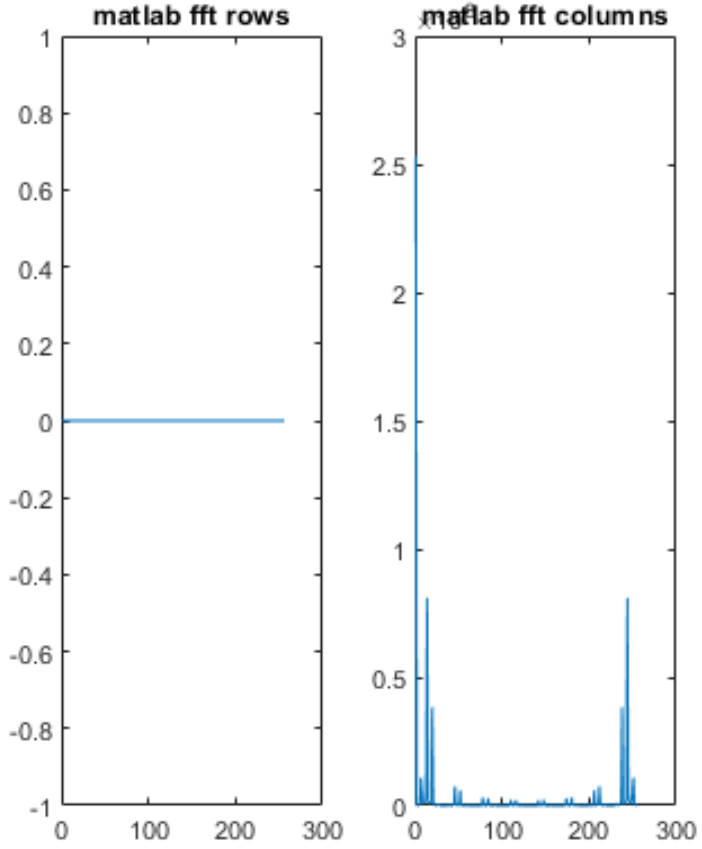
3. Real life

- Implementation IRL
- Face variability and noise
- Adjust strategy



FFT Analysis with Matlab

- The method used to differentiate the 3 signs was the Fast Fourier Transforms (FFT)
- At first FFT analysis with noise free sign's picture on Matlab
- Graph 1 : Results of FFT applied on rows and columns of sign 1 (noise free) :



Graph 1

FFT Analysis with Matlab

Sign/FFT's Amplitude	Rows	Columns
1 : Horizontal lines	0	Several big peaks
2 : Vertical lines	Several big peaks	0
3 : Black	0	0

Table 1 : Results of FFT analysis

Recognition's strategy: Ratio between amplitude on rows and columns

Compare either the **average** or the **maximum** value

Webots Implementation (1.1)

- First strategy

Only one vector can be sent for the FFT.

One mean vector for the rows and one for the columns -> two FFT

Ratio between the two mean amplitude of the FFT vectors

FFT : moy_ratio = moy_col/moy_lin	Direction
moy_ratio < 1.5 && > 0.65	Turn 180°
moy_ratio < 0.5	Turn right
moy_ratio > 2	Turn left

E-Puck Implementation (1.1)

Efficient decisions must be made at all steps due to low processing power conditions

- Wall avoidance
- Image Capture
- Image Processing
- Direction Decision

E-Puck Implementation (1.2)

Image Capture :

- 160x160 image downsized to 40x40 pixels
- 1 image taken each time a sign is detected
- Average all columns and lines into two vectors
- FFT_BLOCK_LENGTH = 64
- Need to fill in the line and column vectors
- Add first values of each vector to the end

```
Set moy_col, moy_lin and max_ratio to 0
For (j=0; j<40; j++) {
        Set count_lin and count_col to 0
        For (i=0; i<40; i++) {
                count_lin[j] += pic[i+40*j]
                count_col[j] += pic[j+40*i]
        }
        moy_col += count_col[j]/40
        moy_lin += count_lin[j]/40
}
j=0
for (i=0; i<FFT_BLOCK_LENGTH; i--) {
        if j >= 40   j -= 40
        vec_lin[i] = count[j]/40
        vec_col[i] = count[j]/40
        j++
}
```

