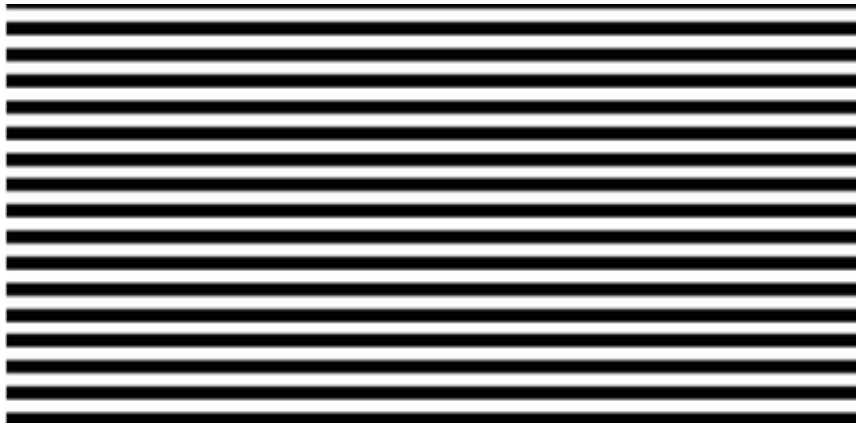
SIS project

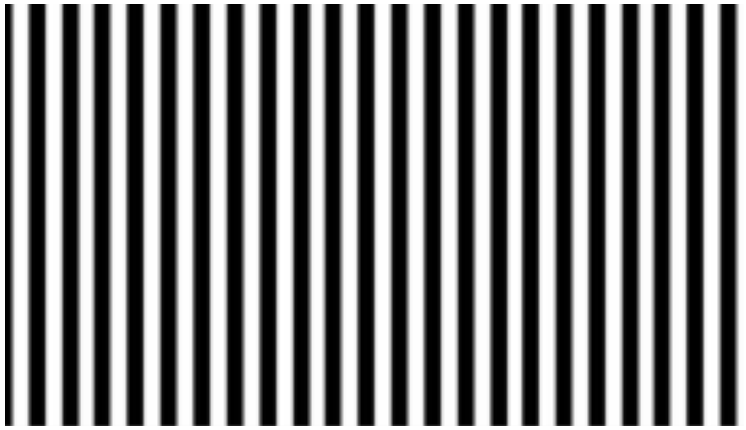
Esther Barberis and Manuel Walser
28.05.2020

