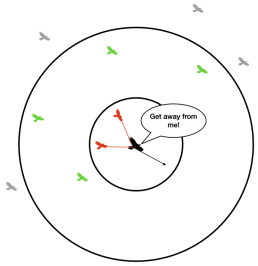
Project - Simulation Phase

Controlling a Swarm of Robots in a Simulator Stage using Reynolds Rules

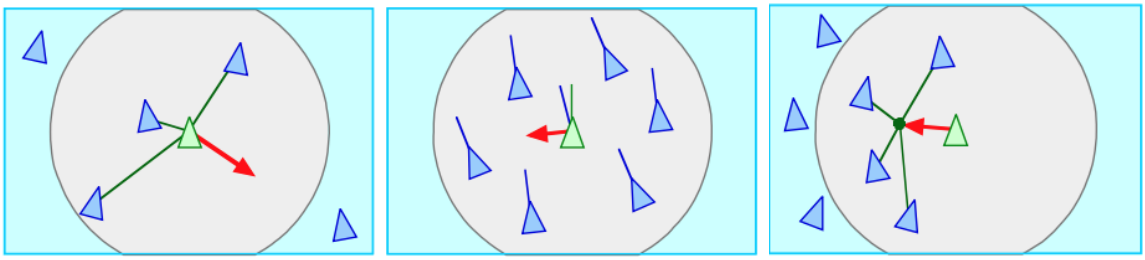


The goal of the project is to implement algorithms based on Reynolds rules to control robots in simulation. The first part of the project includes the implementation of basic Reynolds rules, while the second part requires the development of additional Reynolds rules. The algorithms will be tested in a simple 2D simulator Stage in Robot Operation System (ROS).

Introduction to Reynolds Rules

In 1987 Craig Reynolds introduced a set of behavioral rules aiming to simulate the movement of groups of animals, such as flocks of birds or schools of fishes. He created a computer program in which entities called boids move following these rules. To best represent the behavior of real animals, Reynolds didn't create any central control but instead programmed each boid to sense its own environment and decide where to move. This resulted in a fluid movement very similar to real bird flocks. The flocking rules are represented in Figure 1 and can be described as the following:

1. **Separation:** boids try to move away from nearby flock mates to avoid collisions.
2. **Alignment:** boids try to match their velocity and heading with other boids.
3. **Cohesion:** boids try to move closer to the other boids to form a flock.



(a) (b) (c)

Figure 1: Reynolds flocking rules. (a) Separation. (b) Alignment. (c) Cohesion.

These three rules alone are sufficient to make the robots group and move randomly together.

Each boid has direct access to the whole scene's geometric description but flocking requires that it reacts only to flockmates within a certain small neighborhood around itself. The neighborhood is characterized by a distance (measured from the center of the boid) and an angle, measured from the boid's direction of flight, as shown in Figure 2. Flockmates outside this local neighborhood are ignored. The neighborhood could be considered a model of limited perception but it is probably more correct to think of it as defining the region in which flockmates influence a boid's steering.

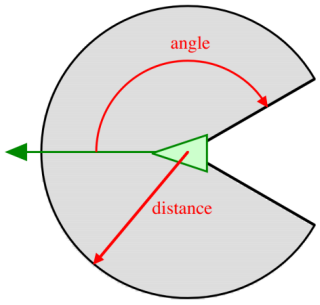


Figure 2: A boid's neighborhood.

In addition to these three simple rules, there are other Reynolds rules to implement, depending on the type of behavior the swarm needs to realize. Your task here is to implement two additional rules for:

1. **Navigation to a certain point**, where the whole swarm needs to reach a given point in the environment. Swarm should slow down as it is coming closer to the goal position, and finally stop at it.

2. **Obstacle avoidance**, where swarm or any member of it should avoid obstacles in the environment. It should become active only when the boid is in a certain range of the obstacle and increases as the swarm is closer to the obstacle.

In this project, the implemented Reynolds rules will be tested in the 2D simulator Stage.

A Simple Robot Model in 2D Simulator Stage

The robots you will be using are simulated in Stage (Figure 3) as simple omni-directional points (actually squares, but the shape is not important) without mass. This means that they have no inertia, i.e., they can achieve the commanded velocity instantly.

