E-punk in Webots

Wall Following on Reverse

Fragkoulis Logothetis
Tuc University of Crete
Electrical and Computer Engeneering
Autonomous Agents
Crete
flogothetis95@gmail.com

I. Introduction

II. Mobile robots are both fascinating objects and the result of the fusion of multiple competences. This fascination leads to the organization of plenty of robotics contests worldwide annually . From an engineering point of view, the design and control of mobile robots requires skills in many disciplines such as mechanics, electronics, energy management, computer science, signal processing, and automatic control. The combination of these two aspects (fascination and inter-disciplinarily) makes mobile robots an excellent educational platform that enables students to address a broad range of engineering fields.

A. *Simulation Tool*

Several simulators support the e-puck. Among them, we use Webots. Webots is commercial and supports three-dimensional physics through the ODE10 library.

B. *Our Exercise*

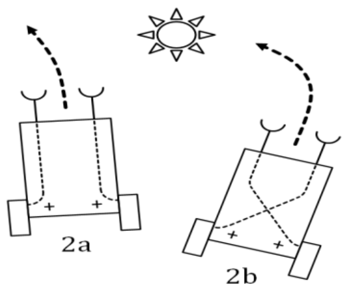
As a part of our Laboratory Exercise, our Proffessor has commissioned us to prepare some tests on e-punk .Specifically , one of those tests was to change the basic-webots code to make e-punk going on reverse as it follows a wall

III. Useful Tips of the Design

A. *E-punk works as A Braitenberg vehicle*

A Braitenberg vehicle is an agent that can autonomously move around based on its sensor inputs. It has primitive sensors that measure some stimulus at a point, and wheels (each driven by its own motor) that function as actuators or effectors. In the simplest configuration, a sensor is directly connected to an effector, so that a sensed signal immediately produces a movement of the wheel.

Depending on how sensors and wheels are connected, the vehicle exhibits different behaviors (which can be goal-oriented). This means that, depending on the sensor-motor wiring, it appears to strive to achieve certain situations and to avoid others.



.

B. *Wall Following; a clever dicession or not ?*

- The wall follower, the best-known rule for traversing mazes, is also known as either the *left-hand rule* or the *right-hand rule*. If the maze is simply connected, that is, all its walls are connected together or to the maze's outer boundary, then by keeping one hand in contact with one wall of the maze the solver is guaranteed not to get lost and will reach a different exit if there is one.

C. *Structure of the Robot*

The e-puck model in Webots is depicted in figure 8.2. This model includes support for the differential wheel motors (encoders are also simulated), the infra-red sensors for proximity and light measurements, the accelerometer, the camera, the 8 surrounding LEDs, the body and front LEDs; the other e-puck devices are not yet simulated in the current model. Besides standard epuck devices, the model also provides two slots that can be used to extend the functionality and to equip the robot with additional devices. In particular, the ground sensors module extension of the real e-puck robot is modeled in Webots to provide 3optional infra-red sensors pointing to the ground in front of the robot.

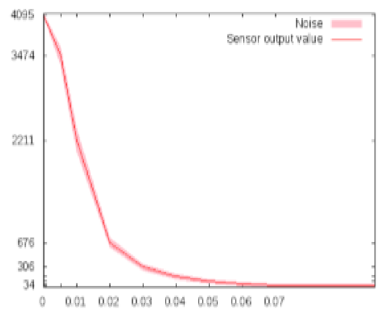


Figure 8.3: Proximity sensor response against distance

The impulse response of Proximity Sensor has a great amount of Noise .As a result SNR is big and these situation lead to difficulties which can't be solved easily . The work become more and more buggy as our robot has only a few P.R sensors behind .As a result , traversing on reverse it's a strange process

D. Speed of Actuators = f (Sensor_values)

As it mentioned above ,e-puck used to work following Braitenberg's Idea . But it's important to include that the speed of the actuators it's a value of a function

Speed = (AvoidCollisonMatrix[i] [+ -] OtherMatrix[i]) *distance[i]

IV. Implementation

.

A. Built the appropriate Matrix (Weights)

To achieve the above goal ,it's important to design smart matrix to avoid e-puck collisions and speed matrix factor to control the speed of each actuator .From am avoid collision point of view ,the matrix should bring a wall attraction .

Ex1 (Right Wall Follower).

```
double[] collisionAvoidanceWeights = {0.0,0.0,0.015,0.03,-0.14,0.08,0.0,0.0};
```

Wall attract

```
double[] slowMotionWeights = {0.0125,0.00625,0.0,0.0,0.0,0.0,0.00625,0.0125};
```

B. Right Wall Following

It's approximately easy to make e-punk follow a wall by controlling (left /right) side sensors and the weights .

```
if (distance[3]+distance[2] > 2200 or distance[3]+distance[4] >1300 || distance[4]+distance[5] > 3000) {

        if (!turn) {
          turn = true;
          right = true;
        }
        if (right) {
          ledValue[2] = 1;
          leftSpeed  =  maxSpeed;
          rightSpeed = -maxSpeed;
        } else {
          ledValue[6] = 1;
          leftSpeed  = -maxSpeed;
          rightSpeed =  maxSpeed;
        }
      } else {
        turn=false;
      }
```

a.

Simulation (Videos)

There are 2 video of the simulation .The first provide a full view of a Rat0 (Right wall follower) and the second video presents botha Rat0 and Rat1(Left wall Follower) together in the same world.

References

[1] Webots User Guide

[2] The e-puck, a Robot Designed for Education in Engineering Francesco Mondada1 , Michael Bonani1 , Xavier Raemy2 , James Pugh2 , Christopher Cianci2 , Adam Klaptocz3 , Stephane Magnenat ´ 1 , Jean-Christophe Zufferey3 , Dario Floreano3 , Alcherio Martinoli2

[3] Wikipedia Braitenberg vehicle