Project goal

Implement a c program for the E-puck to exit the maze by reading road signs



Horizontal/Left turn



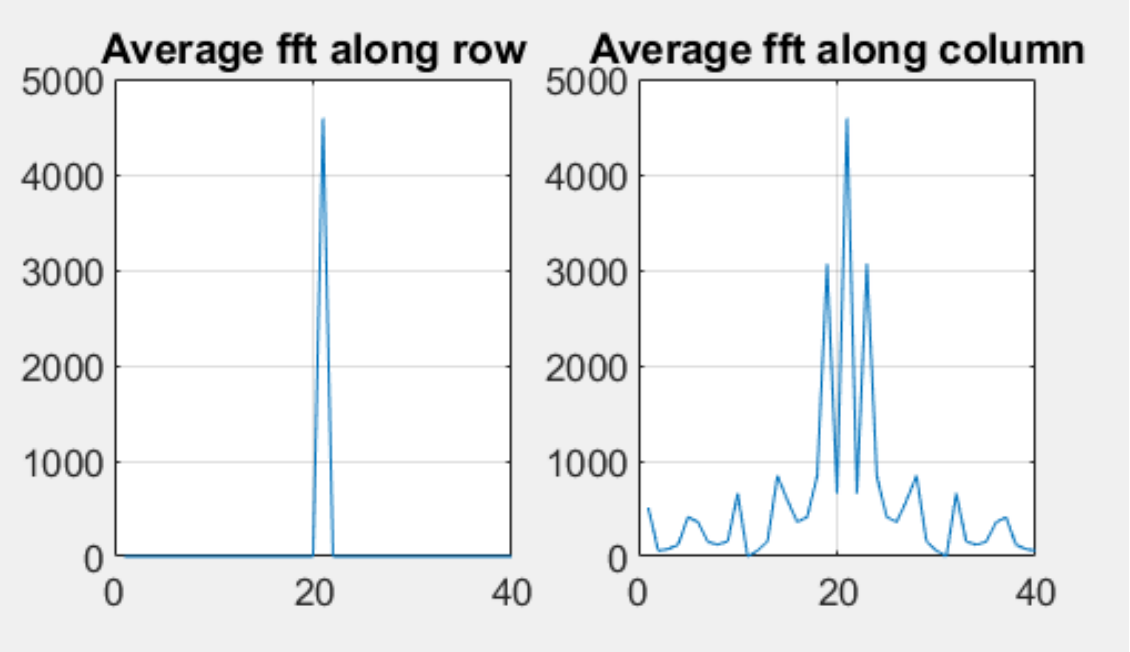
Vertical/Right turn

Black sign/ turn around

1. Matlab

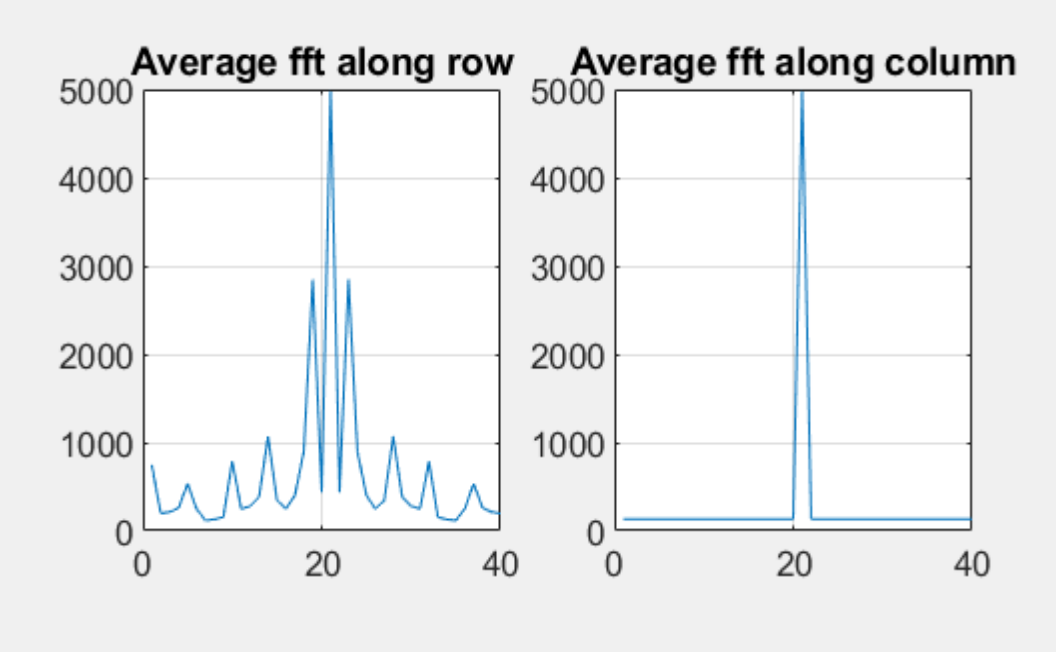
horizontal stripes

- high variance of the amplitude for the columns
- only one peak for the rows.
- Ratio col/rows = 2.222222e+00

