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
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

```plaintext
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

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 obstacles
        action = 'O';
    }
    break;
Implementation: avoid obstacles

E-puck maximum distance with an object:

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

\[
\begin{align*}
speed_{right} &= \sum_{i=0}^{n} \alpha_{right,i} \cdot (1 - \frac{ps\text{value}_i}{ps\text{range}}) \\
speed_{left} &= \sum_{i=0}^{n} \alpha_{left,i} \cdot (1 - \frac{ps\text{value}_i}{ps\text{range}})
\end{align*}
\]
Results: follow line

About turn
S shape
Results: obstacle

Obstacle
Oval shape
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

A) Comparison between position given by the supervisor and the motor sensors

- Supervisor position data
- Position given by motor sensors
- Initial position given by supervisor
- Initial position given by motor sensors

![Diagram showing comparison between supervisor and motor sensor positions](image)
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

Thank you for your attention!