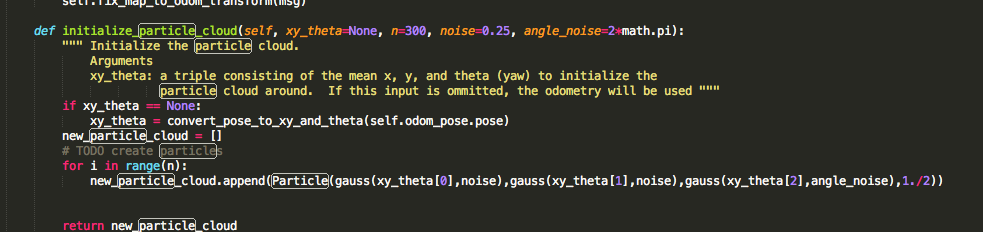
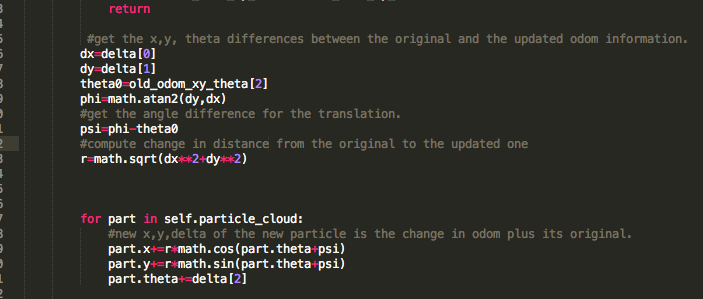
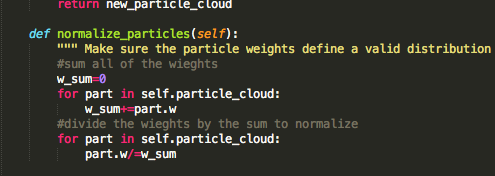
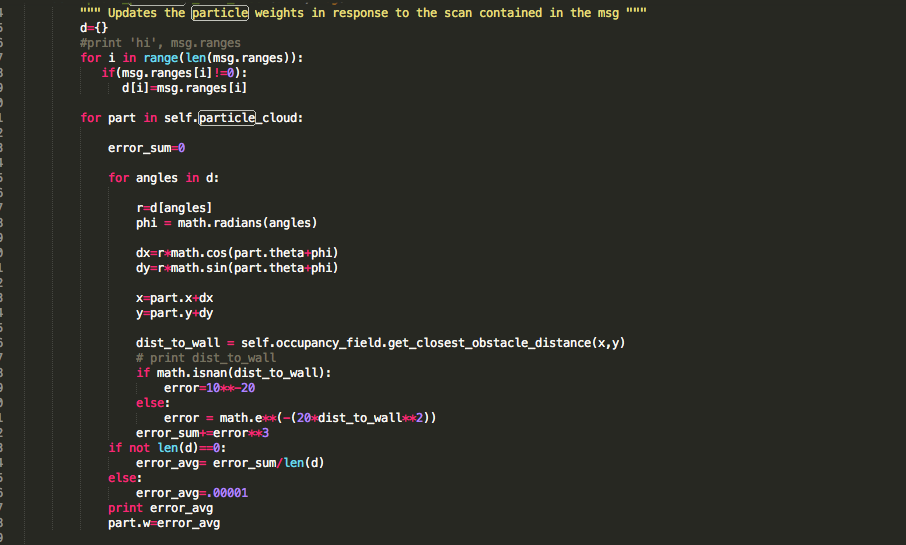
* **What was the goal of your project?**
* The goal of our project was to find the robot’s location using particle filter. Initially, the robot does not know where in the map it is. Our goal is to use the robot’s sensor input (through laser scan) to find out obstacles surrounding it and re-generate new particles based on their compatibility with the sensory input. Eventually, the particles will converge and we will have a good idea of where the robot is.
* Additionally, we had to worry about computational speed vs. accuracy. The more angles we scanned, the slower it will be for the algorithm to run.
* **How did you solve the problem? (Note: this doesn't have to be super-detailed, you should try to explain what you did at a high-level so that others in the class could reasonably understand what you did).**