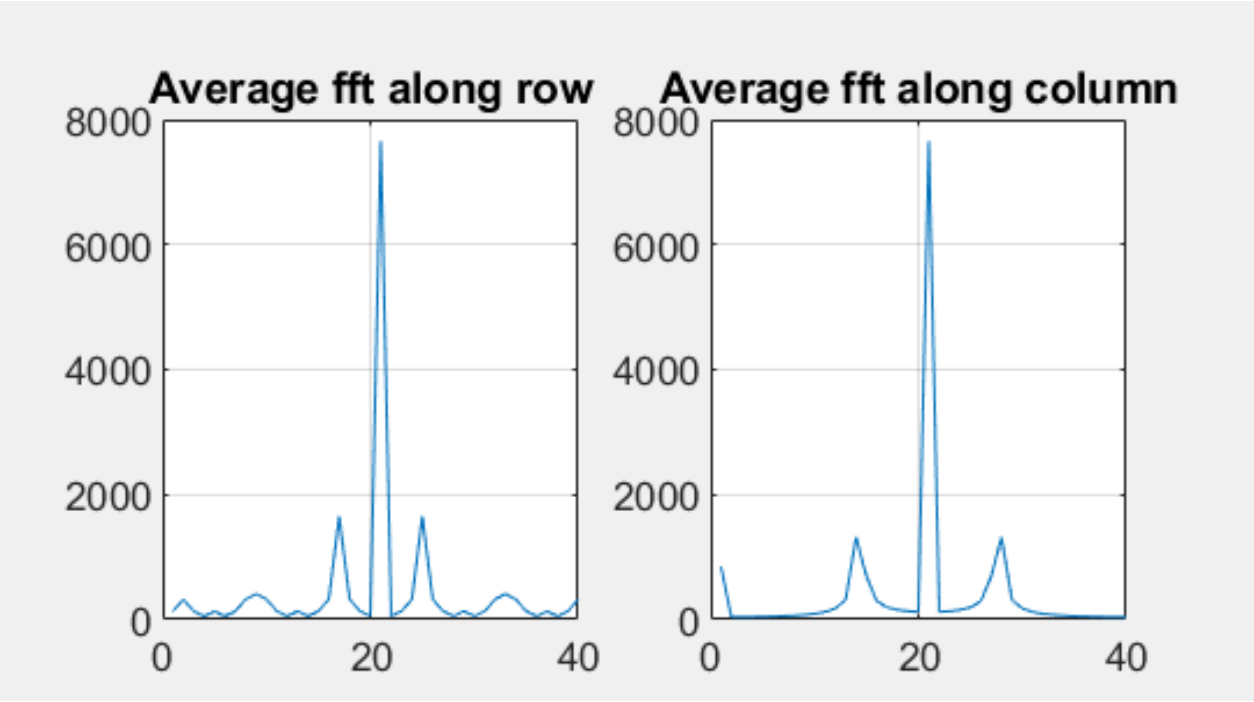


vertical stripes

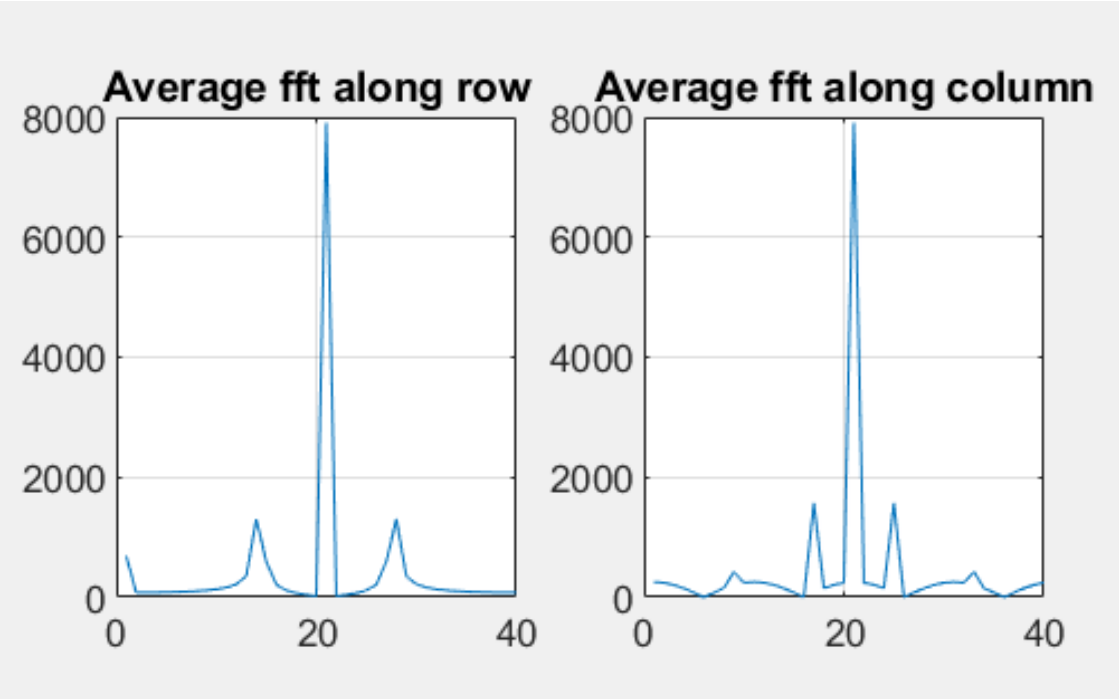
- high variance of the amplitude for the rows.
- only one peak for the columns.
- Ratio col/rows = 0.5154005



In the case of the noisy image, we notice on the graphs more ambiguity between the amplitudes for the rows and he columns.



