Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lecture: Introduction to Mobile Robotics
Summer term 2020

Prof. Dr. Wolfram Burgard
Daniel Büscher
Lukas Luft
Marina Kollmitz

mobilerobotics@informatik.uni-freiburg.de

Sheet 11

Topic: Path Planning

Due date: 30.07.2020

Global (Path-) Planning

Graph-search algorithms like Dijkstra or A^* can be used to plan paths in graphs from a start to a goal. If the cells of a grid map are represented as vertices of a graph with edges between the neighboring cells, graph-search algorithms can be used for robot path planning. For this exercise sheet we consider the 8-neighborhood of a cell $\langle x, y \rangle$, which is defined as the set of cells that are adjacent to $\langle x, y \rangle$ either horizontally, vertically or diagonally.

You can find an implementation of graph-based 2D path planning in the planning_framework tarball provided on the website. Complete the missing pieces following the instructions below.

Exercise 1: Dijkstra Algorithm

The Dijkstra algorithm can be used to calculate minimum cost paths in a graph. During search, it always chooses the vertex from the graph with the lowest cost from the start and adds its neighboring vertices to the search graph.

(a) Let $M(x, y)$ denote an occupancy grid map. During search, the grid cells are connected to their neighboring cells to construct the search graph. Complete the function `get_neighborhood` in the provided planning framework. The function takes the coordinates of a cell and returns a $n \times 2$ vector with the cell coordinates of its neighbors, considering the boundaries of the map.

(b) Formulate a function for the edge costs between two cells that allows for planning of the shortest collision free path on the grid. Include occupancy information in your edge cost function to prefer cells with low occupancy probability over cells with higher probability. Regard a cell as an obstacle if its occupancy probability exceeds a certain threshold. Which threshold would you choose? Implement this function in `get_edge_cost`.

(c) Implement the update step of the Dijkstra algorithm in `run_path_planning`. For the current parent node, consider all of its neighbors and calculate their *tentative* distances from the start location (cost) and their predecessor in the grid. Under which condition should the update be done? You are now ready to run the Dijkstra algorithm with `python planning_framework.py`.

Exercise 2: A^* Algorithm

The A^* algorithm employs a heuristic to perform an informed search with higher efficiency than the Dijkstra algorithm.

(a) What properties of the heuristic are required to ensure that A^* is optimal?

(b) Define a heuristic for optimal 2D mobile robot path planning. Complete the function `get_heuristic` in the planning framework. The function takes the coordinates of a cell and the goal and returns the estimated costs to the goal. You are now ready to run the A^* algorithm with `python planning_framework.py`.

(c) What happens if you inflate your heuristic by using h_2, which is a multiple of your defined heuristic h? Try different multiples: $h_2 = \{1, 2, 5, 10\} \cdot h$