LINE FOLLOWING USING A CAMERA SIGNALS, INSTRUMENTS & SYSTEM
INTRODUCTION

- C language
- Webots software
- E-puck robot

- Proximity sensors
- Camera

Figure 1: Real e-puck
OBJECTIVES

1 - Program an e-puck in Webots
   ▪ Find the line
   ▪ Follow the line
   ▪ Avoid obstacle placed on the line

2 - Make this method work with a real e-puck
DIFFERENT VARIABLES

- 8 variables for proximity sensors
  \[ \text{sensor\_value}[i] \]

- left\_speed and right\_speed

- matrix[ j ][ i ]
  \[ \text{Contains the greyscale values of the pixels taken by the camera} \]
  \[ \text{0 : black pixel} \]

Figure 2: matrix[ j ][ i ]
STATE 0 : DETECTION OF THE LINE

- The robot looks for a black pixel:
  \[ \text{matrix}[j][i] = 0 \]
- Black pixel detected:
  \[ \text{state 1} : \text{going to the line} \]
  \[ \text{state 2} : \text{following the line} \]
  \[ \text{if the black pixel is located at the bottom of the matrix} \]
- No pixel detected:
  \[ \text{move straight forward} \]
  \[ \text{rotate on its axis} \]
**STATE 1 : GOING TO THE LINE**

- Black pixel detected
  - « j » and « i » of the matrix are saved as the position
- These two variables will help to calculate the speed of each wheel when approaching to the line

  - Distant line $j = [1,18]$
    
    \[
    \begin{align*}
    \text{right}_{-}\text{speed} &= \text{initial}_{-}\text{speed} + 26 - i \\
    \text{left}_{-}\text{speed} &= \text{initial}_{-}\text{speed} - 26 + i
    \end{align*}
    \]

  - Intermediary line $j = [19,29]$
    
    \[
    \begin{align*}
    \text{right}_{-}\text{speed} &+= \text{initial}_{-}\text{speed} \times (1 - i/26) \\
    \text{left}_{-}\text{speed} &+= \text{initial}_{-}\text{speed} \times (1 - (52 - i)/26)
    \end{align*}
    \]

  - Closer line $j = [31, 35]$
    
    \[
    \begin{align*}
    \text{right}_{-}\text{speed} &+= \text{initial}_{-}\text{speed} \times (1 - i/40) \\
    \text{left}_{-}\text{speed} &+= \text{initial}_{-}\text{speed} \times (- (52 - i)/40)
    \end{align*}
    \]

  - On the line $j > 36$

    \[
    \text{STATE 2 : Line following}
    \]
STATE 2 : FOLLOWING THE LINE

- The speed of the right wheel is a function of the position of the left edge of the line.
  \[ \text{right\_speed} += \left(\frac{\text{initial\_speed}}{40}\right) \times (26-i) \]

- The speed of the left wheel is a function of the position of the right edge of the line.
  \[ \text{left\_speed} += \left(\frac{\text{initial\_speed}}{40}\right) \times (i-26) \]

- If the line is only 3 pixels thick:
  \[ \text{there is a T-shape in front of the epuck} \]
  \[ \text{STATE 2.2: make a u-turn} \]
STATE 2.2 : MAKE A U-TURN

To do that, the speed of each wheel is implemented this way:

→ left_speed = - initial_speed;  → right_speed = initial_speed;

The robot turns around and the next state is initialized as the state 2 (following the line). The time step is also slowed down to let the robot make the 180° rotation.
EXPERIMENTS WEBOTS

STATE 3 : AVOIDING OBSTACLE

- If one of the 8 infrared sensors measures a higher value than 700 :
  \[ sensor\_value[j] > 700 \quad i = [0,7] \]
  \[ \rightarrow \text{STATE 3} \]

- The braitenberg equation :
  \[ \rightarrow \text{left\_speed} += 2*(\text{braitenberg\_coefficients}[j][0] \cdot (1 - (\text{sensor\_value}[j] \div \text{RANGE}))) \]
  \[ \rightarrow \text{right\_speed} += 2*(\text{braitenberg\_coefficients}[j][1] \cdot (1 - (\text{sensor\_value}[j] \div \text{RANGE}))) \]

- If \( \text{sensor\_value}[j] < 60 \)
  \[ \rightarrow \text{STATE 0 : detection of the line} \]
EXPERIMENTS
REAL E-PUCK

- Some differences with the epuck in Webots:
  - Configuration of the camera: 40x40
  - the matrix that contains the color value of each pixel in Webots is in one dimension for the real e-puck: it's a vector which length is 1600:
  - New specific function for the real epuck
RESULTS

- Webots simulation works pretty well but has some limitations:
  - The e-puck doesn’t get around the obstacle.
    - He just avoid the obstacle and then try to find the line again.
  - Sometimes the e-puck goes to state 2.2 instead of state 3
    - when finding an obstacle on its way: just turned around itself

- Our real e-puck can take pictures and avoid obstacles, but it can’t find the line and follow it.
CONCLUSION

- Complex but interesting project: professional project

- Incomplete code for real e-puck we spent a lot of hours trying different ways, sometimes without results

- Need more time to optimized our code