Vertical : Ratio columns/sums = 0.98

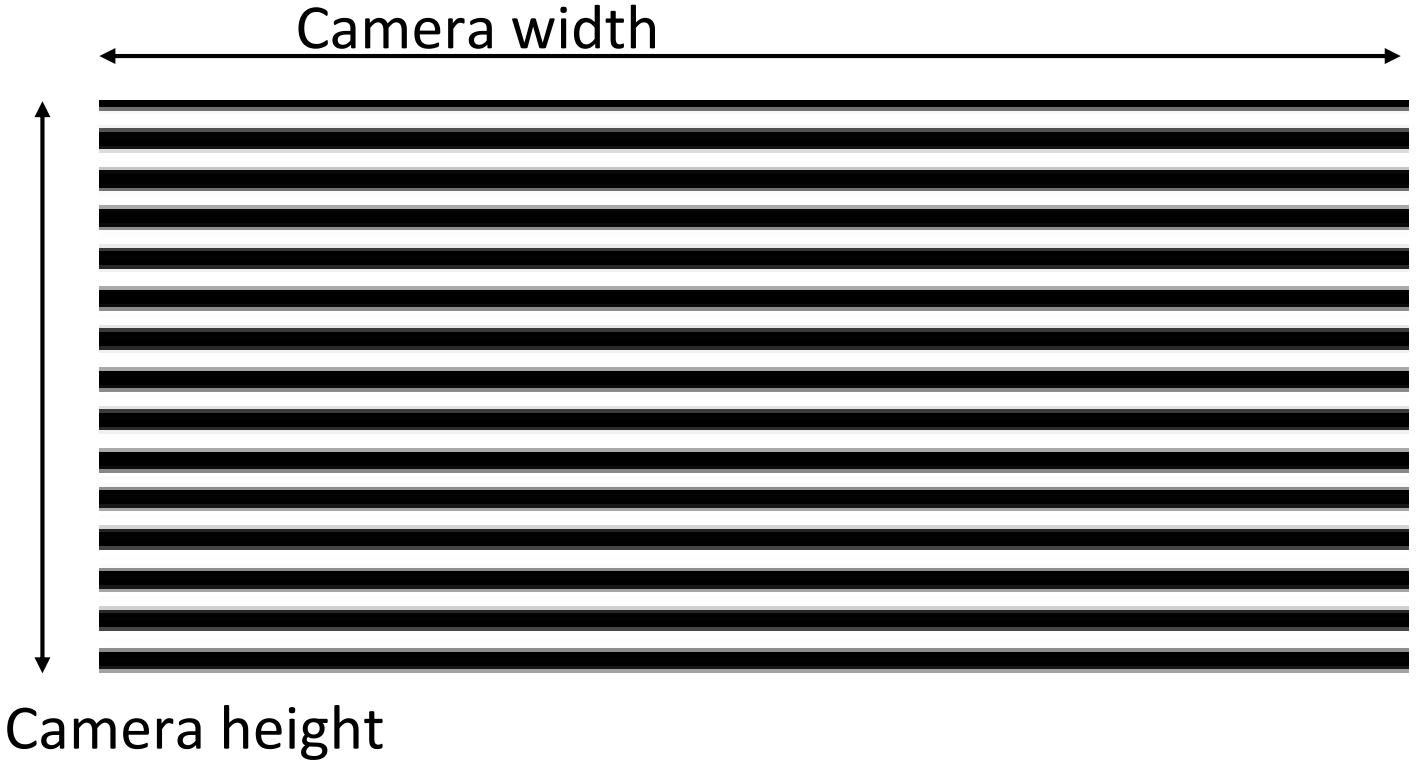


Horizontal : Ratio columns/sums = 1.017

Amplitude row

- For every row of the image we applied:
- $mean\ of\ rows = \sum_{m=0}^{camera\ width} \frac{(fft\ of\ image(m))}{(camera\ width)}$
- Finally, we got the Amplitude of the mean of rows and calculating the over all mean
- $Amplitude\ row = \frac{real(mean\ of\ rows)^2 + imaginary(mean\ of\ rows)^2}{camera\ height}$

Amplitude row



→ Get mean of fft row 1
→ Get mean of fft row2
→ …

1. Get mean of each row (see image)
2. Get amplitude for each mean calculated before
3. Get mean amplitudes calculated in 2
4. If Amplitude of row < Amplitude of columns → Horizontal, turn left

Optimizations

- Use of a filter, to get mean over neighbour cells to smooth out noise

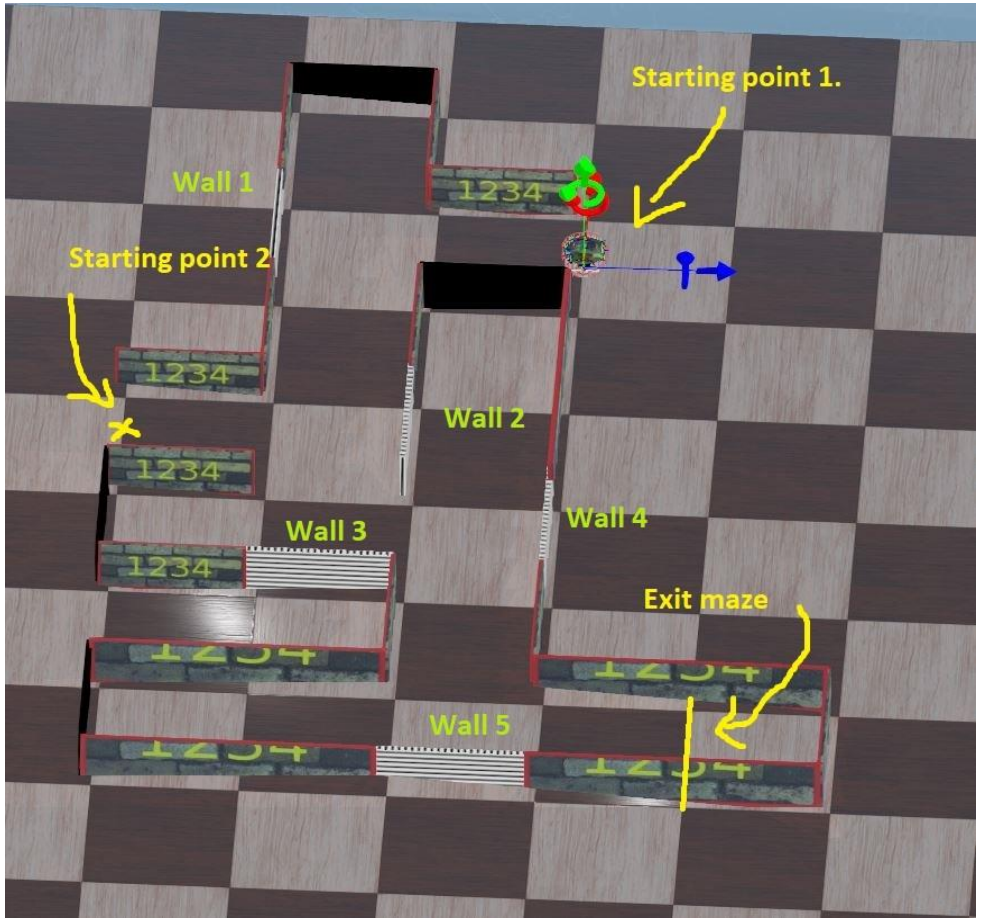
```
for (j=0; j<camera_height; j=j+1) {
    for (i=0; i<camera_width; i=i+1) {
        image[j][i] = signal_data[j][i];}}

for (j=2; j<camera_height-2; j=j+1) {
    for (i=2; i<camera_width-2; i=i+1){
        som = 0;
        for (y=-2; y<=2; y=y+1) {
            for (x=-2; x<=2; x=x+1) {
                som = som + image[j+y][i+x];
        signal_data[j][i] = (int) som/25.0;
    }}}}
```

