Exploration Homework

Autonomous Robotics Labs

Labs 04 (2.3./5.3. 2020)

Basic description

Your task will be to implement the algorithms for path planning and exploration shown at the labs. You will be given nodes that will publish an occupancy grid and tf containing the robot position in relation to the grid. Supplementary nodes will be provided to visualize the occupancy grid and test your nodes.

Specifically, you will need to implement theses nodes:

planner.py

- provides service for planning from robot position to a goal
  - path planning done using a graph-based search algorithm (recommended A* or Dijkstra, but can be any graph-based algorithm you like)

frontier.py

- provides services for frontier based exploration
  - check if any frontiers exist
  - return random frontier
  - return closest frontier

Instructions

To start working on the homework, first download the template source code either from the webpage or go to your workspace and:

```
curl -O https://cw.fel.cvut.cz/wiki/_media/courses/aro/tutorials/hw_lab_04_src.zip
unzip hw_lab_04_src.zip
```

The zip-file contains a “src” directory. In that directory, there are two packages: **dummy_grid** and **exploration**. The code templates that you should finish are in the *exploration* package in the **scripts** folder (i.e. you will find the *planner.py* and *frontier.py* scripts). There are comments inside these scripts where you should most likely put your own code. Feel free to modify the code as you like. The structure and comments are there just to get you started. You can also put additional helper code into the *exploration/src/exploration* folder. You can then load these modules in your main scripts using:

```
from exploration import your_module
```

Don’t forget to properly initialize your workspace, build the packages and source the configuration file.

Dummy_grid

To test your codes, you can use scripts and a launch file from the *dummy_grid* package. Here is a list of the included files:

grid_publisher.py Publishes the “dummy” occupancy grid (loaded from a numpy 2D array). You can change the origin position (in meters) and rotation (in radians) by specifying these private parameters: **x**, **y**, **theta**. E.g.:

```
rosrun dummy_grid grid_publisher.py _x:=1.0 _y:=-0.5 _theta:=0.5
```

grid_visualizer.py Listens for grid and plots it using the matplotlib library.

pose_sender.py Publishes tf2 pose of the robot. You can change the robot position by changing the private parameters **x** and **y**. Be careful that the robot’s position land on the grid and on an empty cell. Starting this node will also open a window where you can change the robot’s position in real time (if it does not work for you, just specify it via the command line and leave the window as it is). Example command:

```
rosrun dummy_grid pose_sender.py _x:=1.0 _y:=2.5
```

compute_front.py Tests the frontier exploration part of the homework. You can call it with private parameter **cmd** that can have either value “*any*” (calls “any_frontiers_left”), “*random*” (returns a random frontier), “*near*” (returns the closes frontier)

test_planner.py Tests the path planning part of the homework. You can specify the goal position by providing the **x** and **y** private parameters (in meters).

To speed up the testing pipeline, you can launch the provided launch file “**dummy_grid.launch**”. Running it will start the grid_publisher and pose_sender and it will also start *rviz* with some basic configuration. There, if the dummy grid is being published, you should see the grid itself. You should also see small cubes marking a path if you run the test_planner script (and your path planning works). And small green cubes should be displayed at the locations of the frontiers when calling the compute_frontier script.

The syntax for launching the dummy_grid.launch file is:

```
roslaunch dummy_grid dummy_grid.launch ox:=-0.5 oy:=-0.2 theta:=0.1 rx:=2.6 ry:=1
```

Where **ox**, **oy**, and **theta** are the x and y coordinates and rotation (around the z-axis) of the origin. The parameters **rx** and **ry** are used to specify the robot's position in the real-world (relative to the map frame).

Troubleshooting

Some error involving *matplotlib* You probably need to install/update the matplotlib package. The newest version (3+) however does not work with Python 2.7. Therefore, you need to specify lower version, i.e. "`pip install -U matplotlib==2.1`"

Assignment deadline: **9.3./12.3.** – the homework will be personally checked and evaluated at the corresponding Labs.