ME 401/5501 – Robotics and Unmanned Systems
HW #2: DUE September 11th, 2023

Anh Doan

Problem 1:

Create a function that calculates the distance from one **node** to another. Pass two nodes to the function and return the Euclidean distance. Test your function by having it calculate the distance from (2,1) to (3,2). Make sure the answer is correct.

Submit your Python code.

anh_doan_unmanned_systems/anh_doan_unmanned_systems/home-work_2/question_1.py at main · asd109a/anh_doan_unmanned_systems (github.com)

Problem 2:

Create a function that checks if the current **node** is valid based upon the list of obstacles, grid boundaries, and current location.

Using an obstacle list of (1,1), (4,4), (3,4), (5,0), (5,1), (0,7), (1,7), (2,7), and (3,7); and a bounding box of 0 to 10 for both x and y, and step size of 0.5, verify that the location (2,2) is valid. Assume the obstacles have a diameter of 0.5 (only occupy the node at which they reside).

Pass the obstacle list, node, and map boundaries/step size, and return a Boolean True/False depending on if the node location is valid (reachable).

Submit your Python code.

anh_doan_unmanned_systems/anh_doan_unmanned_systems/home-work_2/question_2.py at main · asd109a/anh_doan_unmanned_systems (github.com)

Problem 3:

See pseudocode on Canvas for significant assistance in the implementation of Dijkstra's Algorithm.

Integrate your functions into a script with a function that runs Dijkstra's algorithm given the starting location, goal location, grid information, and obstacle list. The function should end with a plot showing the grid space with obstacles (can just use markers for the obstacles) and the desired path

from the start location to the goal location. Your function should return this list of waypoints for the desired path. The waypoint list will be used for the Turtlebots later in the semester.

Use the grid information above (Problem 2) with a starting location of (0,0) and a goal location of (8, 9).

Show the plot of the grid and desired path.
Submit your Python code.

anh_doan_unmanned_systems/anh_doan_unmanned_systems/home-work_2/question_3.py at main · asd109a/anh_doan_unmanned_systems (github.com)

Problem 4:

Assuming the robot has an actual size, you will need to inflate the graph such that the algorithm does not plan such that it would contact the robot. Use a robot diameter of 1.0. This inflation can occur in several different methods including: 1) inflate the obstacle list and grid boundary, 2) when checking for a collision between a node and the grid boundary or obstacle, see if the distance to the obstacle is less than the robot diameter (rather than checking for distance = 0).

Rerun the same grid/configuration used for Problem 3 and show the results.

anh_doan_unmanned_systems/anh_doan_unmanned_systems/home-work_2/question_4.py at main · asd109a/anh_doan_unmanned_systems (github.com)