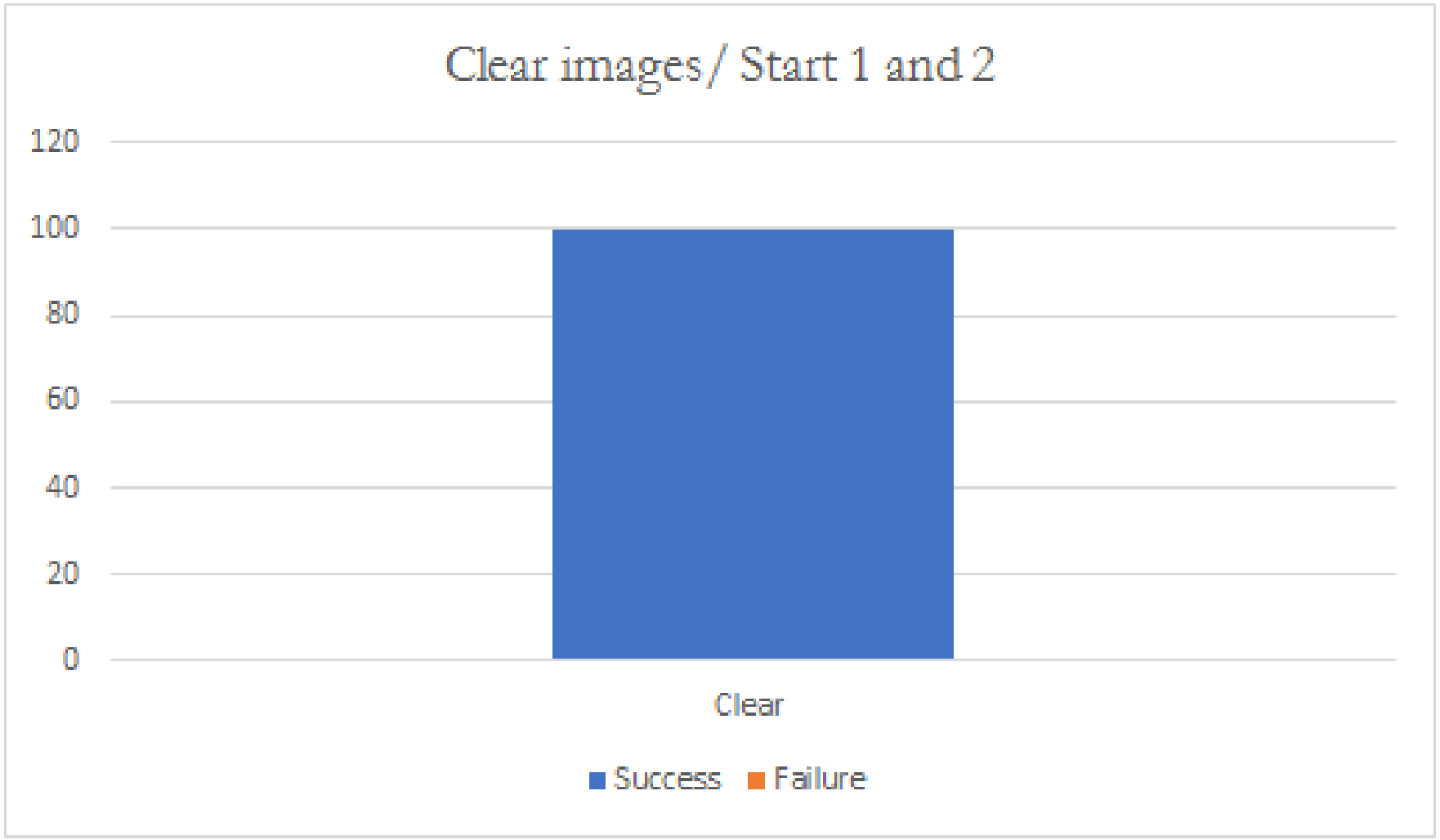
Criteria for the ratio rows/column:

