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

Instructions

The following exercises concern the coding of a given robot behaviour. Use a scientific approach to design your controller: study the problem and design the control software; 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!

When you are satisfied with the results, read the last section ("Food for thought") and answer the questions raised (just in you mind, or take some notes). Answering those questions help you understand possible problems and find ways to overcome them. Moreover, they prepare the path for next ideas that will be illustrated in the class.

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/`

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. You can also test the behavior in presence of noise (see definition of sensors and actuators; use values around $0.02 - 0.1$).

Suggested variants of the experiments:

- add one extra light;
- what happens if you add actuator/sensor noise? (see definition of sensors and actuators; use values around $0.02 - 0.1$)

2 Random walk with collision avoidance

Program the robot such that it is able to move randomly avoiding obstacles. Arena: perimetral walls and some boxes. Test the robot in arenas with different number and shape of boxes.

Suggested variants of the experiments:

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

3 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?
- Obstacle avoidance: to what extent are we sure to avoid any kind of collision? How can we attain a safe wandering? Did you experience situations in which the robot gets stuck? 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?
- 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?