Signals, Instruments and Systems

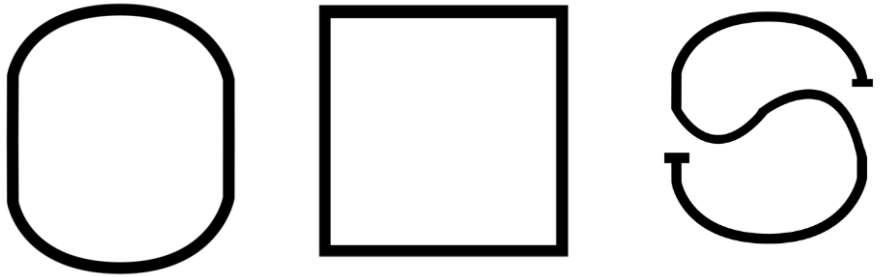
Project presentation:
Line Following using e-puck's camera

Group 8: Camille Dross, Jeanne Estienne

26th May 2020

Goals

- Find line
- Perform line following
- Avoid obstacles
- Try different pathways
- Implement odometry

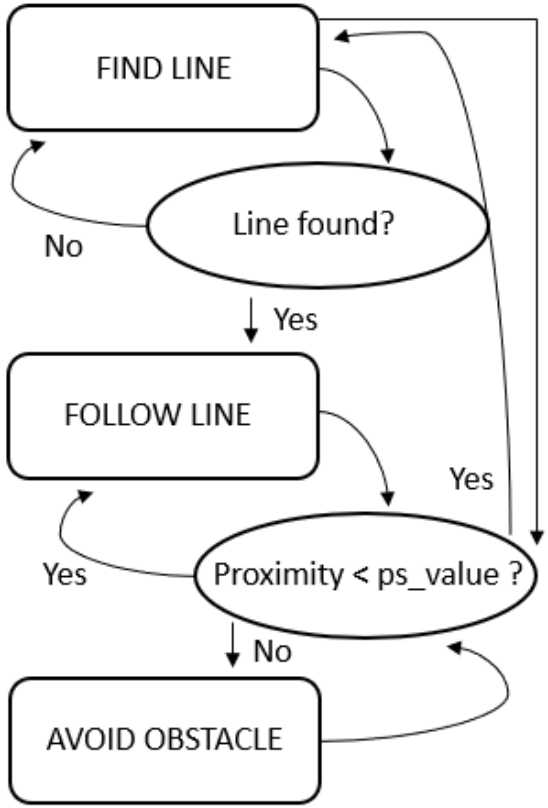


Implemented shapes

Strategy: state machine

Sensors used:

- camera sensor
- proximity sensors



Strategy: odometry

- **Method for the robot to estimate its position using its devices**
- **Proprioceptive sensor used : Motor sensors** in Webots controller to take wheels rotation angles
- Implementation of a **supervisor, an exteroceptive sensor**, to compare positions

