Multi-Agent Systems

B. Nebel, F. Lindner, T. Engesser — University of Freiburg
Summer Semester 2016 — Department of Computer Science

Exercise Sheet 2

Due: May 9th, 2016, 10:00

Exercise 2.1 (Swarm Formation, 3+3+3)

Your task in this exercise is to implement some of the swarm formation algorithms by Sugihara and Suzuki.[1] For your convenience, an implementation template (`ex02/formation.py`) is already provided in your group's repository. The idea is to control the whole group of agents using the keyboard. Currently, the user can instruct all the agents to randomly *walk around* (by pressing **W**), or to *freeze* (by pressing **SPACE**). Furthermore, it is possible to select and deselect agents with the mouse. Selected agents can later be used as designated corner agents of a polygon.

(a) Implement the Circle and Fillcircle behaviors. The user should be able to trigger them by pressing respectively **R** and **F** on the keyboard.

(b) Implement the Contraction and Fillpolygon behaviors. The user should be able to trigger them by pressing respectively **C** and **P** on the keyboard (e.g., after having selected the corner agents – which will remain on their places – from a circle of agents).

(c) Experiment with the different controllers that are available in the framework (UnicycleController, DifferentialController and KilobotController), as well as different settings for the parameters D and δ, in order to optimize the swarm behavior. If necessary, try to add a simple collision avoidance behavior. Note that for Exercise 2.1, you only have to commit one agent implementation (including all the behaviors), which does not have to be perfect! Explain your implementation choices and briefly describe the problems you encountered.

Exercise 2.2 (Gradient Formation and Localization, 3+3)

The algorithms from Exercise 2.1 assume that each agent knows the relative position of every other agent. Usually, in robotics this assumption does not hold. In the swarm formation approach proposed by Rubenstein et al.,[2] robots have to communicate (by broadcasting and receiving messages) in order to localize themselves within the bulk of agents. Your task in this exercise is to implement gradient formation and localization in 2D-coordinates, starting from the implementation template `ex02/kilobot.py` in your repository.

(a) Implement gradient formation (lecture 4, slide 11) for the agents. Indicate the agents who already know their gradient value by giving them a different (inner) color. If such an agent is clicked on, its gradient value should be printed out to the console.

(b) Implement the agents' 2D-localization via trilateration (lecture 4, slide 12). The optimization problem can be efficiently solved by applying the nonlinear least squares method, e.g., using SciPy's function `curve_fit`. You can find more information about this on the web. [3] [4] [5] Run the simulation until all of the agents have a position estimate. Then make a plot of the estimated and actual positions of all the agents (and commit it to your repository, alongside the implementation).

[1] `http://dx.doi.org/10.1002/(SICI)1097-4563(199603)13:3<127::AID-ROB1>3.0.CO;2-U`
[2] `http://science.sciencemag.org/content/345/6198/795`
[3] `http://gis.stackexchange.com/questions/40660/trilateration-algorithm-for-n-amount-of-points`
[4] `https://en.wikipedia.org/wiki/Non-linear_least_squares`
[5] `http://www.walkingrandomly.com/?p=5215`