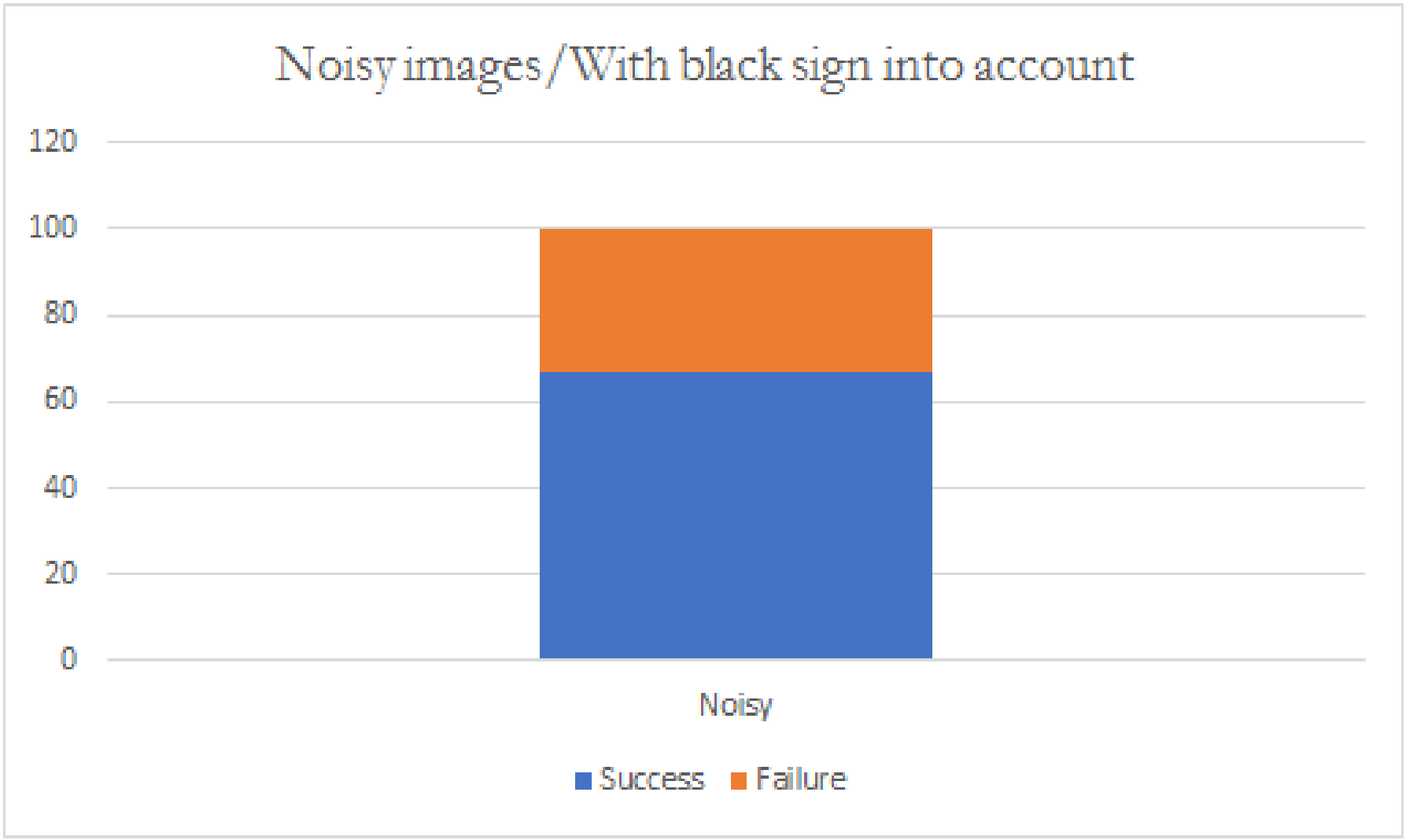
- if mag col/mag rows < 0.6 → vertical, turn right
- if mag col/mag rows > 0.6 → horizontal, turn left
- if (mag col + mag rows) /2 < BlackThreshold, turn around