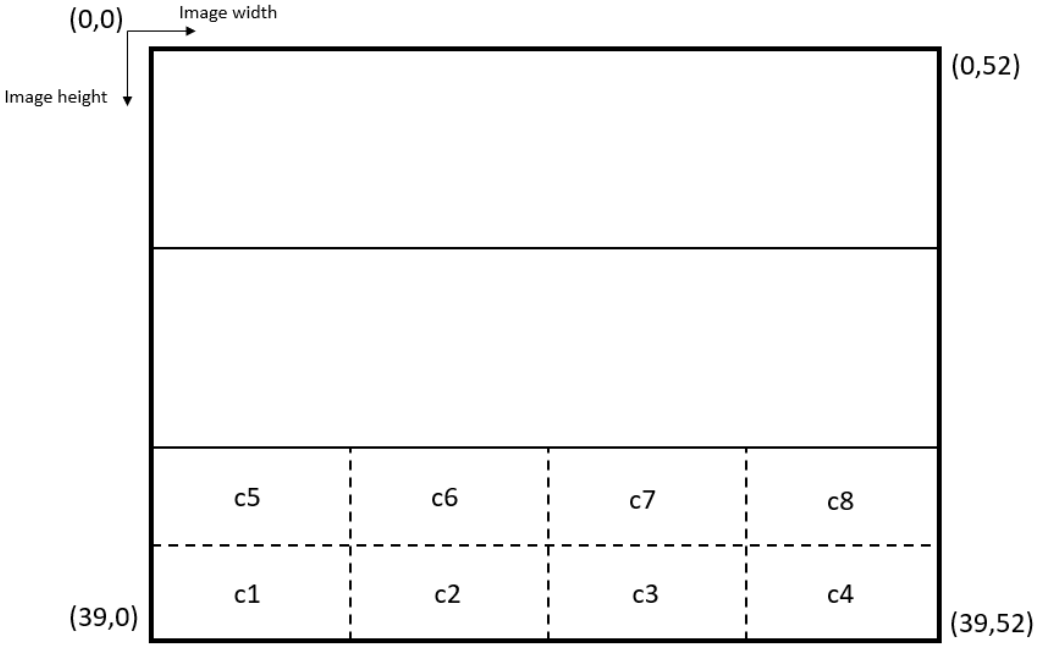
Implementation: camera & image analysis

- **Camera** configuration
- Conversion to **grayscale** pictures

→ wb_camera_image_get_gray

Image slice allow:

- count black pixels
- concentrate only on a specific part



Implementation: code

Steps:

1. Reading **sensor values** (proximity and camera)
2. Action = … → switch **case** ‘ ‘
3. Execution of the part of code belonging to the state

Algorithm 1 Code explanation

```
Main function{
  while(){
     call function: analyse_image, obstacle ...
     switch (action){
          case ‘L’: ...
          case ‘F’: ...
          case ‘O’: ...
      } end switch
   } end while
} end main
```

Implementation: find line

```
Looking line :
    if (*c2!=0){ //condition for line found
        return 1;
     }
     else{ //for the other case
        --> charge the wheel speed to go straight
        printf("Looking for the line");
     }
     return 0;
```

```
switch(action){
      case 'L': //Looking for the line
        line_found = looking line;
        if (line_found == 1){
            action = 'F';}   //follow line
      break;
```

Implementation: line following

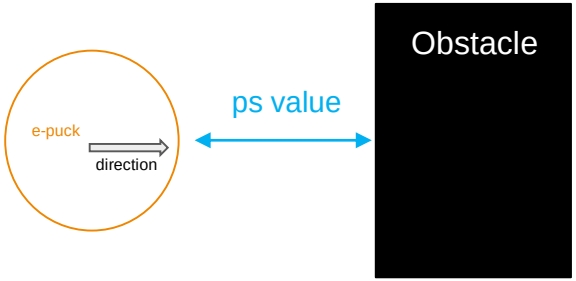
- Counting pixels
- Checking if there is black pixels

```
Follow line:
   if(*c==0){ //arrive at the end of the line
       --> turn back
    }
    else if (*c_left > *c_right){ //follow the line without turning to much
       --> turn left
    }
    else if (*c_left < *c_right){ //follow the line without turning to much
       --> turn right
    }
    ...
    else{ //if it does not enter a condition go straight
        --> go straight
    }
  --> put speed in the wheels
```

```
case 'F': // follow the line once the line was found
 follow_line();
  if(ds_value[] > 350){ //avoid obtacles
      action = 'O';
  }
break;
```

Implementation: avoid obstacles

E-puck maximum **distance** with an objects:



Braitenberg **coefficient**:

