Series 01 – Setup & Braitenberg vehicles

Robotics, BSc Course, 2nd Sem., Dr. Julien Nembrini, Manuel Mondal

Handout on February 21st 2019 Due on March 3rd 2019

Readings

— Study the lecture notes available on Moodle
— the "Robotics Cheat Sheet" available on Moodle
— Webots Reference Guide → Nodes and API functions → DistanceSensors [1] : Infra-Red Sensors
— "Vehicules", V. Braitenberg, 1984, Chap. 1-4 (pp 1-19) [2]

The reading material is available on Moodle

Sections to be completed in the Report 1 template

— Section 2.1 *Sensors* → *Proximity infra-red sensors*
— Section 3.1 *Behaviours* → *Braitenberg vehicle*

Installations

— Webots R2019a (see "Robotics Cheat Sheet")
— Anaconda (Python 3.6 distribution) `https://www.anaconda.com/download/`

Get started with Webots

Get familiar with the Webots environment and try to connect with the robots as explained in the "Robotics Cheat Sheet" document from Moodle.

1. Webots Reference Guide www.cyberbotics.com, available from Webots→Help or by pressing F2(link)
2. Valentino Braitenberg : "Vehicules : Experiments in Synthetic Psychology", MIT Press, 1984

E-puck first steps

This part is intended to guide you for your first steps with the E-puck robot. Its aim is to make you familiar with :

— the process of running code on the robot

— the dynamics of the robot

— potential differences between simulation and real robot

This part should not be included in your submitted report. However, you are highly encouraged to do these steps before Lecture LN_02 in order to be able to ask questions.

Forward/backward

Explore the behaviour of the robot when modifying the `S01_forward_backward` controller presented in class

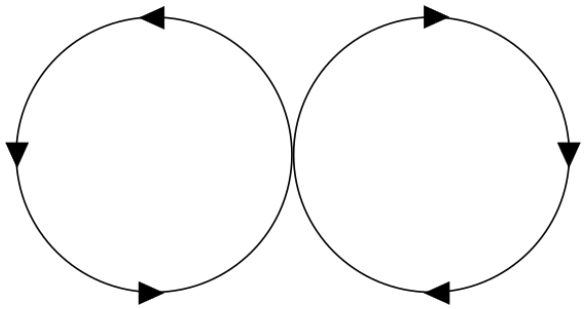
a) in simulation,

b) on the real robot (remote control).

Observe the kind of changes you made and explain their effects on the robot's behaviour.

A controller and a Webots world is available on github (see the "Robotics Cheat Sheet").

Figure 8



Modify the `S01_forward_backward` controller in order to make the robot move in a figure 8 (as depicted above). This should be possible by changing speed values and modifying counters.

a) in simulation

b) on the real robot (remote control)

Observe the differences between the simulation and the real robot and try to find their explanation.

Proximity Sensor Values

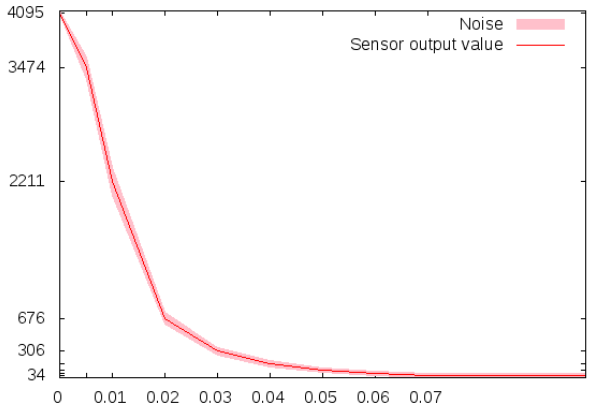


FIGURE 1 – Webots infra-red response graph

Modify the provided controller `S01_IR_record` and use the python script `plot_calibration.py` (or your preferred plotting software) to implement a controller reconstructing the equivalent of the webots infra-red response graph shown in Figure 1. Feel free to reuse code excerpts from the controller `S01_forward_backward`.

Generate one graph with the calibrated values, using only the front proximity sensors (`PROX_RIGHT_FRONT` and `PROX_LEFT_FRONT`) of a **real robot**. Analyze and explain potential differences between your graph and the Webots infra-red response graph. describe in sufficient detail your experimental setup (do not forget to mention the robot number).

Your analysis has to fill in the *Section 2.1 Sensors* → *Proximity infra-red sensor* section in the report template.

Hint : Place the robot near an obstacle and use the code `S01_forward_backward` to drive the robot further away, while recording sensor and counter values. Keep in mind that the calibration step needs the robot to stand away from obstacles at the beginning.

Explorer & Advanced Lover

Implement the Webots controllers for the explorer and advanced lover behaviour as discussed in class, and explain for each controller :

— your chosen weights for the proximity formula,

— the formulas you chose for the speed of the wheels.

Expand the provided controller `S01_basic_lover` to include all proximity sensors.

Exploring Lover

Using the controllers from the previous exercise, design, implement and test a behaviour that :

1) starts by running LOVER
2) if encountering an obstacle move towards it
3) detects when reached equilibrium distance to obstacle
4) switches to EXPLORER until he has moved away from the obstacle
5) switches back to step 1.

Follow these steps :

a) Find a robust way to detect when the robot in LOVER behaviour is at equilibrium : discuss your solution, providing examples when it may fail and proposing solutions to this problem.

b) Using two states (LOVER/EXPLORER) represent schematically the AFSM (using latex or your preferred drawing program). Pay special attention to have transitions only dependent on the current state and to consider all possible events.

c) Implement the behaviour and comment on your solution.

d) Record a video with LEDs visible to indicate the current state (rule : all LEDs on for the LOVER state, off for the EXPLORER state), upload it on your preferred sharing site, and include the link in your report.

Describe and document your implementations. Both of the above exercises have to fill in the *Section 3.1 Behaviours → Braitenberg vehicle* section of the report.

All controllers, a Webots world, as well as a plotting script in python 3 for this series are available on github (see the "Robotics Cheat Sheet").

Reminder : folder structure and naming convention

```
gg_nn_ff.zip/worlds/
            /controllers/
            /gg_nn_ff.pdf
```

gg = group number **00 if undefined**
nn = last name
ff = first name

A python 3 script is provided to test the correctness of your submission.