Multiagents search of lost targets in a static 2D world using range sensors

Practical Assignment 2

Denis Karakoç, s4835093
Piotr Leszmann, s4771826
Pelle Kools, s1010033

1. Introduction

Our task was to implement and argue about the choices made considering the following agents' behaviour:

a) A Greedy Agent
b) A Greedy Multiagent; lacking common belief, thus lacking communication, 1 step discrete.
c) A Greedy Multiagent; sharing common belief, thus lacking communication.
d) Multiagent; search n-step continuous optimization (centralized)

2. A Greedy Agent

a) Environment:
For this plot, we varied the figure from the Python & Numpy Basics Notebook.

b) Sensors:
We implemented a function for the probability of non-detection that looks only 1 step ahead, by directly translating the formula of $p(\neg D)$ into code. For plotting, we stored the probability of non-detection for every possible location in the environment given a starting location of $x = (20, 20)$ (center)

c) Agents:
To simplify movement, we defined every possible movement beforehand in 'mat'. The rest of the implementation of the feasible states follows the guidelines in the comments.

d) Initial Belief:

We defined the bivariate gaussian function by translating the mathematical equation[1]into code. Similarly, as for the sensors, we looped over every location in the environment, stored the belief values and normalized them before plotting. We plot both a 3D rendering, as well as a 2D projection.

e) **Utility Function:**
The utility function is a direct translation of the mathematical equation. The next best state calculates the utility of each feasible state, given a current location and returns the feasible state with the lowest cost.

f) **Greedy Algorithm:**
In essence, in a while loop, the agent determines the next best state as defined previously, moves there and updates its beliefs by multiplying the prior belief with sensor information (and normalizes the belief). This continues for 'nite' cycles. The plot shows the agent's trajectory and current position (implemented by storing each visited state in a list) and also depicts the agent's current beliefs as a 2D map.

3. Multiple Greedy Agents with their own beliefs

This is mostly a reimplementation of problem 2. We migrate most features like the utility, belief, feasible states and probability of non-detection functions into an agent class, which can be used to create multiple agent objects with their own different beliefs, utilities, etc. We also define an environment class that defines the size of the search space. This object is shared by all agents.

The search implementation is similar to the previous problem: In a while loop, two agents take turns moving to their next best states and updating their own beliefs. The final plot depicts both agents' locations and trajectories in their shared environment, with agent 1's belief on the bottom and agent 2's beliefs on top of the graph.

4. Multiple Greedy Agents sharing a common belief

This problem is the same as 3, except that the agents share a common belief. We define a global variable called belief and change each agent's update_belief function to take this global belief as a parameter. In a while loop, both agents consecutively move to the next best state and update the global belief parameter. During the plotting, only the global belief is depicted. To achieve this, the plotting for the belief had to also be taken out of the agent class and put into the while loop.

[1] Multivariate Gaussian

5. Three Continuous Agents sharing a common belief looking N-Steps Ahead

This task is by far the most complex computationally, due to how many nested loops are required. We defined the continuous agent class and reused most functions from the discrete agent, such as the probability of non-detection, bivariate gaussian,

The utility function is different: since we now look N steps ahead, we need to simulate movement. To achieve this, we make copies of the agents and the common belief, which we move and change to determine the utility of a set of movements. The function is a direct translation of the mathematical equation. The movements are defined by turn angles only, which are used by the agent to transition states: an agent adjusts the direction its heading to using the turn angle and moves toward it for a set distance, determined by its velocity.

We use a supplied optimizer object to determine the best turn angles for movements, calculated by minimizing the utility of different possible turn angles. Next, in a while loop, the best turn angles are determined for each agent looking $N = 2$ steps ahead, after which each agent makes its move and updates the shared belief after using their sensors in their new states. The plotting is handled as in the previous problem.

Division of Labor

We all worked together on the entire assignment, with Pelle screensharing and Piotr and Denis back-seat coding[2]. We considered this as the best way of work as this allowed us for instant feedback and improvements towards the best solution.

[2] A back-seat coder is the programmer's equivalent of a back-seat driver. Back-seat driver