Persistent coverage

1. GOAL

The different aspects involved in the persistent coverage of an area by a group of mobile robots are the basis of this session. The practical goal is to implement a 2D simulator of the robots performing the persistent coverage.

2. METHODOLOGY AND SETUP

Before starting the exercise, read carefully the complete instructions of the practice. The practical session consists of programming from scratch the persistent coverage algorithm by following different steps. Python is recommended, but any other programming language can be used.

3. PERSISTENT COVERAGE

The next steps propose the order to program the main components of the persistent coverage algorithm:

3.1. Environment. Define a discrete 2D environment where each point coordinates features a coverage level that degrades with time:

$$\frac{\partial \Lambda}{\partial t} = A \cdot \Lambda + B \cdot \alpha \quad \text{(equation 1)}$$

With A<0. At this moment, there are still no agents, so alpha will be null.

This is a continuous model and you need to discretize the equation with sample time T. In particular the discrete algorithm of the model in pseudo code yields:

```
For each T do:
    Lamda(k+1) = F * Lamda(k) + G * alpha
End for
```

Where F = exp(A * T), and G = (B/A) * (exp(A*T) - 1).
Show in a figure the evolution of each point of the map coverage. Plot the quadratic coverage error of the domain over time.

3.2. A first agent. Introduce one agent in the environment with integrator dynamics ($\dot{p}_i = u$) performing a coverage action (α) and update the model of the coverage formulation including the action (equation 1).

3.3. Static path planning. Define a path offline for covering the full area. The agent will periodically follow this path to perform the persistent coverage.

3.4. Study the behavior of the model by modifying the different parameters of the problem: The state gain A, the input gain B, the coverage action parameters (coverage radius, action power level), or the agent speed.

4. OPTIONAL TASKS

Task C1: Local motion control: Compute the direction of motion of the agent by using the gradient of the variation of the coverage error with respect to the agent's position.

Task C2: Define a set of agents in the environment to perform the coverage.

Task C3: A gradient based motion control can get stuck in local minima. Implement a global motion control to reach global objectives and combine this with the local motion control.

Task C4: Implement the function over the domain that gives the priority to cover each point of the domain. Define zones of the domain with different levels of coverage priority and study the performance of the coverage error evolution

Task C5: Modify the coverage action to control the action power level of each agent. Define the control of the action power level as the weighted error of the agent's domain.

5. SESSION REPORT

As a result of this practical session, the final developed code will be submitted through the ADD (https://moodle.unizar.es/add/).