1 Lab 7: Multi-Level Modeling of Robotic Swarms

This laboratory requires the following equipment:

- C++ programming tools
- Webots
- Matlab

The laboratory duration is up to five hours. The TAs will leave after 3 hours, but if you want to continue, talk to the TA about returning the hardware later. We encourage you to take notes during the course of this laboratory to aid in the exams. A solution will be posted a few days after the lab session.

Office hours

Additional assistance outside the lab period (office hours) can be requested using the dis-ta@groupes.epfl.ch mailing list.

Information

In the following text you will find several exercises and questions.

- The notation $\mathbf{S_x}$ means that the question can be solved using only additional simulation.
- The notation $\mathbf{Q_x}$ means that the question can be answered theoretically, without any simulation; if you decide to write a report, your answers to these questions should be submitted in your report. The length of answers should be approximately **two sentences** unless otherwise noted.
- The notation $\mathbf{I_x}$ means that the problem has to be solved by implementing a piece of code and performing a simulation.
- The notation $\mathbf{B_x}$ means that the question is optional and should be answered if you have enough time at your disposal.

To prepare you for the exams, we show an indicative number of points for each exercise between parentheses.

2 Modeling Robotic Swarms

In this lab you will first develop models for a simple scenario, where the robots are either in “wandering” state or “collision” state. After that, you will look into a more complex case study where the robots can also enter a “collaboration” state.

As you have seen in the lecture, there are several abstraction levels at which we can model a robotic swarm. The idea of modeling at each of these abstraction levels is summarized below:

- At **Submicroscopic** level, each robot is seen as a physical entity composed of several parts, each of which is affected by a certain physical phenomena. This is the level at which we carefully look into, and eventually model different physical details such as the wheel slip, noisy communication channels, individual sensors, etc. At this level we can always track trajectories of individual robots.

- At **Microscopic** level, each robot is considered as an atomic entity following a certain behavior usually expressed as a state-machine. In the specific subcategory of non-spatial microscopic models, we no longer look in detail into the trajectories of the agents. The model consists of states for an individual agent which are exactly the same states for a real robot; however, the state transitions are probabilistic rather than deterministic or being event-based as implemented in a real robot controller.
- At **Macroscopic** level, the model continues to consist of states which are normally the same states for a real robot, but can also be new states resulting from aggregation of the robots' states. However we no longer care about the state of individual agents, rather we are interested in the average number of robots in a certain state and the average rate of transition between states.

As briefly explained above, you can see that it is possible to model the behavior of a robotic swarm while skipping exact representation of some details, and merge them in some other high-level parameters in the model. For example one can overlook the robots' trajectories in the environment and solely model the *population dynamics*, i.e. the time-dependent value of the number of robots in a certain behavioral mode or state, by setting the right values for transition probabilities. To do that, we assume that a robot changes its state with a certain probability that is a function of the other robots' states and also the environment. This probability will affect all robots in a certain state, and thus will change the number of robots in that state at every time step considering a time-discrete model. The following section elaborates on the above discussion.

Q1 (3): Consider the "law of large numbers", which states that the average of the results obtained from a large number of trials approaches the expected value, as more trials are performed. For the case we are dealing with here, i.e. experimenting with and modeling a robotic swarm, what do you think are the parameters that according to the law of large numbers need to be large for our eventual model outcomes to match the reality closely? Name at least two cases.

2.1 Detection Probabilities

We define **geometric probability** g_i as the fraction of time that an individual robot, performing an unbiased random walk all the time, spends within a certain fraction A_i of an arena of area A. If A_i is the footprint of an embodied object (e.g., a robot, another obstacle), we refer to A_i as the **detection area** of the object i in the arena. You can think of this probability as how likely it is that a dart would hit a board within one of these areas. Transporting this metaphor to robots instead of darts, the validity of geometric probability in this realm is a result of the fact that we neglect the robots' individual trajectories and assume them to be uniformly distributed within the arena – i.e. hopping around randomly.