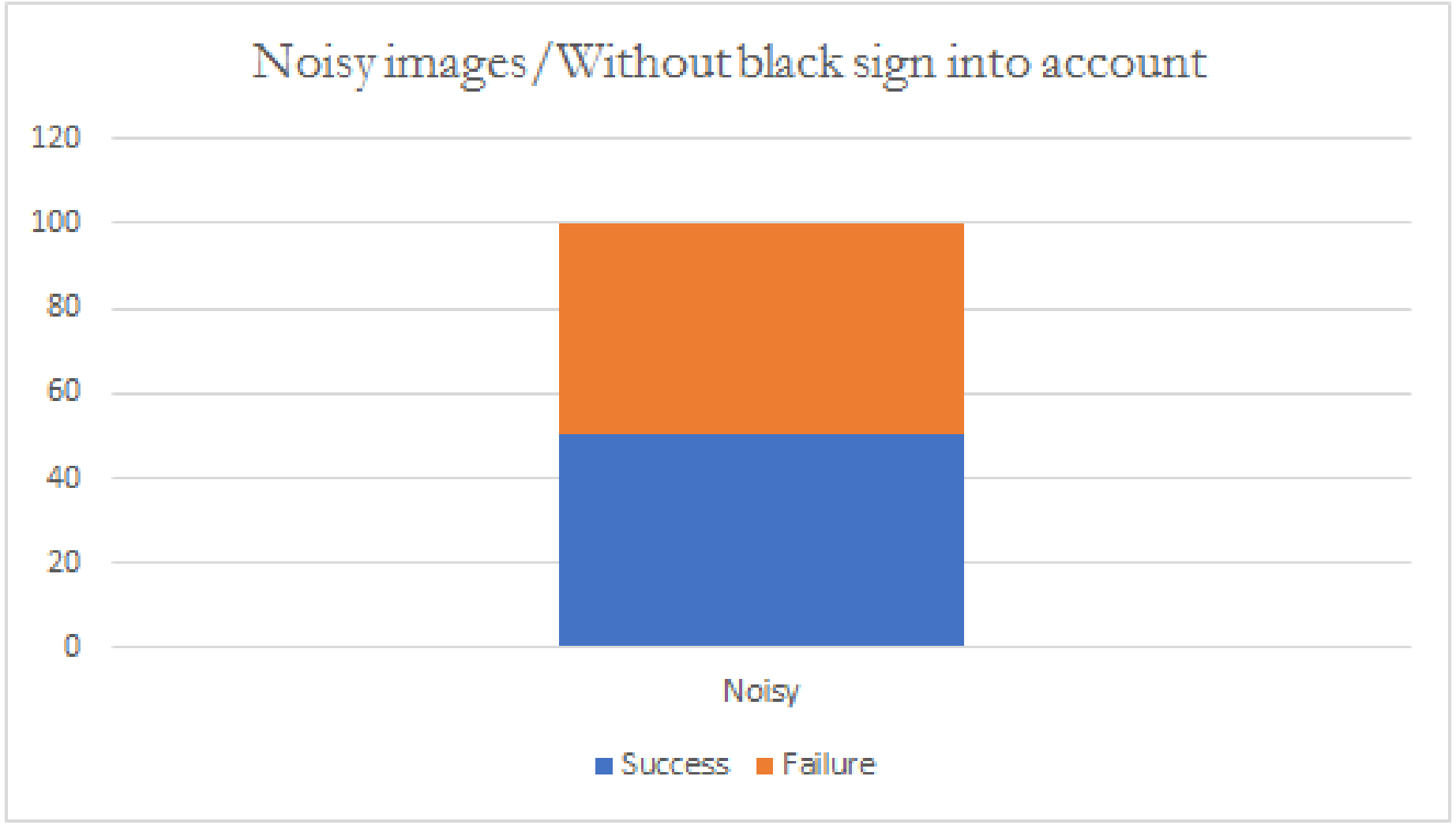
Different Scenarios tested

- Start 1 and 2
- Clear and noisy images
- Camera noise
- fft processing with filter




Clear images / Start 1 and 2
120
100
80
60
40
20
0
Clear
Success
Failure


Noisy images /With black sign into account
120
100
80
60
40
20
0
Noisy
Success
Failure


Noisy images/Without black sign into account
120
100
80
60
40
20
0
Noisy
Success
Failure

Time [s] for start 1

Failures	0	1	2	3
Clear images /No filter	54.9			
Clear images/With filter	54.54			
Noisy/No filter		63.4	93.55	126.3
Noisy/ filter		63.26	90	124.25

Odomerty

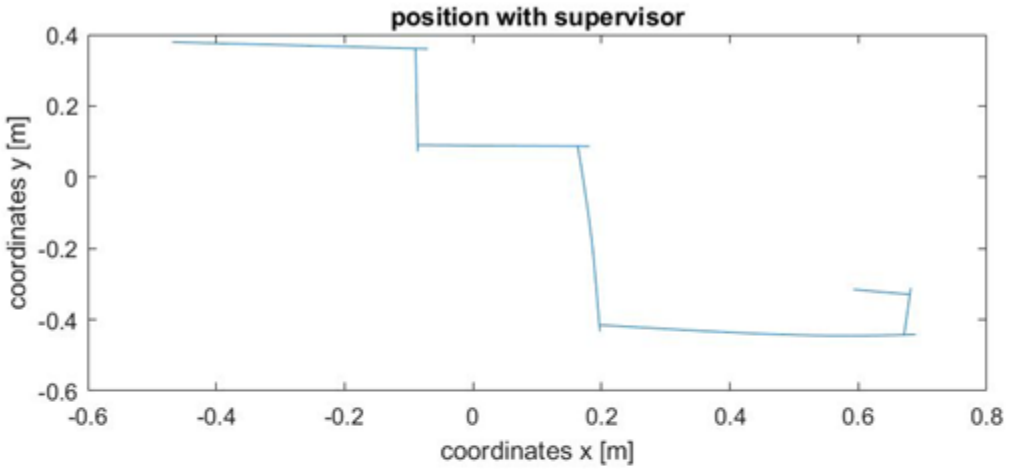
Implementation in webots:

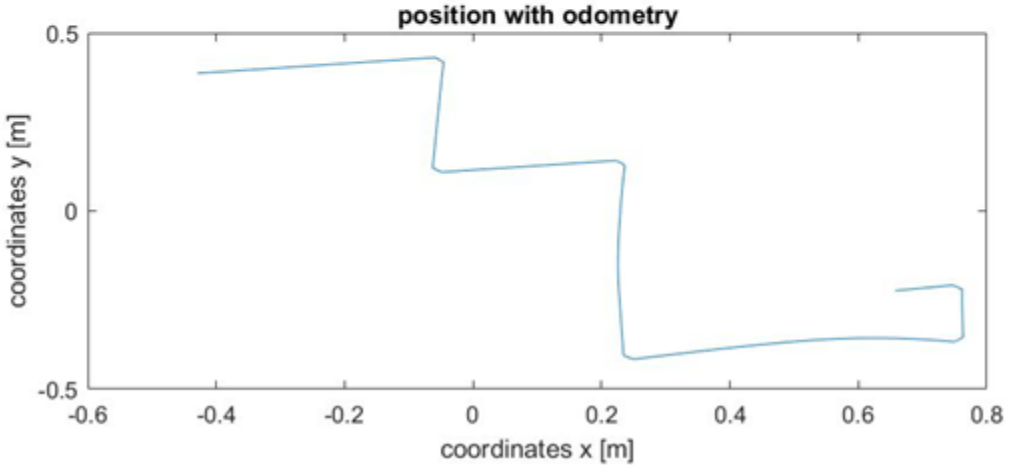
- $\varphi = 2 * \pi * \left(\frac{N_{turns\ per\ wheel}}{N_{total\ of\ turns}}\right)$ = *wb_position_sensor_get_value()*
- Distance covered by the left wheel $dl = \frac{\varphi(left)}{\Delta t} * \text{wheel radius}$;
- Distance covered by the right wheel $dr = \frac{\varphi(right)}{\Delta t} * \text{wheel radius}$;
- The mean distance of both tyres $dis = \frac{dl+dr}{2}$;
- delta orientation = θ'= $da = \frac{dl-dr}{(axe\ lenght)}$

