Introduction to Robotics – Exercise 4

Date of submission – 28/01/20 (23:55:00)

Description

Robotic coverage is a fundamental problem in robotics, where one or more robots are required to visit every part of a given area using the most efficient path possible. Coverage has many applications in a multitude of domains, including search and rescue, mapping, demining and surveillance.
In this project you will implement the STC (Spanning-Tree Coverage) algorithm for a single-robot coverage using ROS.
You will be using the TurtleBot simulator in Gazebo to run and test the algorithm.

Assumptions

You can assume the following about the environment and the robot's movement:

- Offline coverage – the world's map is given in advance.
- Approximate cellular decomposition – the environment can be decomposed into a collection of uniform grid cells, each cell of size D.
- The robot can move only in four compass directions - East, West, South and North.
- The robot has perfect localization.
- The given area will be covered by the robot itself and not by an external coverage tool.
- The robot has a circular shape, whose diameter is equal to the cell size D.

Assignment

Input: The algorithm is given the map of the environment as a .pgm file. For testing, you can use the map playground.pgm from turtlebot_gazebo/maps.

Algorithm: The robot should visit all the cells that are accessible from its initial position.
The main steps of the algorithm are:

1. Load the occupancy grid map by calling the static_map service.
2. Switch to a grid, where each cell is equal to the size of the robot D. To avoid collisions of the robot with obstacles, if one of the subcells in a given grid's cell is occupied then mark the entire cell as occupied.
3. Use an appropriate TF listener to find the robot's initial position in the map.
4. Find all the free cells that are reachable from the robot's initial position and ignore all the other cells.

5. Switch to a coarse grid, each cell of size 4D. You can assume that if one of the subcells of a given 4D cell is occupied then the entire cell is occupied.
6. Build a spanning tree on the coarse grid using DFS.
7. From the spanning tree create a Hamiltonian cycle that visits all the free cells in the fine grid.
8. Send the appropriate velocity commands to the robot that will make it walk along the spanning tree until the robot returns to its initial position

Output: Your application should generate one output file:

1. new_grid.txt - The new grid with D sized cells.
2. Coverage_path.txt – the path (indexes) the robot did on **the D-grid.**

*****Do NOT hand in any of these files! Your application should generate it!!!!**

Instructions:

1. Make a new ROS package called coverage, with the appropriate dependencies.
2. Fill in your name(s) and ID(s) in the package manifest file.
3. Copy maps and world files into the appropriate sub-directories of your package.
4. Your task is now to write the coverage node as explained above.
5. You can use this launch as the base of your launch file, don't forget to add your node:

```
<launch>
<param name="/use_sim_time" value="true"/>
<!--  Launch turtle bot world  -->
<include file="$(find turtlebot_gazebo)/launch/turtlebot_world.launch"/>
<!--  Run the map server  -->
<arg name="map_file" default="$(find coverage)/maps/playground.pgm"/>
<node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file) 0.05"/>
<!--
 Publish a static transformation between /map and /odom
-->
<node name="tf" pkg="tf" type="static_transform_publisher" args="6.9 5.9 0.0 0.0 0.0 0.0
/map /odom 100"/>
<!--  Launch coverage node  -->
<node name="coverage_node" pkg="coverage" type="coverage_node" output="screen"/>
</launch>
```

Rules

1. You can write your code in python2.7 or python3.
2. Make sure that your code is tidy and well-commented.
3. You should do this lab on your own or in pairs.
4. If you use any external sources of inspiration, other than ros.org, then let us know in a README file.
5. Specify which ROS distribution you implemented the code in in your README file.

What to Hand In

1. You should **zip** your package and submit it.
2. In the package.xml you should have the line
   <author> Jane Doe 111111111 <\author>
   (pairs : <author> name1 ID1 name2 ID2<\author>)
3. **Add readme file with your names and IDs.**
4. You should not hand in executable files, or any other files that can be regenerated.
5. Your code should be easy to run. After getting a copy of your code, running:
   catkin_make --pkg coverage
   roslaunch coverage coverage.launch
   should be enough to start up Gazebo and make the robot move!
6. You should also hand in an example `coverage_stat` file from one of your tests.

Good luck!