E-Puck Implementation (1.3)

Image Processing:

- Do FFT of column vector
- Calculate Magnitude of FFT
- Calculate Average of Magnitude
- Repeat for line vector
- Choose direction according to ratio and magnitude of vectors

```
e_subtract_mean(vec_row, FFT_BLOCK_LENGTH, LOG2_BLOCK_LENGTH);
e_fast_copy(vec_row, (int*)sigCmpx, FFT_BLOCK_LENGTH);
e_doFFT_asm(sigCmpx);

for (i=0; i<FFT_BLOCK_LENGTH; i++) {
        mag_sqr_row[i] = sigCmpx[i].real*sigCmpx[i].real + sigCmpx[i].imag*sigCmpx[i].imag;
        avg_fft_lin = avg_fft_lin+mag_sqr_row[i];
}
avg_fft_lin = avg_fft_lin/FFT_BLOCK_LENGTH
```

Results (1)

- Vast difference between simulated and real-world conditions
  - Good results on webots
  - Unreliable e-puck results
- Need to change our strategy --> Back to webots

Webots Implementation (2.1)

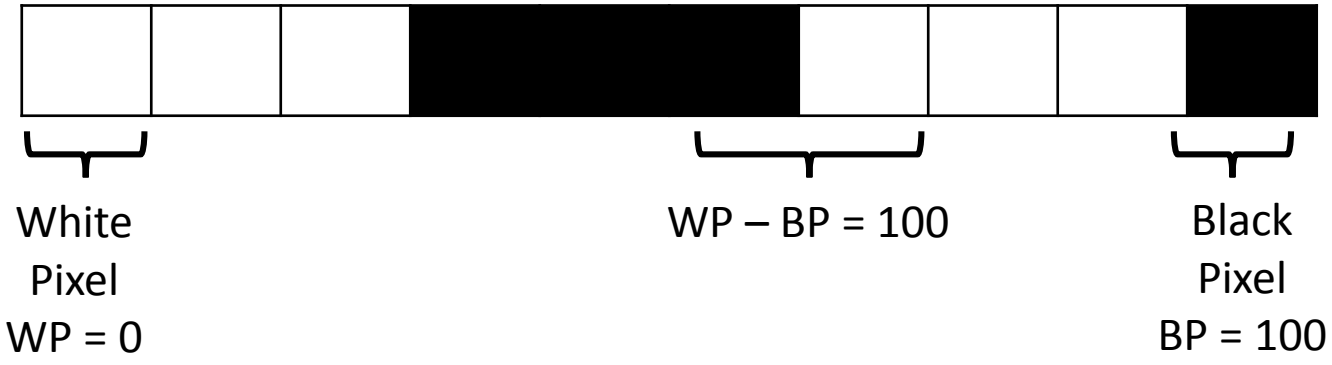
- Second strategy

Same strategy as the first to sent vectors for the FFT

Ratio between the two **max** amplitude of the FFT vectors

+ Simplest High pass filter (before sending to FFT)

```
//High pass filtering
for(i=FFT_BLOCK_LENGTH-1;i>1;i--)
{
  vec_row[i]=(vec_row[i]-vec_row[i-1]);
  vec_col[i]=(vec_col[i]-vec_col[i-1]);
}
//Remove the first value which's a problem
vec_row[0]=vec_row[1];
vec_col[0]=vec_col[1];
```

