Road Sign Recognition with e-puck

- INTRODUCTION
- EXPERIMENTS
  - DISPLACEMENTS
  - IMAGE PROCESSING AND DECISION
- RESULTS
- CONCLUSION

Supervisors: Haghigat Bahar, Duarte da Cruz Baptista Dias
Implementation of a code providing the e-puck the capability to read a simple “road sign” with its camera, analyze the picture on board and decide whether: turn left, turn right, turn 180°.
Code’s structure - Infinite loop:

1. Move straight forward
2. Detect the wall with IR sensors
3. Straighten if not perpendicular to the wall
4. Recoil for a user-defined distance
5. Stop
6. Take a picture
7. Process the image
8. Turn left/right/180° + LEDs on
9. Back to point 1
Go straight forward:
Wheels' speed set at a defined maximum speed

Stop in front of a wall:
Wheels' speed set to zero when the front sensors reach a defined threshold value

METHOD 1 – Time dependent

Move back:
Inversion of wheels' speed for a defined time, then speed is set to zero

Obstacle avoidance:
Using Braitenberg coefficients

Straightening when in front of a wall:
Slowly turn until the difference between sensors 1 and 6 values is in the interval [25-30]

METHOD 2 – Motor steps dependent

Move back:
Inversion of wheels’ speed until a defined number of motor steps is reached

Turn left/right/180°:
Wheels’ speed set opposite for a defined amount of time, then speed is set to zero

→ Not implemented because of problems in the library. Different motor steps definition into motors.c and motors_timer3.c file (int vs long int)

Turn left/right/180°:
Wheels’ speed set opposite until the sum of the angular displacements exceeds a defined angle (90°, -90°, -180°).
Piecewise linear transformation with automatic dynamic range:

\[
f_j = \begin{cases} 
  
  f_j / v_{\min} & \text{for } f_j < v_{\min} \\
  f_j - \frac{v_{\min}}{v_{\max}} \cdot 253 + 1 & \text{for } f_j \in [v_{\min}, v_{\max}] \\
  f_j - v_{\max} + 254 & \text{for } f_j > v_{\max} 
\end{cases}
\]

where:

\[
  v_{\min} = \mu + \sigma \cdot \alpha - \sigma \cdot \sqrt{k - 3 \cdot \alpha^2} \\
  v_{\max} = \mu + \sigma \cdot \alpha + \sigma \cdot \sqrt{k - 3 \cdot \alpha^2}
\]

Expands the range of values of the picture
Highlights the main peak of the sinusoid
METHOD 1: DFT – Main peak of the sinusoid

“Take the max value of the coefficients between frequency 1 and 33 for both types of FFT computed (sum of rows and sum of columns). Then, if the max value in the selected range of the FFT applied to the sum of columns (rows) is higher than that one of the sum of the rows (columns), the stripes are horizontal (vertical)”
“If the sum of all the coefficients of the FFT made on the sum of rows (columns) is lower than the sum of all the coefficients of the FFT applied to the sum of columns (rows), the stripes are horizontal (vertical)”
- LEDs really useful to debug the code
- Length of the vector required to execute the FFT on e-puck: 64, 128, 256, 512
  → Since the picture is 40x40, the 24 initial value of the sum of columns and sum of rows are added at the end of each vector in order to have a 64 values.
  → No influence on the decision-process observed
- The distance from which the picture is taken has a strong influence on the result in not optimal light conditions.

**Statistics (not based on the real maze) - 30 attempts per sign**

- **Good light condition:**
  - ✓ 100% success with all road signs
- **Poor light conditions:**
  - ✓ 95% success with simple road sign (shadow)
  - ✓ 70% success with difficult road sign (confused with all-black sign)
    → Decrease in rate of success due to threshold imposed to recognize all-black sign
  - ✓ 100% success with all-black sign
Problem in the library significantly slowed down the schedule

E-puck unable to connect to the computer the last day
→ Change from 134 to 66 and recalibration

Better solutions exists to increase efficiency

Great potential of C language to treat and analyze signals and images

Considering all problems and the background from previous courses we are satisfied of our results