First experiments with ARGoS: Programming simple behaviours

– Intelligent Robotic Systems –

Andrea Roli
andrea.roli@unibo.it
Dept. of Computer Science and Engineering (DISI)
Alma Mater Studiorum Università di Bologna

1 Play with the example

The files `hellorobot.argos` and `hellorobot.lua` provide an example of the experimental setting (`.argos` file) and controller programming in Lua (`.lua` file). Open and inspect them, then run the experiment with: `argos3 -c hellorobot.argos &`

A comprehensive illustration of the main robot-related functions can be found at:
`https://www.argos-sim.info/plow2015/`

2 Exercises

The following exercises concern the coding of a given robot behaviour. Use a scientific approach to design your controller: once you have written the code, and **before** running the robot, make an hypothesis on the expected behaviour. Then, check your hypothesis against the actual behaviour. If the outcome is as expected, do more tests by changing the initial and environmental conditions so as to add more evidence to support your conjecture. Conversely, if the behaviour is not as expected this means that your *theory* has to be corrected and so the robot program. Discrepancies between expected and actual behaviour are our friends, not enemies!

2.1 Phototaxis

Program the robot such that it is able to perform *phototaxis* (i.e., it goes towards the light). Arena: only perimetral walls and one light bulb. Suggestion: remove the floor picture and all sensors and actuators that are not necessary for the task. Note that some sensors depend on some features of the environment, e.g. the motor ground sensor needs a floor picture, and the light sensors needs a light.

- Variant 1: add an extra light
- Variant 2: what happens if you add actuator/sensor noise?

2.2 Obstacle avoidance

Program the robot such that it is able to move randomly avoiding obstacles (this behaviour is also called *random walk with collision avoidance*). Arena: perimetral walls and some boxes. Test the robot in arenas with different number and shape of boxes.

- Variant 1: what happens if you add actuator/sensor noise?
- Variant 2: try with more robots (just use `distribute` and set `quantity` to a value greater than 1)

2.3 Composite behaviour

Program a robot such that it is able to go towards the light avoiding collisions, i.e. the robot should be able to perform phototaxis with obstacle avoidance. To this aim, you may want to use the code developed in the previous exercises.

Compare the behaviour in the case without and with noise.

Try the controller also with more than one robot in the arena.

Food for thought

- What does exactly mean to "Program the robot such that it is able to perform phototaxis"? Is the task correctly, properly and completely defined?
- About obstacle avoidance: to what extent we are sure to avoid any kind of collision? How can we attain a safe wandering? Did you experience deadlock situations? If so, what might be done to avoid them?
- What are the main difficulties in each of the robot programming exercises?
- Which of the two tasks between phototaxis and obstacle avoidance is harder to program? Why?
- Is it better to focus first on the two basic behaviours and then combine them, or to program a composite behaviour from scratch?
- Does the controller need memory to let the robot achieve the desired task? If yes, why? If not, would it help?
- How would you assess the *performance* of the robot?