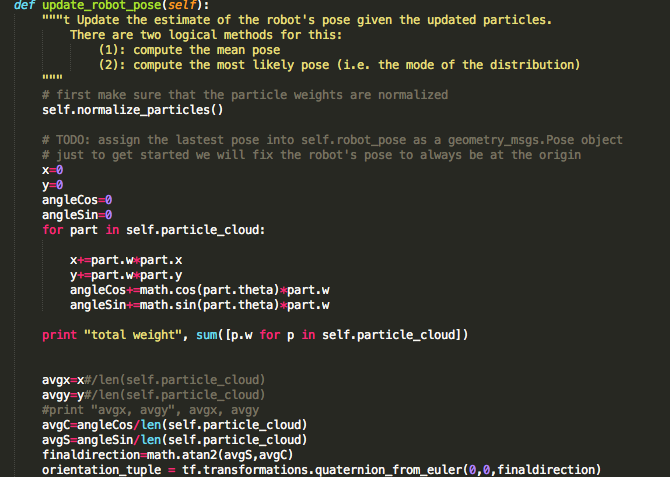
1. We first made 300 particles go through a for loops and append these particles to the particle cloud. In order to create some noise around the particle, we used Gaussian model to create the particles for it to have some noise but still center at our desired average.



1. We then had to update our particles when the robot moves. So we got the difference in the x and y direction to calculate the hypothesis it moved. And we also had to calculate the orientation it moved.



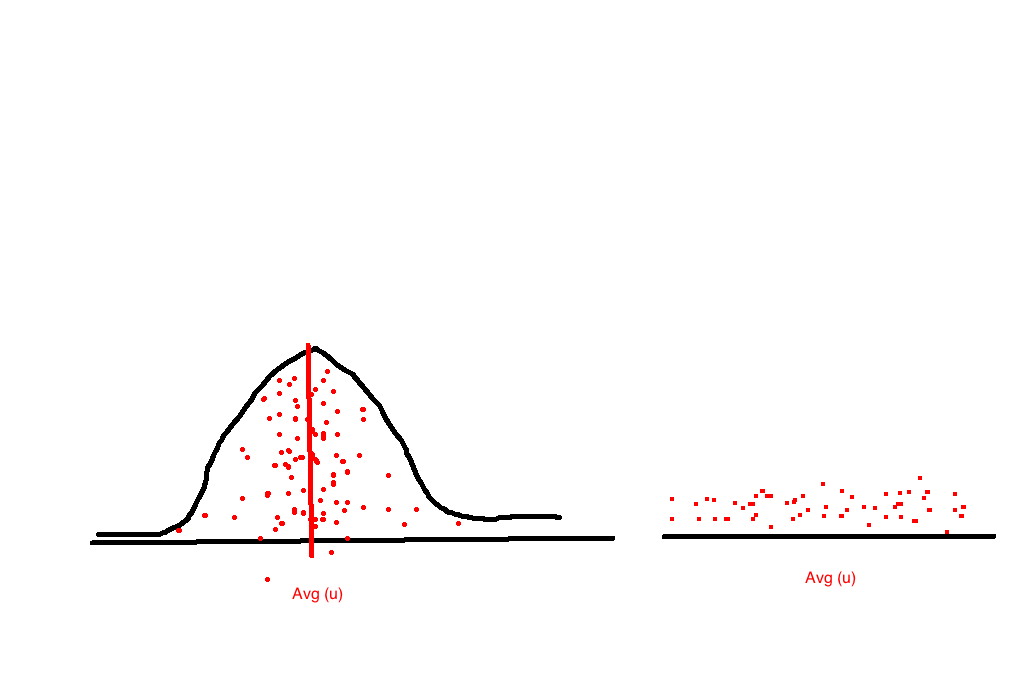
1. Then we need to make sure to normalize the particle weights, so that the weight actually all add up to one.
2. Base on laser range that the robot actually receives, we want to re-calculate the particles’ weight. Initially when the program first start, all particles had the same weight. Base on the laser range that are greater than 0, we figured out where the obstacle is relative to the robot. Then with this data, we assume that each particle is at the robot’s location and compute its obstacle location base on the robot’s location. Then we compared each particle’s obstacle location with actual obstacle location in the map with a given helper function “get\_closest\_obstacle”. Finally, we update the particle’s weight base on the particle’s obstacle location to the actual obstacle. The closer it is, the more weight the particle will have. 
3. Then we want to resample the particles base on their weight.
4. After getting the weight of each particle, we want to update the robot’s estimate (guess) position base on the predicted particle’s average. By this time, the particles should be close to the actual robot’s position, so take its average will give an estimate of the actual robot location. We had to make sure that we are not averaging the angles, but we should average the vectors of the angles (x direction and y direction).



