Road Sign Recognition Project

EL BERRIA Nabil
BERNARD Hippolyte
MOURADIAN Alex

Introduction
Experiment
Analysis Of Road Signs In Matlab
Implementing In Webots
Implementing For Real E-puck
Results
Conclusion

EL BERRIA Nabil
BERNARD Hippolyte
MOURADIAN Alex

EPFL

Bachelor 6 - 2016
1 Introduction

2 Experiment
   Analysis Of Road Signs In Matlab
   Implementing In Webots
   Implementing For Real E-puck

3 Results

4 Conclusion
Goal

Implementation of a C code allowing the e-puck to read a simple road sign with its camera and analyze the picture on board in order to escape the maze.
Sensors

- IR proximity sensors: to avoid obstacles and stop in front of the walls
- Camera: to read and analyze the taken image in order to take a decision
FFT For Horizontal Stripes

The criteria for selecting the appropriate direction will depend on the values of the FFT magnitude peaks of the vectors corresponding to a summation of the pixel through the rows, and the columns.
No variation in the magnitude of the fft.
Multiple magnitude peaks: noticeable pixel variance in the vertical direction.
Advantage Of Working In Webots

Figure : modelisation of the e-puck and the make in Webots

- The robot and it’s movement are already implemented it allows us to focus only on the road sign recognition part.
- Since we’re simulating on a computer, there are no constraints on the resources available.
- The debugging is easier, we can simulate the comportement of the robot much faster.
E-puck Movement

In order to make the robot advance while avoiding the obstacles of the maze, we implement a function that computes wheels speeds using Braintenberg coefficients. These coefficients were calibrated in order to reach high speeds and increase performance in the maze.
This function is ran indefinitely, until the e-puck is presented with a road sign in front of it.
Rotating The E-puck Without Odometry

- The e-puck microchip has a fixed number of operations per second.
- We use a wait function where a simple operation (subtraction) is executed.
- We calibrate the number of processor operations needed to execute a full turn of the e-puck at a speed of 500:

```c
#define OPS_PER_TURN 1525000

void wait(unsigned long num) {
    while (num > 0) { num--; }
}

wait(I_PER_TURN);
```
Stop Criteria And Setup

The criteria on which the e-puck stops in front of a sign is based on it’s front sensors data. If they reach a certain threshold value, the robot stops and prepares for taking a picture.

```
#define DISTANCE_THRESHOLD 200

// If an obstacle is in front of the epuck, stop
if ((sensor[0]>DISTANCE_THRESHOLD) & (sensor[7]>DISTANCE_THRESHOLD)) {
    e_set_speed_left(0);
    e_set_speed_right(0);
}
```

Once the robot is immobile, leds are lit up to increase the luminosity of the image taken.

```
// Light up the leds
e_set_front_led(1);
e_set_led(0,1);
e_set_led(1,1);
e_set_led(7,1);
wait(I_PER_TURN/50); // Stop before taking picture to avoid blur
```
Black Sign Recognition

The picture is taken at a format of 40x40 pixels. But because of memory limitations, and because Fourier transforms using the e-puck library can be done for vectors with 512 values at most, we isolate two stripes of the picture with 256 values each. These stripes correspond to a few vertical and horizontal lines of the picture.

```c
// Picture is inverted for our epuck!
for (i=0; i<strlen(pic); i++) {
    if (i<FFT_BLOCK_LENGTH) { // FFT_BLOCK_LENGTH=256
        col[i] = (int)(unsigned char)pic[i]; // Stock the first 256 values of the pic
    }
}
```

Mean pixel value of stripes is computed and if it’s inferior to threshold value, the sign is black and the robot has to turn back.

```c
if (black_criteria<40){ // Turn back if black sign detected
e_set_speed_right(-500);
e_set_speed_left(500);
wait(T_PER_TURN);
}
```
Other Road Signs Recognition

A filter is used to increase the picture’s quality. Using the mean pixel value we assign black (0) and white (255) to all of the pixel values:

```python
// Conversion to black and white values
for (i=0;i<FFT_BLOCK_LENGTH;i++){  
  if col[i]<=mean_pixel_value_col    
    col[i]=0;                      
  if col[i]>mean_pixel_value_col    
    col[i]=255;                    
}
```

We compute the fourier transforms magnitudes for both stripes.

```python
// FFT on rows
e_subtract_mean(row, FFT_BLOCK_LENGTH, LOG2_BLOCK_LENGTH); // Substract mean of vector
e_fast_copy(row, (int*)sigCmpx, FFT_BLOCK_LENGTH); // Copy vector into sigCmpx
e_doFFT_asm(sigCmpx); // Do FFT. The result is stored in sigCmpx.

// FFT Magnitude rows
for(i=0;i<FFT_BLOCK_LENGTH;i++){  
  mag_sqr_row[i] = sigCmpx[i].real*sigCmpx[i].real + sigCmpx[i].imag*sigCmpx[i].imag; // Calculate the magnitude
}
Other Road Signs Recognition

If the black patterns are horizontal, the sum of the fft magnitude values of the vertical stripe should be higher than the fft magnitude values of the horizontal stripe, and vice versa. We base our direction decision based on this criteria.

```c
if (criterial_row < criterial_col){ // If variation in columns are > variation in rows :Turn left
e_set_speed_right(500);
e_set_speed_left(-500);
wait(1*PER_TURN/2);
}
```
Success percentage for different light conditions (40 tests for each road sign). (a) is the black sign, (b) and (c) are respectively the horizontal and vertical and the (d) and (e) the horizontal with noise and vertical with noise.
Performance In The Maze

Presentation Video

We got an average time of 36.4 [s]
Conclusion

- Finding a criteria for recognition in matlab
- Implementing using Webots
- Implementing for the real e-puck
- Difficulties
- What we learned?