Assignment 5: Localization
CS 393R: Robotics
Due Date: Thursday, October 31, 2013

Your task: Write the sensor update and re-sampling steps of a particle filter. Demonstrate the accuracy of your particle filter by adding functionality for your robot to walk to a point.

In this assignment you will be using your beacon detection and analysis skills from Assignment 4 to localize your robot. You will fill in the processFrame method in `core/localization/LocalizationModule.cpp` to look at position estimates (i.e. particles), and compute each particle's probability score based on the beacons you've observed that frame. You will need to implement resampling as well. Upon computing these scores, you will use the weighted average of your particles to determine your robot's position, and store this information in the `loc` member of the `WO_SELF` world object. You will then update your python code to direct the robot to walk toward the center of the field at (0,0) based on its current position that was supplied in the localization module.

You can debug your particle filter by viewing the World window in the tool. This window will show the robot's current pose, the positions of the beacons, the computed positions of any beacon sightings, and the pose of each particle. You can use the left/right arrow keys to move through the log.

You may notice that there is a `randomWalkParticles` function in the code. This function moves all of your particles a small amount randomly. This is not a traditional piece of a particle filter but can be a useful workaround for having a small number of particles available. You'll find that processing constraints (particularly lookups) limit the number of particles you can use.

You may want to use some of the existing code for `Point2D`, `Pose2D`, and `Vector2<float>` objects, which can be found in the `math` directory. There are also a number of parameters to play with, most of which are at the top of `localization/LocalizationModule.h`. Additionally, the number of particles is set in the `memory/LocalizationBlock.h` file.

Finally, the field layout is not fixed and may be changed when the deadline is closer. When accessing beacon locations, make sure you access them through the "loc" member in their associated `WorldObject`, rather than hard-coding the positions, so that your code can adapt to any changes in the field layout.

Checklist: (Scores will be out of 10)

[____] (2 points) Implement particle vision updates from known observations (i.e. the six beacons).

[____] (2 points) Implement and demonstrate particle re-sampling.

[____] (4 points) Demonstrate that your robot can localize to (0,0) (the center of the field). Once reaching the center of the field, it should continue to localize (and hence, walk in place). I will kidnap the robot a few times, and kidnappings may occur at any point (including once the robot is at the center of the field).

On each kidnapping, I will place the robot facing any direction I want and on any part of the field that I want (it will be able to see at least one beacon if it scans). After each kidnapping, the robot should relocalize and walk to the center of the field. I may take a log of your robot attempting to localize if it does not perform well.

[____] (2 points) Clarity and quality of your memo. Email it - along with any code files you changed - to Peter and Jake by class time on October 31.

Extra Credit:

[____] (1 point) Implement clustering or some other method to get the robot's pose instead of weighted averaging of the particles.