SCHOOL OF ENGINEERING AND MATERIALS SCIENCE

Queen Mary University of London

Supplementary Work

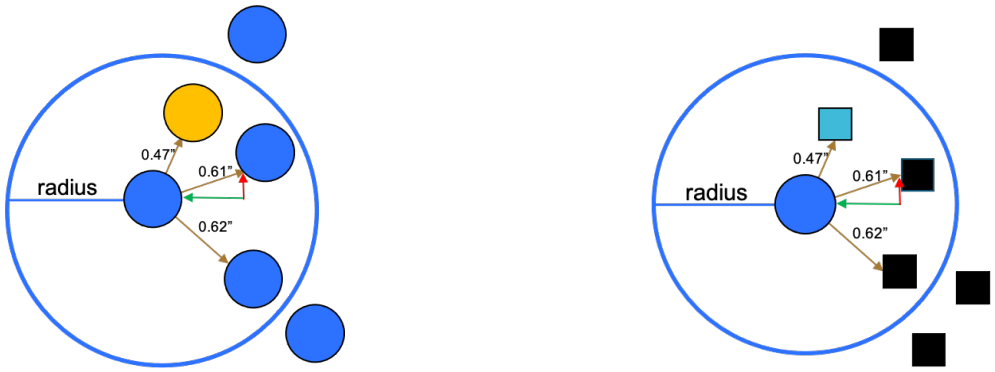
by Joseph Zahar
Supervisor: Dr. Ketao Zhang

April 28, 2021

1 Control of the Nanorobots

This branch covers the mathematical background theory behind the algorithms presented in section 3.3. Three different notions reign the control algorithms of the multi-agent system. Each subsection details the mathematics in each notion and how the swarm locally interact to build an efficient system.

1.1 Nearest neighbour function



(a) Nearest nanorobot (b) Nearest cancer cell

Figure 1: Schematic representation of the nearest neighbour function for each entity (a) the nearest nanorobot is presented in yellow and blue circles define the other nanorobots, (b) the nearest cancer cell is shown in turquoise and the black square represent the other cancer cells.

The nearest neighbour function is responsible for finding the closest entity, cancer cells or nanorobots, of each agent in a specific research radius. The same procedure happens for both entities, where the radius of vision relies on the power of the sensor used. This function is equivalent to the k-NN algorithm for supervised learning techniques applied in classifications and regressions problems by finding the Euclidian distance to the corresponding entity (Jabbar, Deekshatulu and Chandra, 2013). Equations 1, 2, 3 demonstrate the several steps involved in the calculation of the nearest neighbour function. While figure 1 reveal a simplified representation of the mathematical procedure.

$if\ robot_i \in r_v$ (radius vicinity):

$$D_i((x_i, y_i), (x_{i+1}, y_{i+1})) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{1}$$

The nearest neighbour is considered the global minimum of the list:

$$\frac{\partial D^2}{\partial N^2} > 0 \tag{2}$$

or by using:

$$D_{NN} = \frac{D_i + D_{i+1} - |D_i - D_{i+1}|}{2} \tag{3}$$

1.2 Nanorobots alignment function

The nanorobots alignment function implicates the separation and adjustment of the agents towards a flock formation once the distance between two agents is smaller or larger than 1 unit. Two related procedures take part in this process, where each one involves different inputs. Either the nanorobots turn towards their nearest neighbour or away from the average swarm direction, controlled by the main angular direction. Every change affects the overall behaviour of the swarm and its orientation. The mathematical process that induces the nanorobots to turn away and separate from the swarm is displayed in equation 4. The two inputs here are the heading of the initial agent and the heading of its closest neighbour, thus also involving the nearest neighbour function.

$if\ D_{NN} < 1$:

$$f(x) = |\theta_{i+1} - \theta_i| > \eta \begin{cases} \theta_i = 90^\circ \times \eta\ if\ (\theta_{i+1} - \theta_i) > 0 \\ \theta_i = -90^\circ \times \eta\ if\ (\theta_{i+1} - \theta_i) < 0 \end{cases} \tag{4}$$

D_{NN}: Distance with nearest-neighbour.
θ_i: Angular direction of the initial nanorobot.
θ_{i+1}: Angular direction of the closest nanorobot from the initial one.
η: Define the alignment variable.

Equation 6 unveil the operation that leads to the adequate swarm formation, where the agents turn towards and align to the flock. The main difference is based on the average heading of the swarm instead of the initial agent closest neighbour. In order to proceed to equation 6, we need to calculate the average direction of the swarm as shown in equation 5.