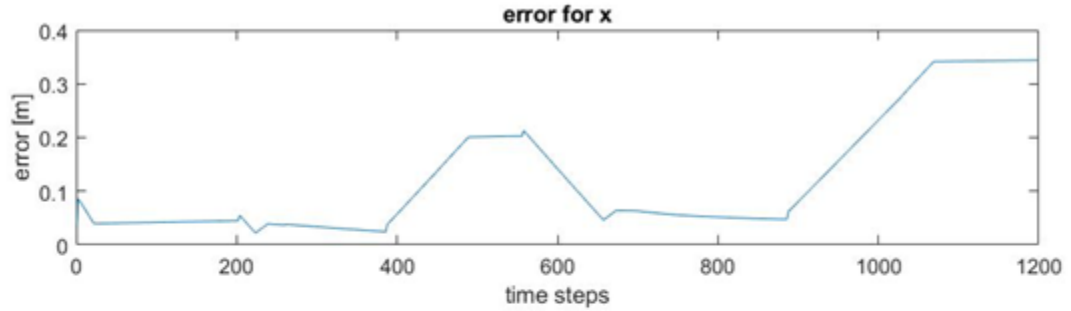
- $x(t) = x(t-1) + \int(dis * \cos(\theta)) * dt$
- $y(t) = y(t-1) + \int(dis * \sin(\theta)) * dt$

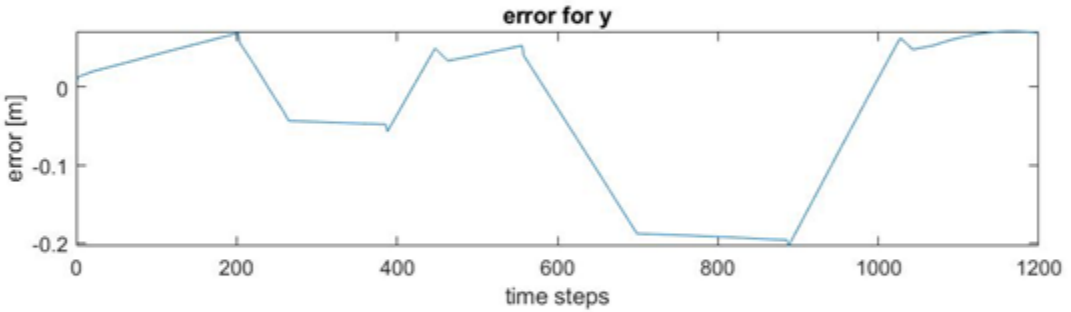
$$\dot{\xi}_I = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r\dot{\varphi}_1}{2} + \frac{r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_1}{2l} + \frac{-r\dot{\varphi}_2}{2l} \end{bmatrix}$$

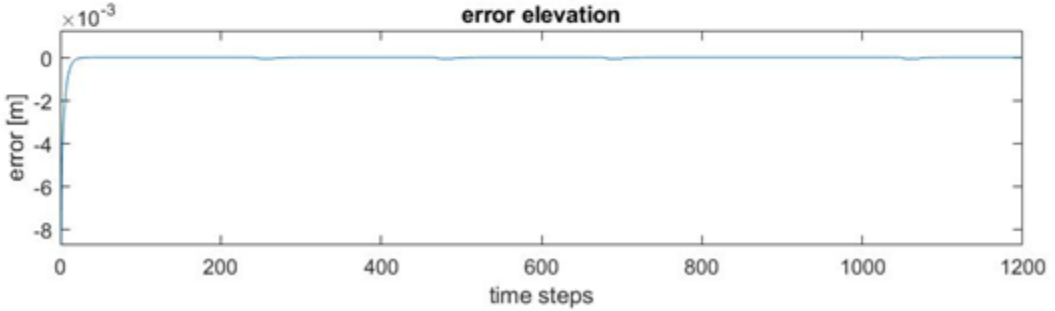
Supervisor/Odometry results – start 2











What went wrong/What could have been done better?

- Implement a high pass filter on the 2D array
- Using the fft with 1D array would have been perhaps simpler in terms of filtering
- Take the ratio of the maximum amplitude instead of the mean of magnitudes for the conditions
- Problem with the first sign in webots in start 1, with z variable.