1 Lab 5: Multi-Level Modeling of Robotic Swarms – Part I

This laboratory requires the following equipment:

- C++ programming tools
- Webots
- Matlab

Information

In the following text you will find several exercises and questions. The type of question is indicated between parentheses:

- The notation **S** means that the question can be solved using only additional simulation.
- The notation **Q** means that the question can be answered theoretically, without any simulation.
- The notation **I** means that the problem has to be solved by implementing a piece of code and performing a simulation.
- The notation **B** means that the question is optional and should be answered if you have enough time at your disposal.

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.

1. **(Q)** 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.

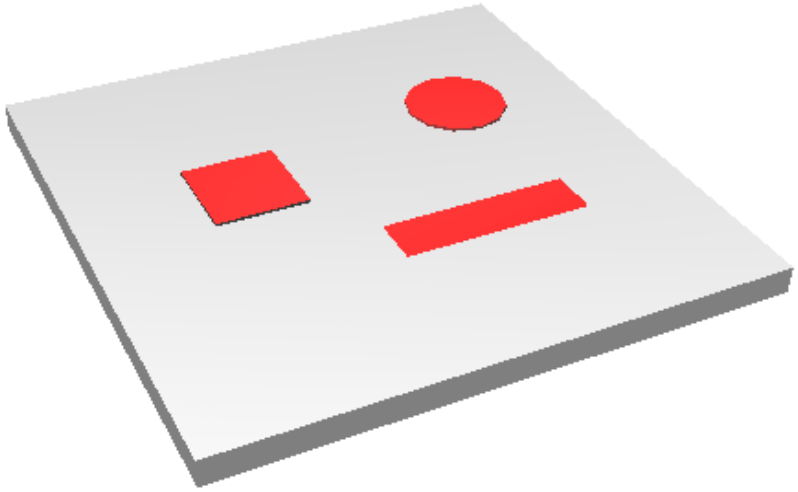


Figure 1: A Webots arena where circular, square, and rectangular objects have been marked. All objects have the same area.

2. **(Q):** 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?
3. **(Q):** 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 poses? Discuss.

2.2 Encountering rates and interaction times

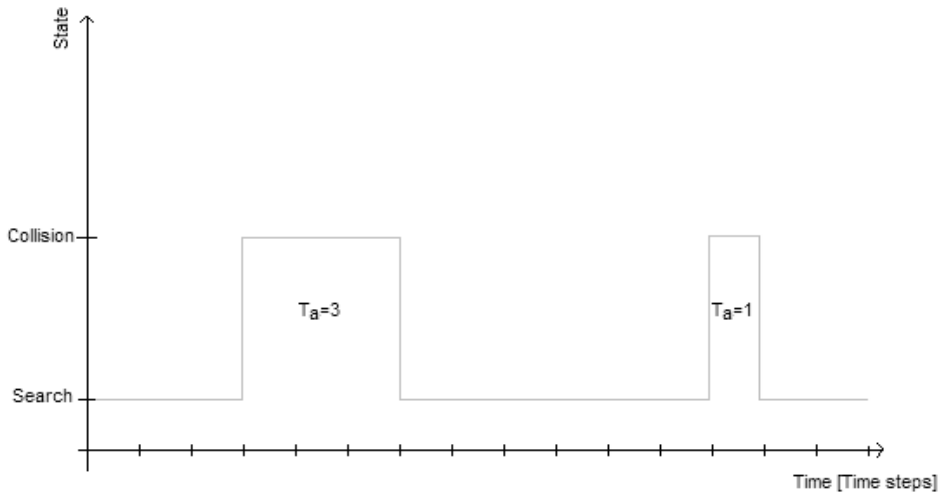
The geometric probability is a timeless probability that only reflects geometric ratios. However, when a robot is moving on a 2D plane in a discrete-time simulation, we are interested in the probability of encountering an object *per* time step. In a time-continuous simulation, where time steps are infinitesimally 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 (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 i (assumed constant over considered time interval) as

$$p_i = c(\Theta)\, g_i\, T$$

i.e. by multiplying the encountering rate $c(\Theta)g_i$ with the length of one time step T. $c(\Theta)$ 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* 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 δ. Then, if a second W joins a δ with probability p_2 within a time T_δ , they form a standing triplet τ for a time T_τ, after which the robots in τ are released; if not, the robots in δ are released after T_δ.

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

5. **(Q):** 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 lab05.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. 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.

6. **(S):** 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 (for Windows, see below), 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 `lab05/webots/data`.

   Note: If you think the script malfunctions (crash or no data generated), then close all instances of webots and run the script again.

   Windows: Unfortunately, the script works only on Linux (and perhaps Mac). Therefore, we have generated the data for you and provided it in webots/data/ folder.

   If you use would still like to run the simulations, see instructions in the end of the document.

   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.

7. **(Q):** 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 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.

8. **(I):** Open the file `collisionmodel.cpp` that you find in the folder `microscopic`. Here, a simple microscopic simulation as described above is implemented. To use this code, do the following:

(a) At the top of the file, change the definition of the variables `PR` and `T` according to the values that you collected in the section above.
(b) Look at the functions `ProbabilisticMicro::Step()` and `DeterministicStateDurationsMicro::Step()`. Implement the probabilistic and deterministic rules for state transitions from search to collision avoidance and vice versa. There are 4 places where you have to change code, and they are marked with a TODO comment.
(c) Compile (`make`) and run the model one time by providing the commandline parameter (`./collisionmodel 1`) (The default number of runs is 100), and look at the output on the screen. Here you see the 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.

Note: use mingw32-make on Windows.

9. (Q): 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.

10. (S): The microscopic model simulates 10 minutes in one experiment. With the command `./collisionmodel 100` it averages over 100 experiments. Additionally, one separate single experiment also is recorded each time (`/data/microscopic.txt`). Analyze this (single run) 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?

11. (S): You can plot the histogram of the average collision duration (with 100 runs) 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.

***12.* (I):** 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?

4 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

5 Appendix

Running simulations in Q6 on Windows

(1) comment/uncomment relevant code around line 175-179 in wrap_around_big.c
(2) In wrap_around_big.c and encountering_probabilities.c, files are written to the folder /tmp/. Replace it with ../../data
(3) run the simulation 10 times manually.