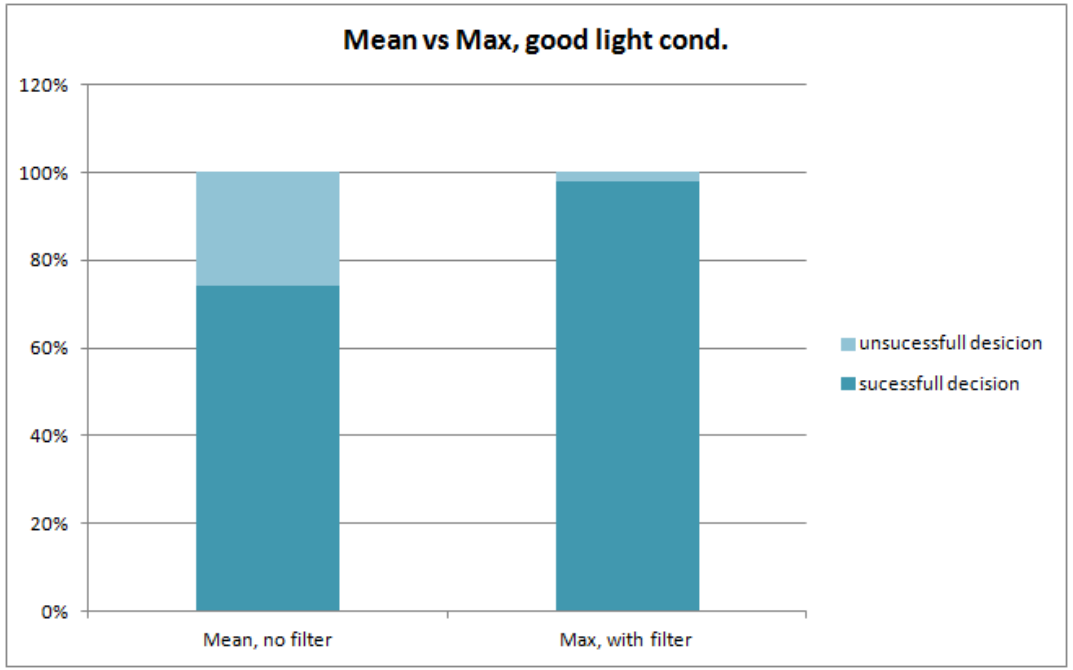


E-Puck Implementation (2.1)

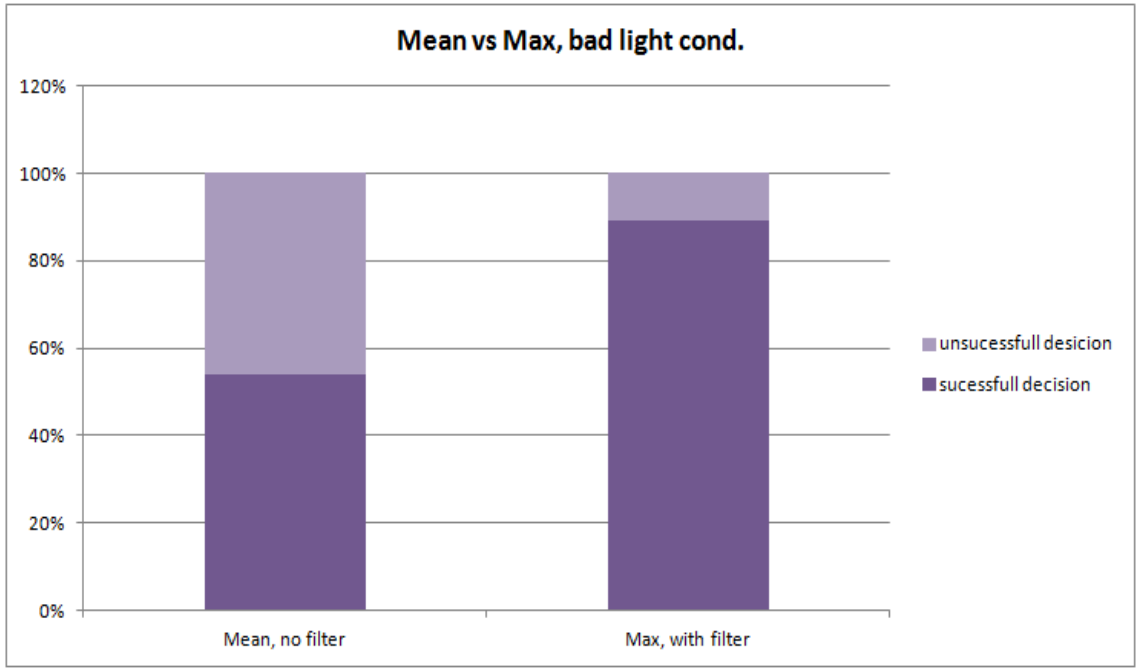
- Promising Results on E-Puck
- Few occasional crashes
- Attempt to use smaller picture (32x32) to reduce memory usage
- 40x40 still yields better results

