Intelligent Robotic Systems
-
Fifth Lab Activity

Simone Ceredi

10/05/2024

Solution implemented

Control architecture

The adopted control architecture is a probabilistic finite state automaton with only two states: *stopped* and *wandering*.

- When in the *stopped* state the robot remains still and sends, using the range and bearing system, to all the neighbouring robots the value 1. It has a P_w probability to change state into *wandering*.
- When in the *wandering* state the robot wanders around the arena, going straight and avoiding obstacles, thanks to a motor schemas sub architecture; it also sends to all neighbouring robots the value 0. It has a P_s probability to change state into *stopped*.

Probabilities

Regarding the current robot, the P_w and P_s probabilities are calculated based on the number of robots in a stopped state around it. Given N equal to the number of standing robots around:

- $P_w = max\{P_w^{min}, W - \beta N\}$
- $P_s = min\{P_s^{max}, S - \alpha N\}$

The selected values for the six parameters are:

- $P_s^{max} = 0.99$
- $S = 0.01$
- $\alpha = 0.1$
- $P_w^{min} = 0.005$
- $W = 0.1$
- $\beta = 0.05$

Evaluation

Running the experiment a few times shows that the first clusters of robots appear after 300-500 ticks, but they are easily disrupted as they contain only a few robots. After 1000-1500 ticks the amount of clusters is usually reduced to two or three main ones. The swarm has a tendency to aggregate into a single one, but this can take anywhere from 2000 to 10000 ticks. One of the biggest improvements in the time the swarm takes to aggregate was switching the implementation of the *wandering* state. Initially, the implementation consisted of a random walk without collision avoidance. After running the experiment a few times, it seemed that the swarm would never aggregate into a single cluster. The main reason was that the robots took a long time to travel across the arena since the random movement often did not allow them to move at full speed. Switching to a *motor schemas* implementation for the *wandering* state showed good results, as the faster movement allowed the wandering robots to more easily find a cluster to join.

Problems faced

The biggest issue faced regards the optimization of the six parameters described above. While these values were given during the description of the problem, if they had to be found by trial and error, I think this task would have taken more than a few hours. In the beginning, the robots didn't behave as expected, but instead of attributing the error to an implementation bug, my first approach was to adjust the parameters, without resolving anything. The problem was that I had inverted the communication to the nearby robots, telling them that a robot was wandering when it was actually stopped, and vice versa. Debugging the control architecture was tough as it wasn't easy to understand if errors were caused by logic bugs or if the parameters were inappropriate for the task.
Another problem had to do with the robustness of the parameters. Changing the number of robots or the configuration of the arena led to significantly different performance. Doubling the length and width of the arena resulted in clusters containing up to five robots, and these clusters only lasted a few hundred ticks.