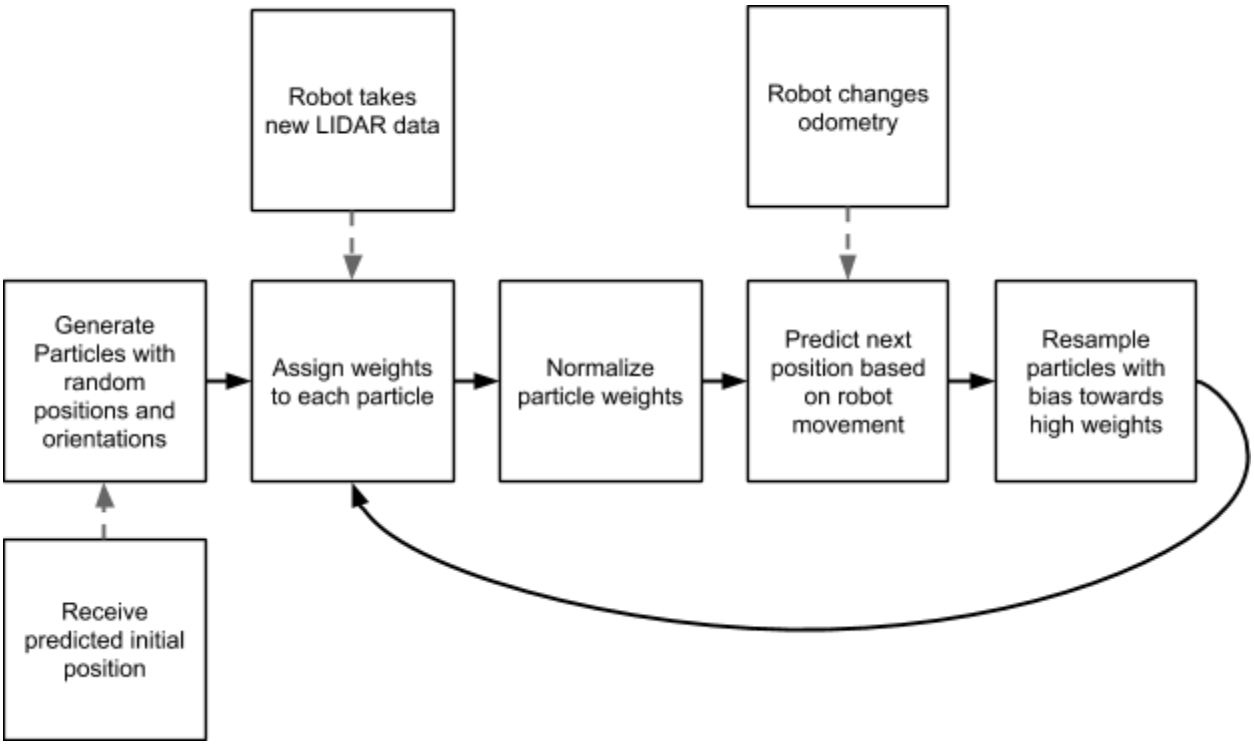
Robot Localization Writeup

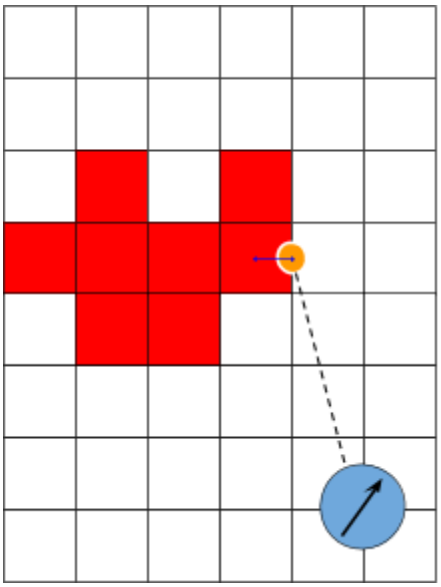
Anna Buchele and Ariana Olson

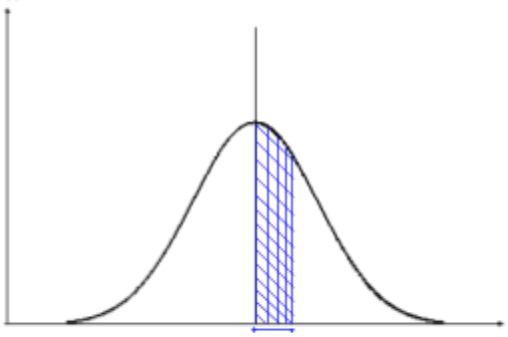
For this project, we used a particle filter to solve the problem of localization. By generating a mass of possible locations around an initial guess, and disqualifying, weighting, and regenerating possible locations as they become more or less likely, we can find the most likely locations of our robot on a map. We can use probability along with the sensor data and the odometry information to disquality each of the locations as we move, until we are left with only the most likely locations, centered around the real location.

We solved this localization problem by implementing a particle filter that validates and updates guesses of the location of the robot using the robot's laser scanner and an existing map of the area. As the particles move and the robot receives new data from the LIDAR, particles with the heaviest weight are chosen and replicated, moving in the predicted direction that the robot moved. By taking this approach, the particles converge to a small region and small set of positions the robot is likely to be in.



A high level block diagram of the particle filter implementation.





For each observation of a particle, the probability of the observation corresponding to the closest occupied cell in the given OccupancyField map is assessed based on the error between the observation and the actual location of the obstacle. The standard deviation of the probability density function is chosen to reflect error in the sensor readings.

One of the design decisions that we made for this project was to internally pass around the positions of the particles in (x, y, theta) format, rather than as a pose. We did this because we realized that the entirety of our algorithm internally required the particle poses to be transformed into this format anyways, so our system would be more efficient if we weren't transforming to and from poses needlessly. We also decided to make the "spread" of the original particles adjustable as well as the noise rates, because we noticed that the initial spread of the particles did have a large effect on the accuracy of our localization.

One of the challenges that we faced in this project was in predicting the next position of the particles after the robot's position had changed. Initially, we decided to update the position by blindly adding the change in odometry to each of the particles' previous positions. We found out that this approach was incorrect and pivoted to the approach in which we treat the movement of each particle as a rotation towards the destination point, a translation forward to the point, and a second rotation to turn towards the correct direction. Some of the challenge came from debugging the code to fix negative signs and angle normalizations, but a lot of it also came from the translation step. We initially just added the distance to translate to the x-component of the position, because the x-component on the neato is forward, but this meant that none of our transformations we touching the y-coordinate and our particles were not moving in the correct way. Keeping track of the odometry frame of each particle proved tough to think through, and it took a lot of whiteboard drawings for us to find the correct solution.

If we had more time, we would likely spend more time calibrating the noise parameters for better performance. We adjusted them to values where our system seemed to function “well enough”, but some more fine-tuning would definitely help our robot. We also noticed that our localization seems to run off the map over time, getting less and less accurate with further driving of the robot, so although initially it does a relatively good job of finding itself, over time the positioning becomes less accurate. If we had more time, we would debug further to figure out why our localization system does this, and try to fix it.

From this project, we learned the value of incremental development and testing as we went. We took the approach of writing most of the code before starting to test, which made it very hard to debug when we ran into problems, especially when it came to publishing visualizations. Had we focused on getting one particle published before creating several updating particles, we may have not gotten stuck for so long. From this challenge, we did learn much more about going between coordinate frames, which will definitely be useful in future projects.