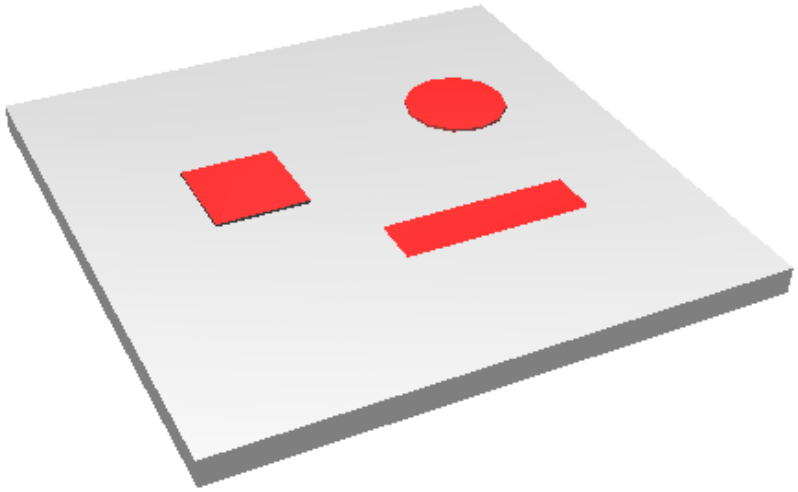


Figure 1: A Webots arena where circular, square, and rectangular objects have been marked. All objects have the same area.

Q2 (3): Imagine an experiment with five robots performing an unbiased random walk within the arena of area A, shown in Figure 1. The rectangle, the square, and the circle all have the same area A_0. How much time would a robot spend on each of the zones with respect to the total experiment time?

Q3 (3): If you plan to validate Q2 experimentally, would you rather choose a few very long experiments, or a lot of short experiments, each time placing the robots at random initial positions? Discuss.

2.2 Encountering rates and interaction times

The geometric probability is a timeless probability that only reflects geometric ratios. However, when a robot is moving on a 2D plane in a discrete-time simulation, we are interested in the probability of encountering an object *per* time step. In a time-continuous simulation, where time steps are infinitesimally small, this probability reduces to a quantity called the *encountering rate*. One can calculate encountering probabilities per time step and the encountering rate as follows:

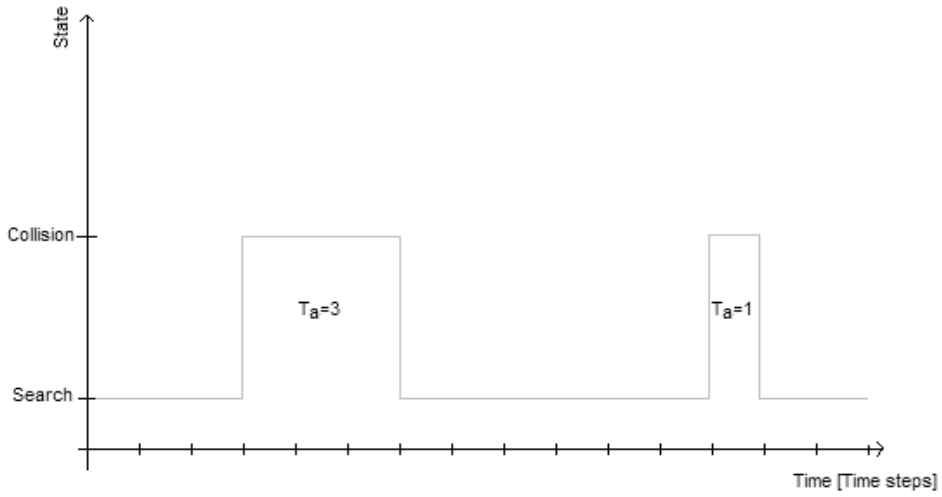
Given: a) a homogeneous system of robots, b) the (reasonably small) time step length of our simulation T_{it} (generally, a time discretization unit of our model), and c) a transformation $c(\Theta)$, with Θ being the parameters describing the robots. We can calculate the **encountering probability** p_i per time step of an object (assumed constant over considered time interval) as

$$p_i = c(\Theta)\, g_i\, T_{it}$$

i.e. by multiplying the encountering rate $c(\Theta)g_i$ with the length of one time step T_{it}. $c(\Theta)$ then operates as a **conversion factor** from geometric probabilities g_i to encountering probabilities p_i as a function of the robot parameters.

Note that the encountering rate of an object is the rate at which a robot encounters an object *while* it is searching. Thus, this rate is calculated as the ratio between the

number of times an object is hit and the total amount of time spent in search mode, and not the total number of time steps.