$if\ D_{NN} > 1$:

$$< \theta_\mu >= \frac{1}{T} \sum_{T}^{i} arctan(\frac{x_i}{y_i}) \tag{5}$$

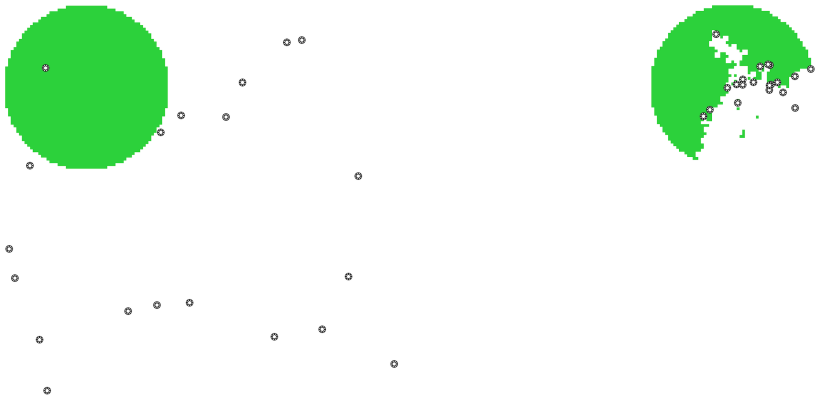
$$g(x) = |\theta_i - \theta_\mu| > \eta \begin{cases} \theta_i = 90^\circ \times \eta\ if\ (\theta_i - \theta_\mu) > 0 \\ \theta_i = -90^\circ \times \eta\ if\ (\theta_i - \theta_\mu) < 0 \end{cases} \tag{6}$$

θ_μ: Average angular direction of the swarm.
x_i: Define the x-increment if the turtle were to take one step forward from its heading.
y_i: Define the y-increment if the turtle were to take one step forward from its heading.
T: Total number of nanorobots.

2 Further applications

2.1 Decontamination floor

The potential applications of swarm robotics are vast and may include any miniaturisation task, collective exploration, targeting and monitoring application that favour the cooperative behaviour of a group of agents to superior performance (Tan and Zheng, 2013). Further applications comprise firefighting drones (Innocente and Grasso, 2019), agriculture and precision farming (Trianni et al., 2016), construction task (Werfel, Petersen and Nagpal, 2014), disinfection robots for hospitals (Rubaek et al., 2016) and decontamination floor use (Sadeghi Amjadi et al., 2021).
In this section, another background environment is created in Netlogo to test the search-dominated motion algorithm's efficacy on a pollution source for decontamination. Furthermore, we will analyse the effect of the population size and speed parameter on the simulation outcome. For this experiment, the background was changed to white patches and the pollution source coloured in lime green. The robots shape and size were changed relative to the real-life scale of the application as seen in figure 3. The rest of the parameters are shown in table 3, and the robot's radius of vision switched to a constant.

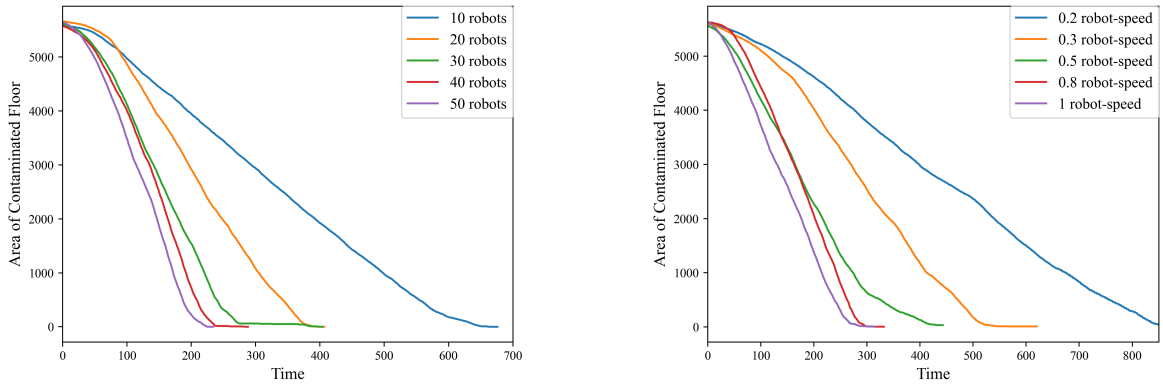


(a) t = 0 s (b) t = 150 s

Figure 2: Decontamination floor simulation where (a) at t = 0 s and (b) at t = 150 s

2.1.1 Investigation of the speed and size of the swarm

The effect of swarm size on the detection and clean up of an infected area was analysed in figure 3a. As the number of robots acting on the 2D grid increase, the time needed to control the contaminated area decrease dramatically from 700 s for ten bots to 210 s for fifty bots and the slope of the curves get sharper. Also, there are almost no significant changes with the results obtained for 40 and 50 robots, setting a benchmark to obtain efficient result while adopting a cost-effective approach by minimising the number of robots for the same outcome. Figure 3b inspect the change of speed on the efficiency of the model for a fixed number of robots. The swarm population was set to 40 bots and the linear velocity v_r altered from 0 to 1 patches/second. Similar observation from figure 3a can be made since the decontamination time significantly decreases with higher velocity.



(a) Number of robots

(b) Speed of the robots

Figure 3: Area of contaminated floor vs time for (a) various swarm population and (b) robot speeds

In conclusion, the algorithm generated in this study can be utilised for many swarm robotics applications while still providing a promising outcome. As shown for cleaning a pollution source, the search-dominated algorithm was very effective in decontaminating the area efficiently.