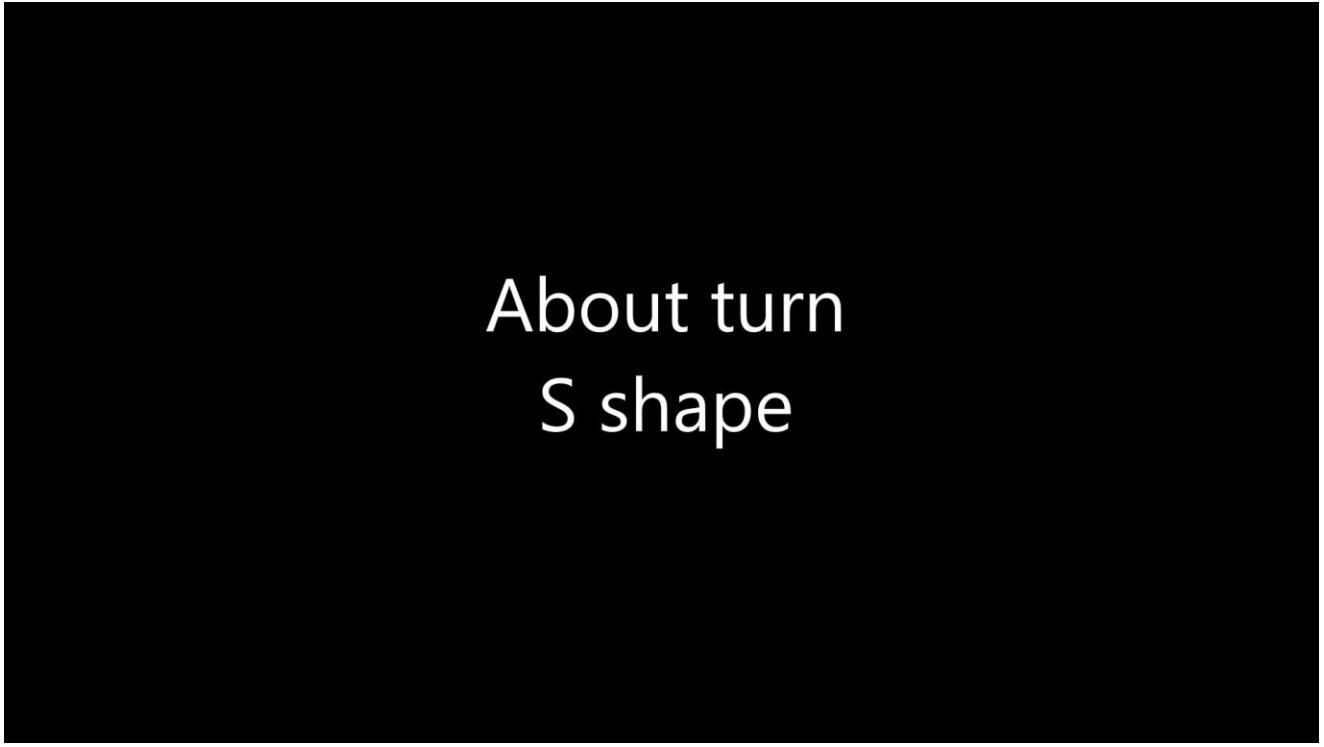
{(140, -35), (110, -15), (80, -10), (-10, -10),
(-15, -10), (-5, 80), (-30, 90), (-20, 160)}

Braitenberg : **simple linear combination** of parameters and sensor values

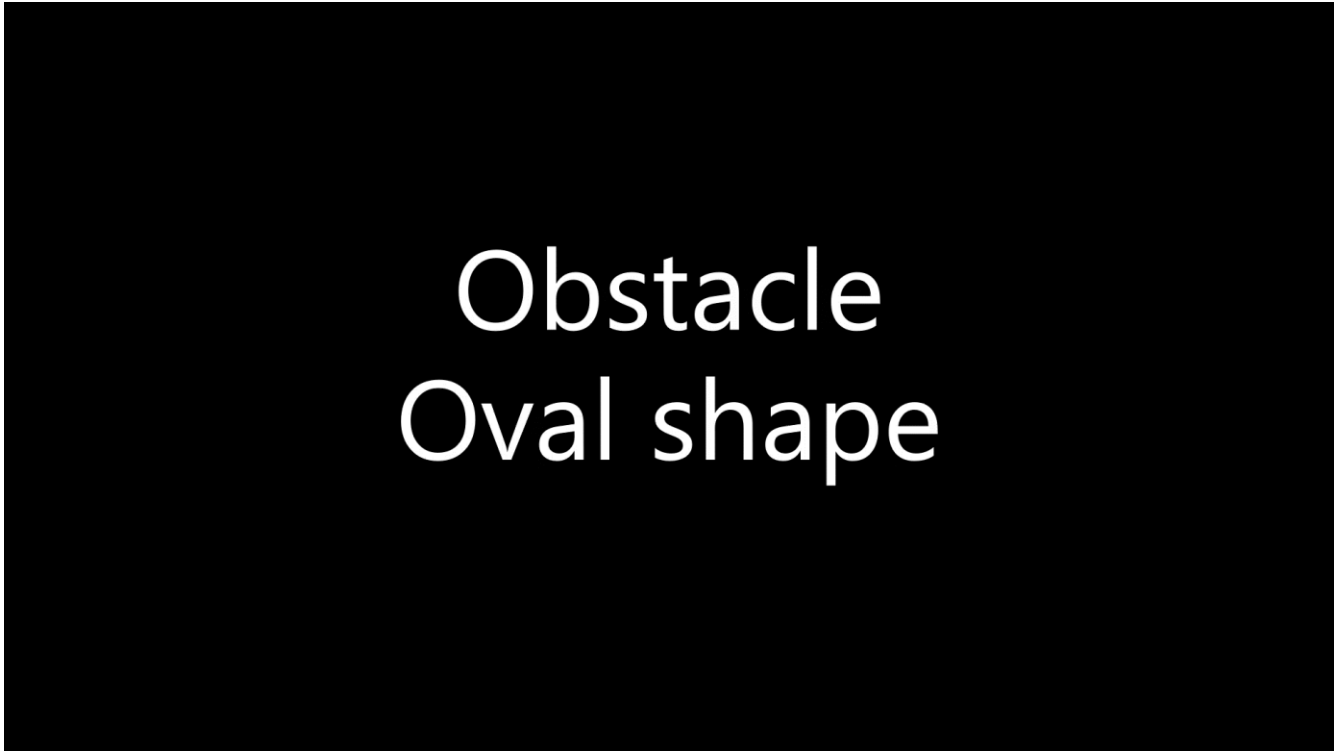
$$\text{speed}_{right} = \sum_{i=0}^{n} \alpha_{right,i} \cdot (1 - \frac{psvalue_i}{psrange})$$

