Prof. Dr. Maren Bennewitz
M.Sc. Stefan Oßwald
M.Sc. AbdElMoniem Bayoumi

Bonn, 22 June 2017

Humanoid Robots
Exercise Sheet 6 - A* Path Planning and Footstep Planning

Exercise 11 (10 points)

Implement the A* algorithm for planning a path on a 2D grid map. All the methods that need to be implemented are in the file `src/11_path_planning/src/PathPlanning.cpp`, you don't have to change any other files.

The code provides implementations for the *open list* and *closed list*. The open list is a priority queue where you can add grid nodes together with a cost value. The method `removeMin()` will then always return and remove the node with the lowest costs.

At the beginning of each method that you have to implement, you will find a list of classes and methods that you can use in your implementation.

a) Implement the cost function for moving from the current node to the next node in `getCosts`. Use the Euclidean distance metric.

b) Implement the straight line distance heuristic for estimating the distance to the goal node in `StraightLineDistanceHeuristic::heuristic`.

c) Implement the Manhattan distance heuristic in `ManhattanDistanceHeuristic::heuristic`.

d) Implement the method `getNeighborNodes`. For the cell given as an argument, this method should return a vector of all nodes in the neighborhood (up, down, left, right, and diagonally). It should only return the cells that are reachable for the robot, i.e., cells that are within the map boundaries and that are not occupied by an obstacle.

e) Implement the method `expandNode`. This method should expand the current node and add new nodes to the *open list*. It should also

- calculate the costs for each neighbor,
- set the predecessor of the neighbor nodes (that will be used later to extract the path),
- update the cost and heuristic values of the neighbors that are already in the open list.

f) Implement the method `planPath`. It should

- add the start node to the open list,
- process the open list in a loop by removing the node with minimum costs,
- expand that node and put it onto the *closed list*,
- check if the goal has been reached using `isCloseToGoal()`,
- call `followPath` for extracting the final path once the goal has been reached.

g) Implement the method `followPath` for extracting the final path by following the predecessors from the current node back to the start node and returning them in the correct order.

If you have Gnuplot installed on your computer, then you can get run `plot.gp` in the `scripts` directory to show an animation of the node expansion and the final path found by your implementation.

Exercise 12 (10 points)

Humanoid robots need to plan foot steps for getting from one location to another. In this exercise, we will extend the A* algorithm from the previous exercise for planning foot steps.

All the methods that need to be implemented are in the file `src/FootstepPlanning.cpp`, you don't have to change any other files.

a) Implement the cost function for moving from the current footstep to the next footstep in `getCosts()` according to the cost function defined on slide 12.

b) Implement the Euclidean distance heuristic for the foot steps in `heuristic()`.

c) Calculate the new coordinates (x', y', θ') that the foot moves to when the robot executes the given footstep action $\Delta x, \Delta y, \Delta\theta$ when it is currently in (x, y, θ) in `executeFootstep`.

d) Implement the method `getNeighborNodes()` that should return the neighbor foot steps that are reachable from the current foot step. The possible foot step actions are given as a parameter to the method. You can use the method `executeFootstep` defined above, and the method `bool isColliding(footstepNode)` that is already provided by the code.

If you have Gnuplot installed, you can display an animation of the footstep sequence by running `plot.gp` in the `scripts` directory.

Deadline: 29 June 2017, 11:59 am