The above graph depicts the number of time steps spent by a robot in two states: search and collision. The encountering rate with the object i can be calculated as follows:

$$r_i = \frac{hits}{time\ in\ search} = \frac{2}{11}$$

One can also calculate the average collision duration, which is the sum over all collision times divided by the number of hits, yielding E[T_a]=(3+1)/2=2 time steps for the above example. The encountering rate can thus be expressed as $\frac{hits}{time\ in\ search} = \frac{hits}{total\ time - hits * average\ collision\ duration}$. Note that the factor T_{it} enables a straightforward conversion between the different time steps, which might be different from the sampling time used in the model (e.g., *50 ms* in Webots and *1 s* in a higher abstraction model).

2.3 Behavior and population dynamics

Consider a wall-bounded arena of area A with two species of robots, black (B) and white (W). The number of Bs is N_B and the total number of robots is N_0. All robots, initially single (S), wander in the arena avoiding walls only, and the Ws are chasing the Bs. The chase takes place in two steps. Upon first encounter between a B and a W with probability p_1, the W locks onto the B forming a standing duplet D. Then, if a second W joins a D with probability p_2 within a time T_D, they form a standing triplet T for a time T_T, after which the robots in T are released; if not, the robots in D are released after T_D.

Q4 (4): Draw a PFSM for the behavior of Bs and Ws in this process.

Q5 (4): Write the macroscopic analytical model describing the populations of single robots $N_B^s(k)$ and $N_W^s(k)$ (please specify equation factors as a function of the parameters given above).

3 A Multi-level Model of a Collision Avoidance Experiment

This section will introduce you to different modeling abstraction levels that build upon the concepts shown in the lecture and above.

3.1 Submicroscopic Model

In our case, the actual implementation of our submicroscopic modeling level is done in a Webots simulation. Extract the archive using the following command: `tar xvfz lab07.tar.gz`. Open the world `5robots.wbt` in Webots and run the simulation. (Note that the controller codes are already compiled. You can however recompile them from within Webots). As you observe, the behavior of the robots is very simple. The robots try to move straight ahead. In this case, the wheel speed is calculated as a weighted sum of the sensor readings (Braitenberg) in order to move away from obstacles. One robot is endowed with the controller `encountering_probabilities` which measures the encountering probability $p_R = p_r(N_0\text{-}1)$ of colliding with any other robot, and the interaction time. p_r is defined as the encountering probability of colliding with one of the other robots and N_0 is the total number of robots. The supervisor controller (`wrap_around_big`) keeps track of the number of robots that are within a specific range of each other, and can therefore be considered as being in collision avoidance. The distance between two robots hereby defines a collision: if the distance between two robots is below a certain threshold, those robots are considered to be in collision. The supervisor terminates the experiment after 30 minutes.

S6: By running the simulation several times in fast mode one can gather a significant amount of data. You can use the script `run` (`./run 10`) to run 10 iterations of the simulation in fast mode and gather data, which is saved in the `/data` folder (overwrites existing files!). The controllers can store the average encountering probabilities, and average number of robots in avoidance for each experiment. Also, the duration of every collision (for the robot running the controller `encountering_probabilities`) is stored. Data gathered from running the experiment for 10 times each time for 30 minutes is provided in the folder `lab07/webots/data`.

Run the script `experiment_S6` in Matlab, to display the average and standard deviation of the above quantities over all experiments, as well as the histogram of the collision durations.

Q7 (4): Note the averaged quantities and their standard deviations from the previous question. The detection area of a single robot can be measured using Webots. Calculate the conversion factor $c(\Theta)$ from geometric probabilities to encountering probabilities, using your simulation results as well as the geometric detection probability of a single robot g_i (which you can compute assuming that the detection range is a circular area of radius 57 mm), and with T_{it} equal to 50 ms.

3.2 Microscopic Model

In a microscopic model, each robot is simulated by one agent. The simulation keeps track of the state of each individual agent. The states that an individual agent can assume are exactly the same states a real robot can be in, given an arbitrary state granularity of interest defined *a priori* for all modeling levels. At this modeling level the state transitions are probabilistic rather than deterministically implemented as in a real robot controller. They are affected by noisy interactions with other robots or the environment.

