**PS6**

**Mapping with Known Poses**

***Out: Tuesday, October 24***

***Due: Tuesday, October 31***

Given perfect knowledge of a robot’s pose in a hallway (*x, y, theta*), build a map of its surroundings, using the log odds ration derivation.

**Reading**

The following material is relevant to this problem set:

* 9.1 Introduction to Occupancy Grid Mapping
* 9.2 The Occupancy Grid Mapping Algorithm
* Burgard et al. slides, <http://ais.informatik.uni-freiburg.de/teaching/ss14/robotics/slides/08-occupancy-mapping-mapping-with-known-poses.pdf>
* Stachniss lecture, <https://www.youtube.com/watch?v=Cj91xll94U4>

**Files Needed**

Use the uml\_mcl code to provide a launch file with the hallway world and the */robot/base\_scan* and */robot/cmd\_vel* topics.

You must use the two topics to move around and sense the world. However, you may also subscribe to the */stage/base\_pose\_ground\_truth* topic to learn where you really are. So, you'll still be using the sloppy motors to move, and the bad laser to sense the world, but you have your “known pose.” You may not use the ray-tracing code as part of the solution.

$ git clone <https://github.com/DeepBlue14/mcl_ws.git>

After cloning, do the following:

$ cd mcl\_ws

$ rm –r devel build

$ catkin\_make

$ source devel/setup.bash

$ cd src/no\_weights/src/raycaster

$ make clean

$ make

$ cd ../../../with\_weights/src/raycaster

$ make clean

$ make

You should now be able to:

$ roslaunch uml\_mcl mcl.launch

Followed by:

$ rosrun no\_weights no\_weights.py

Or:

$ rosrun with\_weights with\_weights.py

**Setup**

***\*\*\*If you are NOT using VLabs\*\*\****

These things might be needed:

$ sudo apt-get install libcv-dev

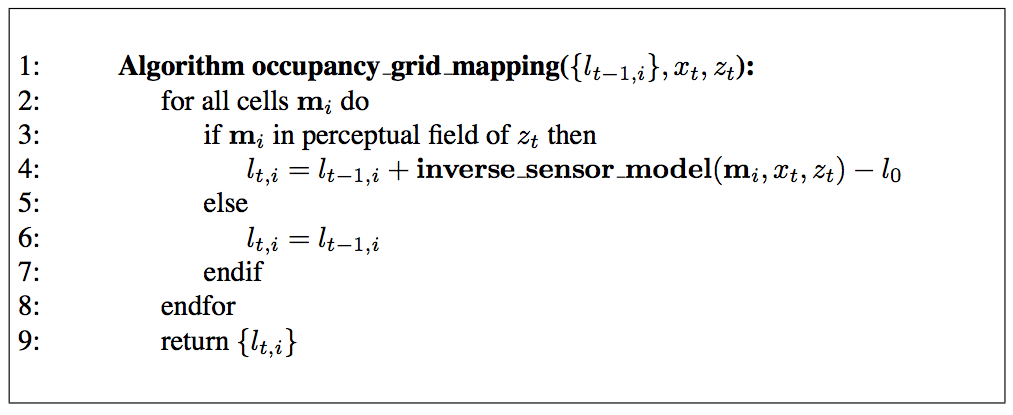
$ sudo apt-get install libcvaux-dev libhighgui-dev

$ sudo apt-get install python-numpy python-opengl

$ sudo apt-get install swig

**The Assignment**

Implement the log odds ratio algorithm as described in the text and lecture:



Build a map of the world where each cell is represented by the probability of its being occupied.

Drive the robot around (probably with a simple wall-avoidance reactive algorithm, similar to the one you implemented in PS3) and have the map accumulate knowledge.

Implement some fashion of displaying the map, most likely as an image with white to black pixel where white is *p(occupancy)* = 0.0, and black is *p(occupancy)* = 1.0.

**To Turn In**

Turn in the following:

1. Your code to implement the “mapping with known poses” algorithm. Submit your entire ROS package.
2. A 300 to 500 word discussion about what was interesting about the assignment. Include as part of the discussion:

* Challenges you encountered (and how you solved them)
* Representation of the world map
* Performance of your implementation
* Code excerpt accomplishing the probabilistic updates of the map

***This write-up file should contain your name and email address.***

**Submit**

If you are a CS major, submit material you wish to deliver electronically via:

$ submit jkuczyns ps6 zip-file

Remember that the “submit” command only works on the cs and mercury servers. Use “scp” to transport your code from VLabs to cs.uml.edu.

If you are NOT a CS major, email your zip file to [jkuczyns@sc.uml.edu](mailto:jkuczyns@sc.uml.edu).

*This assignment was jointly developed by Fred Martin, James Dalphond, and Nat Tuck. Thanks to Eric McCann for updating it for 2015, and James Kuczynski for updating it for 2017.*