Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lecture: Introduction to Mobile Robotics
Summer term 2016

Prof. Dr. Wolfram Burgard
Dr. Michael Ruhnke
mobilerobotics@informatik.uni-freiburg.de
Dr. Bastian Steder

Sheet 13

Topic: Path Planning

Due date: 22.07.2016

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.

In the planning_framework tarball, you find an implementation of graph-based 2D path planning. The `planning_framework.py` file contains a `main` function that implements the planning loop. You do not need to edit this function, but make sure you understand what it does. Use the provided empty functions for implementing the exercises on this sheet. Test your implementation by executing the `planning_framework.py` file.

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.

1. 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_neighborbood` 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.

2. Formulate a function for the edge costs between two cells that allows for planning of collision-free, shortest paths on the grid. Regard a cell as an obstacle if its occupancy probability exceeds a certain threshold. Which threshold would you choose? Implement the `get_edge_cost` function.

3. Include occupancy information in your cost function that prefers cells with low occupancy probability over cells with higher probability.

Exercise 2: A^* Algorithm

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

1. What properties of the heuristic are required to ensure that A^* is optimal?
2. 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.
3. What happens if you inflate your heuristic by using h_2, which is a multiple of your defined heuristic h? Try different multiples: $h_2 = \{1, 2, 5, 10\} \cdot h$