AERO 489/689 Foundations of Aerospace Autonomy

Fall 2023

Programming Assignment 2: Mars Rover – Rule-Based Agent

Context: Consider an autonomous rover on the surface of Mars. The goal of the rover is to explore the terrain around it, find and retrieve a biological sample, and to return it to the starting position. To simplify things, we will assume the Mars rover is traversing a 6x6 grid. The agent's state is specified by its location (x,y) and its orientation (left, right, up, down).

There are certain hazards in the environment:

- Rocks (which destroy the rover on contact (they're very sharp))
- Storms (which destroy the rover on contact (they're very powerful), but also degrade the air quality in adjacent (not diagonally adjacent) cells)
- *Grad students only*: Sandy areas (which increase movement cost from 1 to 2)

The agent can take only certain actions:

- Move forward
- Rotate clockwise/counterclockwise by 90deg (CW/CCW)
- Drill (to obtain the biological sample in the correct cell)
- Spectrometer (to determine if a biological sample is present in the current cell)
- Air quality (to determine if the air quality is degraded in a cell)
- Traction (to determine if a cell is sandy)
- Lidar (to determine if rocks are ahead and if so, how far)

Much of the agent's code was coded by a previous intern at Marscorp (see attached Python files). However, some of the rules of the rule-based system are not implemented yet, and some of the functions need to be written to query the knowledge base.

All student tasks:

1. Complete the check_safe_unvisited function.
2. Complete the ask_cell_bio function.
3. Complete the rotate_cw_rule and rotate_ccw_rule.
4. Complete the air_quality_rule.
5. Complete the biosample_meas_rule.
6. Complete the cell_safe_rule.
7. Augment the existing hybrid agent to check for unvisited but safe spots to visit.
8. Compute the number of trials (out of 100) where the rover succeeded in its goal.

Graduate student tasks:

9. Implement logic to avoid sandy squares when planning actions. This includes:
   a. Implementing a modified path_cost.

    b. Implementing A-star or greedy best-first search.

10. Compute the difference in total movement cost when accounting for sand and when not accounting for sand, averaged over 100 trials.

Deliverables:

Please upload your deliverables as a single zip file containing:

- A pdf with the responses to each of the tasks 1-8 (1-10 for grad students), including snippets of code and graphs as requested in the tasks.
- All the source files (e.g., .py) and inputs required to run the code. The code must run with only the files as provided.