* **Describe a design decision you had to make when working on your project and what you ultimately did (and why)? These design decisions could be particular choices for how you implemented some part of an algorithm or perhaps a decision regarding which of two external packages to use in your project.**

We had to make sure that when we average the angles, we are not averaging the actual angles, rather we need to average the vectors direction of the angles. This way, we get an accurate averaged angle. For example, the average angle of 10 and 350 is 180, but we actually want it to point to 360.

We also changed from using random sample to get the noise to using Gaussian to add noise around the particles. This way, the noise will not affect the actual average and our average is more accurate. See picture for details.



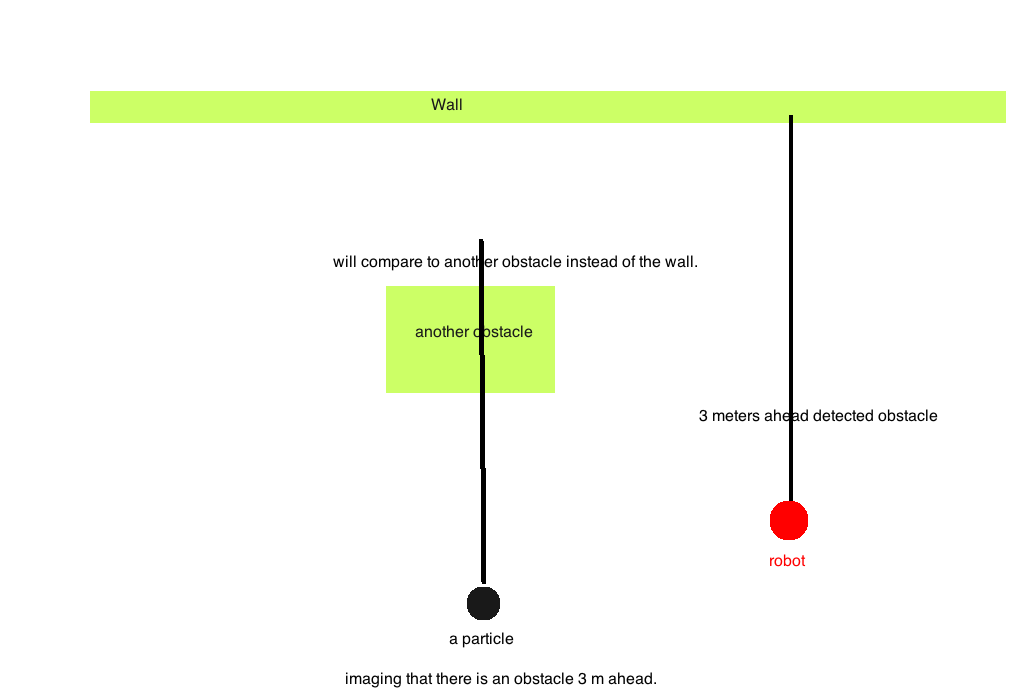
Additionally, we decided to receive laser scan measurement from all 360 degrees and take out 0 values. We understand that this might make the program slower, but we would like to have more accuracy.

* **What is any challenges did you face along the way?**

It was difficult to get started, since the problem seems challenging and long. It was helpful that we were provided a step by step guidance of what to do. Also when we were focusing on each step, it was hard to see the big picture of what is going on with particle filter.

* **What would you do to improve your project if you had more time?**

If we had more time, we probably want to take into account that some particles might have obstacles immediately in front of it and when implementing get\_closest\_obstacle”, the method will only return the closest obstacle that is not the wall. See picture for details.



* **Did you learn any interesting lessons for future robotic programming projects? These could relate to working on robotics projects in teams, working on more open-ended (and longer term) problems, or any other relevant topic.**

We learned some statistics such as Gaussian model and ways to translate from one location to another (translating from robot’s original location to its new location in update with odom). The math part of the problem was challenging but definitely worth learning!