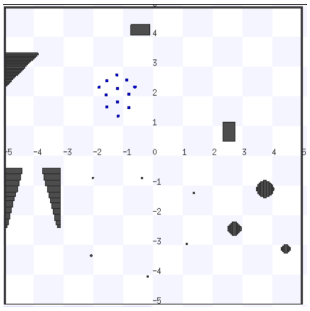


Figure 3: An example of an environment in Stage with 12 robots.

Desired velocities are commanded by sending a [$geometry\_msgs/Twist$] message on $robot\_x/cmd\_vel$ topic where x is the id of the robot in simulation (0, 1, 2, ...). At the same time, the simulator is continuously streaming positions and velocities of the robots on topic $robot\_x/odom$ of type [$nav\_msgs/Odometry$].

Various maps are available online, along with instructions on how to change and use the map you want. Depending on which rule you are testing, you will use the appropriate map for it.

There are also parameters you can change to test your system. The parameters should be changed before each simulation run so you can see how different parameters affect the final results. For example, how the boids behave when the minimum distance between the agents is changed (increased/decreased), what is the result when the distance at which the obstacle avoidance rule is active changes or when different importance (weights) is given to certain rules, etc.

Assignment

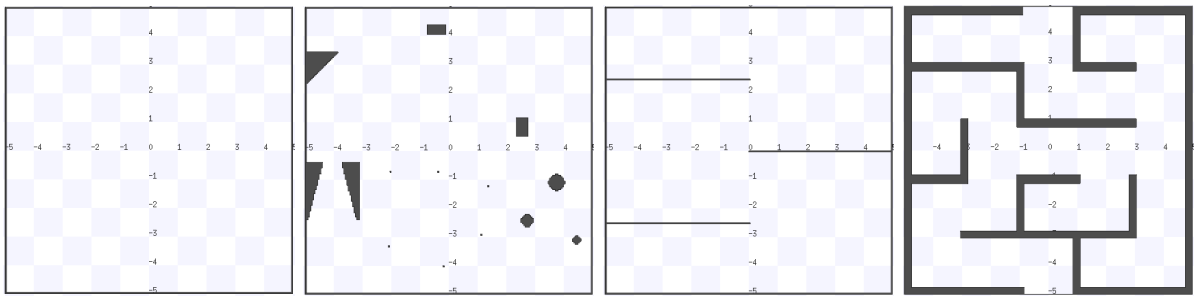
Your system should consist of at least 10 robots.[1] Y We encourage you to play around and analyze how the system behaves with different numbers of robots. Different environment sizes are available in Stage simulator (3x3, 5x5, 10x10). We recommend you use 10x10 so that the swarm has more space to move around. There are 4 maps available for you to test algorithms on as you create and add rules (Fig. 4):

a) a map with a frame (useful to keep boids inside certain space),

b) a map with different types of obstacles (useful to test obstacle avoidance rules),

c) a simple maze map (useful to test the combination of obstacle avoidance and navigation rules),

d) a hard maze map (useful to test the combination of obstacle avoidance and navigation rules).



(a) (b) (c) (d)

Figure 4: Maps to test algorithms on.

Your assignments are:

1. Implement basic Reynolds rules:

   a) separation,

   b) alignment,

   c) cohesion.

   Test your implementation in the Stage simulator in the map with a frame.

[1] You can start with fewer while developing your solution, but make sure to include results with at least 10 in your report.

2. Implement additional Reynolds rules for:

   a) navigation to a given point,

   b) obstacle avoidance.

   Test your solution in the Stage simulator in all given maps. Note that you can create new maps without obstacles and with different types of obstacles (narrow passages, wide obstacles in the way...) and additionally test your algorithms (from both the first and the second assignment) in them. Tests on the given maps are **mandatory**.

Please note that the first task is quite straightforward, but for the second part, you could be more resourceful and implement new and interesting algorithms.

Project report

A project group work will be evaluated by a project report. Each group should submit a report in the form of a .pdf file, tentatively including the following items:

1. Description of the system.
2. Approach and implementation of each task.
3. Figures from the simulation to complete the description and implementation.
4. Change at least 3 parameters and give a detailed analysis of how a specific parameter influences the system.
5. Link to video of the simulation implementation.
6. Key parts of code with description.
7. Conclusion.

A project presentation will take place after the report is submitted. During the project presentation, students will go through the submitted report and orally present their work and analysis. Furthermore, students should be able to run their simulations live during their presentations. All members of the group are expected to actively participate in the preparation of the project, report, and presentation.

The report and the presentation can be in English or Croatian.