CZECH TECHNICAL UNIVERSITY IN PRAGUE

ARTIFICIAL INTELLIGENCE IN ROBOTICS

LUKÁŠ MÁLEK – maleklu6@fel.cvut.cz

FAKULTA ELEKTROTECHNICKÁ
6.2.2023

PROCEDURE FOR INITIATING SIMULATION:

1. Launch CoppeliaSim.
2. Access the parent folder.
3. Execute the Python script, "Explorer.py", by entering "python3 Explorer.py" into the terminal

DESIRED BEHAVIOR:

The script will initiate the simulation within CoppeliaSim and terminate it when the entire map has been explored. This will be indicated by the absence of any detected frontiers for ten consecutive iterations. To choose the preferred planning method among the options of p1, p2, or p3, one may edit the variable "planning" found at the beginning of the "Explorer.py" file.

Frontiers will only be searched for once the previously designated goal has been reached or the path has been recalculated due to newly detected obstacles. This approach has been taken to enhance the response time on slower computers operating within the Windows Subsystem for Linux (WSL) environment.

EDITED FILES:

- Explorer.py
- HexapodExplorer.py
- HexapodController.py
- HexapodRobot.py
- HexapodRobotConst.py

UTILIZED CODE:

- Week 1 - goto: employed in trajectory following - hexapod_controller
- Week 2 - goto_reactive: employed in trajectory following - hexapod_controller
- Week 3 - fuse_laser_scan: employed in mapping - hexapod_explorer
- Week 4 - grow_obstacles + plan_path + simplify: employed in planning - hexapod_explorer
- Week 5 - Dstar: not utilized
- Week 6 - frontier detection: used in planning-find_free_edge_frontiers – hexapod_explorer
- Week 8 – t2a-tspn: employed through the TSP_Solver library

ADDITIONAL IMPROVEMENTS:

- Improved plot visualization
- Debugging with timestamps displayed on screen
- Generated HTML documentation
- Added method to halt the robot during calculations
- Termination of simulation once the termination criteria have been satisfied
- Use of two separate maps for improved route prediction
- Added recordings for each subtask

NOTES:

- For simplicity, Dubins path has not been utilized and the robot turns with zero linear velocity
- Upon simplification of the path, the first coordinate is removed, allowing the robot to bypass the start position, which is equivalent to its odometry.
- In the plot representation:
  - Red lines indicate the path
  - Red crosses indicate points along the simplified path
  - A black cross represents the current frontier o Gray dots indicate the robot's position
  - Red dots indicate all discovered frontiers outside of obstacles.
- The obstacle growing method has been adjusted. Instead of growing both obstacles and unknown areas, only obstacles are grown.
- Three different maps are used in this assignment: n
  - o Gridmap: the default map from the laser scan
  - o Gridmap_processed: map used to assess the obstacle presence along the chosen path
  - Gridmap_astar:The map incorporates an extra margin of safety within the grow_obstacle() function. This is to prevent collisions between the robot and newly discovered obstacles during path following.
- If the robot encounters an obstacle, the goto command is bypassed and a new one is executed, which may result in unintended behavior.
- The p2 parameter has been slightly altered. The likelihood of discovering two frontiers with identical properties is low and has not been taken into consideration for the sake of code readability. The code is available for download on GitHub under the "Semester Project p2 Version 1" commit.

VIDEO DEMONSTRATIONS:

- Task p1: https://youtu.be/NX73DGnrlyM
- Task p2: https://youtu.be/ZbIah73r898
- Task p3 https://youtu.be/MRJW18ZNMPw

BUGS:

- If the robot is too close to an obstacle when the simulation begins, it may become stuck after the obstacle has grown and no path to any frontier is available.
- Occasionally, the script may take longer to initialize, leading to a longer waiting period for frontier and path discovery. No action is necessary; simply wait for approximately 4 seconds.
- When the robot turns close to obstacle and gett stuck into the grown obstacle, then it results in a bug where a_star does not find the correct route

POSSIBLE IMPROVEMENTS:

- Recalculation of frontiers on each iteration and selection of the optimal new path
- Optimization through the use of numpy arrays, loop reduction, vectorized operations, and the utilization of 1D arrays
- Creation of a third map to enhance route prediction, which may be merged with the growing obstacles map • Code Refactoring
- Focus on frontiers that are most accessible, such as those located in front of the robot, to avoid 180-degree turns
- Try JPS search instead of Astar
- Implement DTSP(N) and dubins vehicle for faster robot movements

IMPLEMENTED TASKS

TASKS	EXPECTED POINTS
Report	1
Code_style	2
M1	1
M2	2
F1	1
F2	2
F3	7
P1	1
P2	2
P3	4
A1 (WIP)	5?
Total	28