PS6

3D Particle Filter Localization

Build a particle filter to localize a robot's (x, y, theta) pose in a hallway, given a map.

Reading

The following material is relevant to this problem set:

* 6.1, Introduction to robot perception
* 6.2, Maps (feature-based vs. occupancy grid)
* 6.3, Beam models of ranger finders
* 6.7, Practical considerations

Submit

Submit material you wish to deliver electronically via:

submit jkuczyns ps6 zip-or-tar-file

Files Needed

Here is a preconfigured workspace with the stage file (uml\_mcl) and two different starter implementations (no\_weights and with\_weights):

$ git clone http://[url-to-git-repo]

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

Overview

For this PS, you need to write a program that will localize the robot in the hallway using a particle filter.

There are two performance baselines.

Standard

Manually position the robot at one end of the hallway, aimed straight down the other end. Command the robot at a constant velocity and without any rotation.

Demonstrate visually that your particles indeed cluster around the robot.

Turn in a [screencast recording](https://wiki.ubuntu.com/ScreencastTeam/RecordingScreencasts) of the performance (both the Stage view and the particle cloud view)

Advanced

In addition to the above, create functionality that allows your robot to:

* discover its own localization position
* drive it to a specific position while avoiding obstacles (e.g., walls) on the way there

You should be able to start the robot in any location and orientation, and have it localize and drive itself to the position between the two red Xs on the Stage screen.

Record a screencast showing the performance.

To do this, you should only use the following ROS topics:

- read from /robot/base\_scan

- write to /robot/cmd\_vel

You can also read the image file "hallway.png".

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

***Warning -- below isn't fully tested***

Now, you can build the Python C extension to do raytracing on the image:

(roscd sample\_hw7/src/raycaster && make)

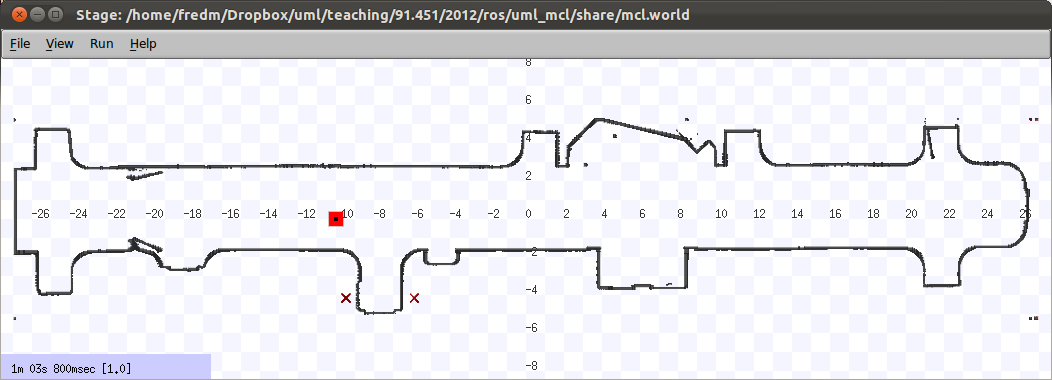
Demonstrating the Code

In a terminal shell, do

roscd uml\_mcl

roslaunch launch/mcl.launch

You should now see Stage running with a map, per below:

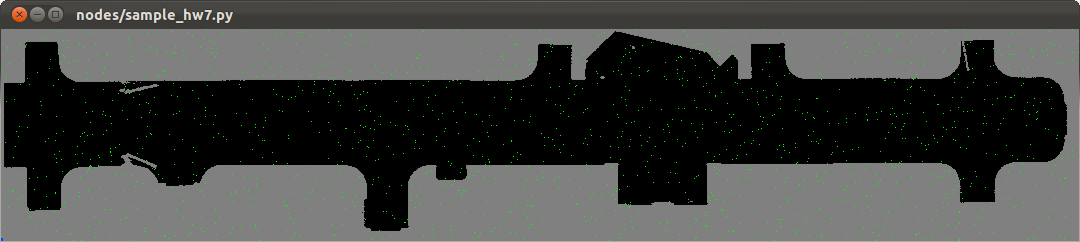


In another terminal shell, do

roscd sample\_hw7

nodes/sample\_hw7.py

You should see the OpenGL map showing particles scattering all around every time step, like this:



Some things to implement:

* Using the beam range finder model from the book (Table 6.1, pp. 158), compute a measurement correspondence for a given particle with the ray-tracing library (which uses the map). Pick constants for *z*hit, *z*short, *z*max, and *z*rand.
* Based on this measurement update, regenerate probabilistically with the robot just sitting there, and see if they converge to clusters. Move the robot around and see how the clusters re-form.
* Put the robot somewhere at the left edge of the map, heading east, and drive it at a constant velocity. Implement a movement update for the particles. See how the clusters behave now.

To turn in

* a write-up file containing your name and email address
* discussion
* code
* instructions on how to run the code

*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.*