$$\text{speed}_{left} = \sum_{i=0}^{n} \alpha_{left,i} \cdot (1 - \frac{psvalue_i}{psrange})$$

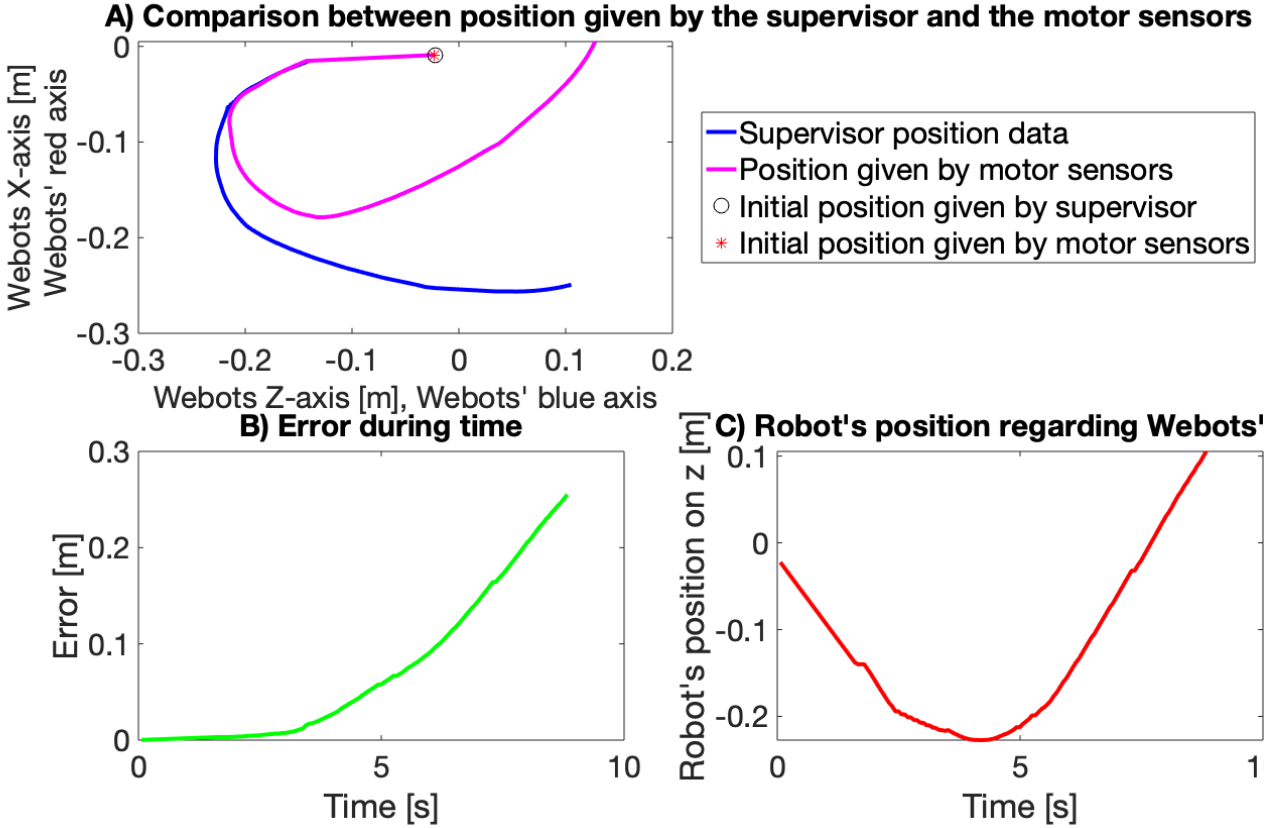
Results: follow line



Results: obstacle



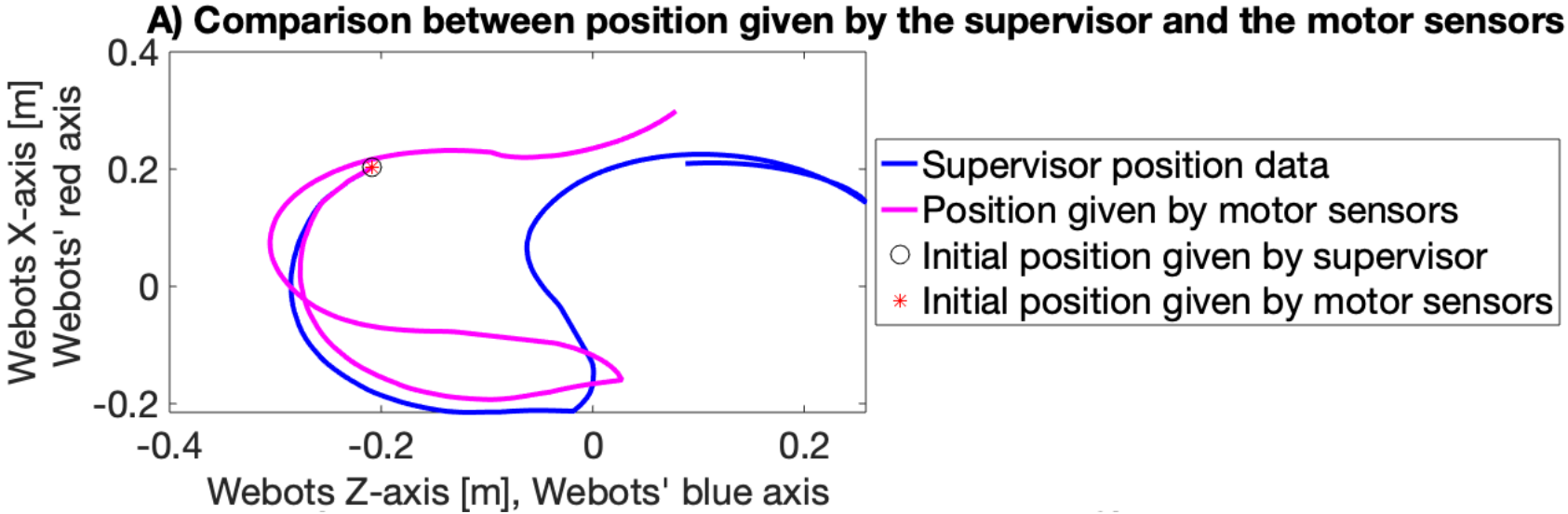
Results: odometry - errors accumulation



Rational angle at time T :

$$\gamma(T) = \theta_0 + \pi + \sum_{t=0}^{T} \theta(t)$$

Results: odometry - problem with the turn back



Achieved

- Detect line
- Follow line
- Avoid obstacles
- Different shapes: oval, square, S
- Different thickness: normal, mean, thin
- Implement odometry

Further developments

- Blue color
- Improvement of the angle for square shape
- Bypass the obstacles
- Adapt method used for odometry

Conclusion

- Reflexion
  - Global reflection for an efficient implementation of the code
  - Understand errors source
- Acknowledge from the project
  - C language and Code debugging
  - Webots
  - Odometry
- Teamwork

Reading and acknowledgements

[1] Lab 8 , Signals, Instruments and Systems course, 19/20
[2] Lab 9 , Signals, Instruments and Systems course, 18/19
[3] Webots User Guide [Online], Available: https://www.cyberbotics.com/doc/guide/index
[4] Week 10 , Signals, Instruments and Systems course 19/20
[5] Advanced Technologies Lab, The University of Michigan, Measurement Correction of Systematic Odometry Errors in Mobile Robots, 4, Available : https://cs.au.dk/ocaprani/legolab/DigitalControl.dir/NXT/Lesson10.dir/paper58.pd

Thank you for your attention !