Mean vs Max

Comparison between mean ratio without filter and max ratio with filter



Graph 2



Graph 3

Final results

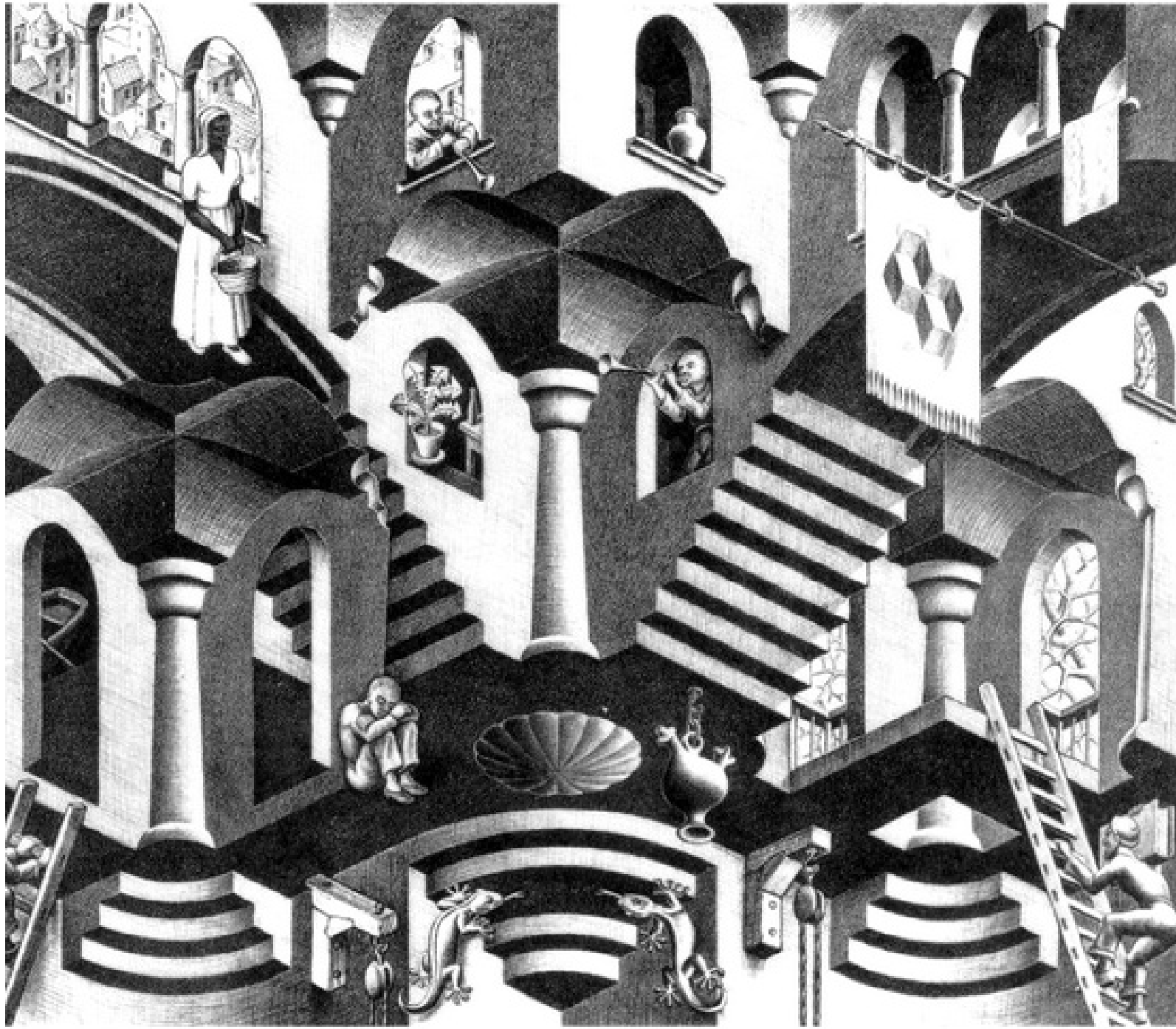
- 3th of June : Maze test

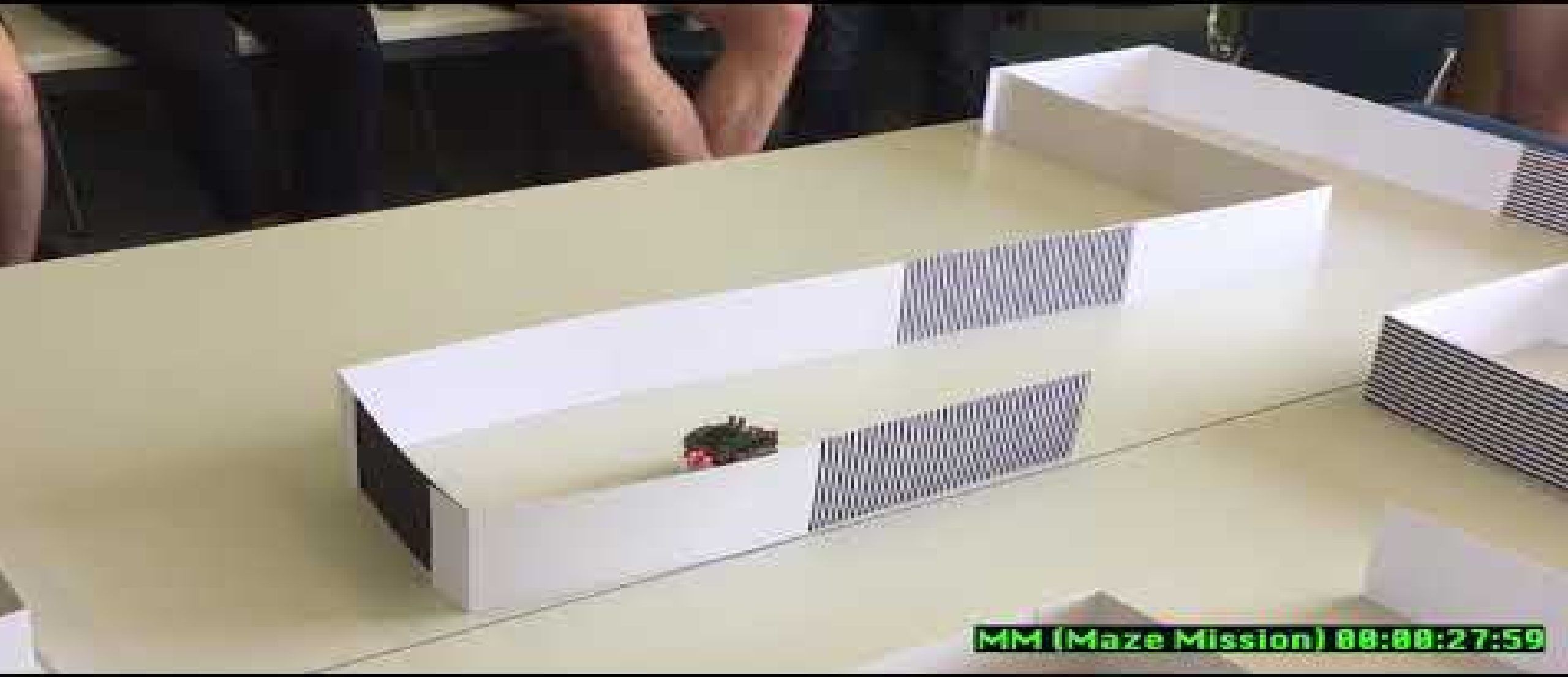
Try#	1	2	3	4
Time	2min21,8s	0min46,4s	0min45,2s	0min39,6s

- Average : **1min8,25s** or **43,73**s without first try

- Best attempt : **39,6s**

The maze




MM (Maze Mission) 00:00:27:59

Conclusion

- Globally : good final performances,
- The project's goal is achieved
- Improvement can be done :
  - Recognition of the sign 1 (horizontal lines)
  - Recognition in the shade (bad light condition)
  - Optimization of memory allocation (to avoid crashes)

Thank you for your attention !