S8 (6): Open the file `collisionmodel.cpp` that you find in the folder `microscopic`. Here, a simple microscopic simulation as described above is already implemented. Change the definitions of `PR` and `T` according to the values that you collected above. Compile (`make`) and run the model (`./collisionmodel`), and look at the output on the screen. Here you see the average number of robots in the search (N_s) and collision avoidance (N_a) states. Compare these values with those measured in Webots, and keep track of them in your notes.

Q9 (4): Draw a finite state machine for the collision avoidance experiment. Label the states as well as the state transitions. What causes a state transition in Webots, and what causes a state transition in the microscopic model? Have a look at the code, if you are not sure.

S10 (4): The microscopic model simulates 10 minutes in one experiment, and averages over 100 experiments. One single experiment gets recorded each time (`/data/microscopic.txt`). Analyze this data using the Matlab script `experiment_S10`.
What do the blue and the green line depict (you will need to zoom in)? What would you need to record using a supervisor controller to generate a similar plot from Webots?

S11 (4): You can plot the histogram of the average collision duration using the Matlab script `experiment_S11`.
Compare the result with the histogram you got for your Webots data (`experiment_S6`). Pay attention to the maximal collision duration in Webots and the microscopic model, as well as the shape of the histogram.

S12 (4): Instead of using a probability to leave the collision state, one can also assume that the robot spends a fixed time in collision whose numerical value corresponds to the average time measured in a more realistic implementation (e.g., a submicroscopic simulation or real robot experiments). Make the relevant changes in `main()` in the `collisionmodel.cpp` file, in order to enable this behavior. Recompile and run. Compare the result you got with the results you measured earlier. What do you observe?

3.3 Macroscopic Model

You have seen above that it is possible to describe key properties of a swarm robotic system at a higher abstraction level. In this process, you might have noticed several aggregation processes: first, we calculated the *average* encountering rate from our simulation, as well as the *average* collision duration. In this part and at the highest

modeling level, the metric we use to describe the robotic swarm is the *average* number of robots in each state.

We can now think of a mathematical formula describing the average number of robots in each state, given the probability that individual robots change their state.

Q13 (2): Given the probability $p_R = p_r(N_0 - 1)$ that an individual robot encounters any other robot, and assume that you have $N_s(k)$ robots in search mode at time k. How many robots will encounter a collision during the time step k?

Q14 (6): Assume a probability p_s for a robot to resume search, after being in collision. We might be interested in knowing how this probability translates into time dependent changes in the population of the robots in a certain state. Formulate now a difference equation of the form $N_s(k+1) = N_s(k) + \cdots$ that keeps track of the evolution of the population of the robots in the searching state.

Q15 (6): What is the expression for the number of robots $N_a(k+1)$ in avoidance at time $k+1$? Formulate $N_a(k+1)$ in two different ways, as a difference equation and as function of N_0 and $N_s(k+1)$.

Q16 (8): Consider the difference equation for the search state $N_s(k+1) = N_s(k) + \cdots$ you derived earlier. Can you tell if the average number of robots in the search ever converge to a steady state? If so, calculate it. Will that always happen or only under certain conditions?
Hint: Express the difference equation in the form $f(k+1) = af(k) + b$. This is called an affine dynamical system. Then, for the steady state, try to compute $N_s(k)$ at the limit $k \to \infty$.

4 A Multi-Level Model of a Collaborative Experiment

In this section we introduce a slightly more complex experiment, which uses behaviors developed in previous labs. The robots try to collaborate for the stick-pulling task.

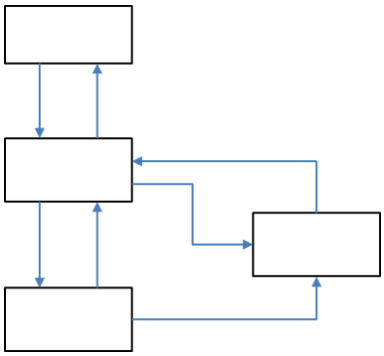
The behavior of the robots can be described as follows: the robots wander in the arena until they detect a free stick. Pulling the stick upwards, a robot waits for a collaborator. If no other robot arrives to help during time t_{wait}, waiting is abandoned and the robot drops the stick back and continues searching. Otherwise, the two robots collaborate, and then resume searching. The number of sticks to pull in the arena is assumed to be constant, as the sticks are replaced in a random place after being handled. We assume that local communication among the robots is loss-less, i.e. that there are never two robots waiting at the same time holding on to a stick. The parameter t_{wait} is a crucial parameter for the collaboration rate, i.e. the ratio of successful collaborations over all time steps, which we want to define as the performance metric for this experiment.

