Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Technical Aspects of Multimodal Systems
Department of Informatics
Y. Jonetzko, S. Li

Robot Practical Course

Assignment #4

Due: 26.06.2020, 13.00

This assignment should demonstrate the capabilities and difficulties of *Probabilistic Road Map* path planning algorithms. Remember to update the repository files with `git pull`.
For this assignment you need the additional package *networkx*, install it with `sudo apt install python-networkx`.

Task 4.1 Launching Example: After updating the repository, launch the new task setup with `roslaunch itr_rpc task_4.launch`. You will see a map with a maze and a small magenta robot. The robot can move on the map but only within the white free space. Thus, it will try to achieve a start position in free space, if launched on occupied space. Before launching the example script with `rosrun itr_rpc dummy_prm.py`, read what is supposed to happen in order to understand the visualization:

- The script moves the robot from the start position $(1, 0)$ to the goal position $(-1, -1)$.
- To achieve this, an intermediate point at $(1, -1)$ is required.
- The point is checked for collision (green means free, red means collision).
- The connections between the points are checked for collision (same colors as points).
- Colliding lines and points will vanish after some time.
- The lines are fed into a graph.
- The shortest path is specified and highlighted in blue on the map.
- The path is executed.

There are many sleep commands in the code to make it easy to follow. Remove the commands after you understood the process. They are commented with a `FIXME` tag.

Task 4.2 Escape: Escape the maze using a *Probabilistic Road Map*. Your extraction point is $(2.5, -4.5)$. You should find useful hints at the bottom of this sheet. Copy the example to your `script.py`.

4.2.1 *Bonus*: Increase the difficulty and find a path in more complex mazes within 2 minutes.

difficulty	start position	goal position
`easy`	$(1, 0)$	$(2.5, -4.5)$
`medium`	$(0, -1)$	$(2.5, -4.5)$
`hard`	$(0.25, -0.25)$	$(2.5, -4.9)$
`honor_student`	$(0.25, -0.25)$	$(2.5, -4.9)$

Hints:

Structure There is a predefined structure. Use it.
Points and lines There are classes for points and lines. Use them.
Solution drawing There is a function which draws your calculated solution in blue. Use it.
Path interpolating There is a function to interpolate the path between two points on a linear line. Use it!
Random The `random` library is very useful for doing things with random samples.
Permutation There are permutation algorithms available. You might want to use `itertools`.
Graph library There are graph libraries in python (networkx).

Difficulty Set difficulty with `roslaunch itr_rpc task_4.launch difficulty:=medium`. Available difficulties are: `easy`, `medium`, `hard`, `honor_student`

Collision paths Turn off the drawing of colliding lines with `roslaunch itr_rpc task_4.launch lines:=false`. This will probably speed up line collision checking.