Q17 (4): What do you expect to happen if the parameter t_{wait} is too high? What happens if it is too low?

We will now try to build a probabilistic model based on the building blocks developed in the previous sections. For that, at the microscopic level, we describe the individual robot behavior as a PFSM, with transition probabilities and state durations

we derive as described above (i.e. the probability to find a stick, and the time robot spend while avoiding each other).

Q18 (6): Draw a PFSM using the framework given below. Label the states and transitions with meaningful names. One can simplify the PFSM by assuming that that the robots leave the stick and resume searching as soon as they find a collaborator. Draw a PFSM with two states that describes the simplified model.



A non-spatial microscopic model has been implemented in C++. Hereby, the program keeps track of the individual robots' and sticks' states and counts the number of successful collaborations, while state transitions are generated by random numbers reflecting the probabilities of the above model. In contrast to a submicroscopic model, specific intra-robot details (e.g., placement and cone of view of sensors, non-holonomicity of the vehicle), the trajectories and positions of robots and objects in the environment are ignored. Thus, it allows much faster simulation of different environmental or robotic parameters.

You find the microscopic model for the above experiment in the directory `microscopic/`. After compilation using `make`, you can run it by typing `./testr {robots} {twait}`, where the parameters allow you to specify the number of robots and desired waiting time.

S19: Run the program for different waiting times (between 0 and 10 seconds) and 4 robots (The maximum number of robots allowed is 10).

Q20 (4): How does the waiting time relate to the collaboration rate (denoted *co*)?

In order to derive a macroscopic model for this experiment we can leverage the structure of the simplified PFSM developed for the microscopic model. We can also start with a PFSM that describes the swarm behavior. Such PFSM has the same structure as the one that models a robot's behavior at the microscopic level, however the states and the arrows linking them have a different interpretation. Here we can think about each state variable as a real number describing the average number of robots in that specific state (e.g. Ns(k), and Nw(k) as the average number of robots searching, and waiting alone, at time step k respectively). We will have N0=Ns(k)+Nw(k) the total number of robots, and M0 the total number of sticks. The arrows linking the states will then represent the flow of the robots changing states. Assume that right after entrance of the second robot, both robots transition to a search state.

Q21 (4): Draw a PFSM that describes the overall population dynamics, i.e. at the macroscopic level.

Q22 (6): Assume p_{stick} is the probability of finding one stick out of the total number of sticks scattered in the arena, p_{collab} is the probability that one searching robot finds one waiting robot, and that there are m sticks in the arena. In order to develop a model describing the dynamics of robots' states, first derive the expressions $\Delta_w(k)$ for the average number of robots that enter the wait state, $N_w(k)$, and then derive an expression $\Delta_{collab}(k)$ for the average number of robots that *leave* a stick after collaboration.

This allows us to formulate the following difference equation that describes the dynamics of the search state population:

$$N_s(k+1) = N_s(k) - \Delta_w(k) + \Delta_{collab}(k) + \Delta_w(k - t_{wait})\Gamma(k - t_{wait}; k)$$

Q23 (2): What does Γ represent? *Hint: "Γ represents the fraction of robots that…"*

Q24 (4): Formulate a similar equation for the waiting state.

Q25 (2): In general, is the macroscopic model linear or non-linear? Why?

Q26 (3): Show that the average time spent in a state can be calculated by taking the inverse of the (constant) probability to leave this state. You can do this by summing all possible waiting times multiplied by the probability that they occur.

$\sum_{k=1}^{\infty} kx^k$	$= \frac{x}{(1-x)^2}$

5 References

[IJRR2004] **Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation** / Martinoli, A.; Easton, K.; Agassounon, W. / *Int. Journal of Robotics Research*, Num. 4, Vol. 23 (2004), pages 415-436